# Single Input Multiple Output Media Based Modulation

by

Kartik Vamaraju

A thesis
presented to the University of Waterloo
in fulfillment of the
thesis requirement for the degree of
Masters of Applied Science
in
Electrical and Computer Engineering

Waterloo, Ontario, Canada, 2015

I hereby declare that I am the sole author of this thesis. This is a true copy of the thesis, including any required final revisions as accepted by my examiners.

I understand that my thesis may be made electronically available to the public.

# Abstract

Modulation is traditionally based on the idea that signal constellations should be deterministically constructed and known by the transmitter and receiver. Communication involves the transmitter randomly selecting from this known finite constellation. This is an example of Source Based Modulation (SBM). Media Based Modulation (MBM) is a departure from this paradigm. Tunable mirrors at the transmitter are used to establish independent channel realizations, the selection of which is used to encode information.

The main topic of this thesis is communication involving a Single-Input Multiple-Output (SIMO) MBM system communicating over a static Rayleigh fading channel when there is perfect Channel State Information (CSI) at the receiver. Simulation results are presented to demonstrate various aspects of MBM system performance that are different from comparable SBM systems communicating over an Additive White Gaussian Noise (AWGN) channel. The application of channel coding to MBM is then discussed and simulation results involving the application of a Single Parity Check (SPC) symbol code applied to an MBM system are presented. The geometry of MBM constellations has a significant impact on coding gain, and consequently the coding gains with MBM are different then with SBM. Finally, a novel algorithm is developed to solve the Maximum Likelihood (ML) symbol detection algorithm for MBM using ideas from sphere decoding. Various methods of improving computation speed at a cost of introducing approximation error are also presented. An approximate ML symbol detection algorithm is presented in which the search radius is determined by the $L_\infty$ norm and the optimal candidate selection is determined by the $L_2$ norm. Simulation results demonstrate the reduction in search effort by using the approximate algorithm.

# Acknowledgments

This thesis would not have been possible without the help of my supervisor Dr. Amir K. Khandani and my thesis reviewers.

# Dedication

This is dedicated to my family.

# Table of Contents

# List of Figures

# Chapter 1

# Introduction

In this chapter, the idea of Media Based Modulation is motivated and relevant background information is presented. An outline of the topics discussed in subsequent chapters is also provided.

## 1.1 Overview

Consider a digital communication system in which a single transmitting antenna sends information over an Additive White Gaussian Noise (AWGN) channel to a receiver with $Q$ receiving antennas. If $Q = 1$, the system is Single-Input Single-Output (SISO) and if $Q > 1$, the system is Single-Input Multiple-Output (SIMO). The SIMO case can be described by the following equation:

$$\mathbf{Y}[t] = \mathbf{1}X[t] + \mathbf{N}[t], \tag{1.1}$$

where $\mathbf{1}$ is a column vector with each entry being 1, $X[t]$ is a transmitted information value selected from a finite alphabet, $\mathbf{N}[t]$ is an additive white Gaussian noise vector, $\mathcal{N}(0, \sigma_n^2 I)$ and $\mathbf{Y}[t]$ is the noisy vector observed by the receiver. It is assumed that the information symbols are all equally likely to be sent.

The random value $X[t]$ is the result of modulation, which is the process of mapping a finite alphabet of information symbols into a constellation of vectors that are defined with respect to a basis of time-domain signals. The constellation of vectors is known by both the transmitter and receiver, and the scheme can be further categorized as Source-Based Modulation (SBM)[1] when the design of the constellation set is determined entirely

by the transmitter. The transmitter uses modulation to construct a time-domain signal to send over the channel and the receiver demodulates the received signal to construct a noisy observation vector. At this level of abstraction, the details of the time-domain signals and the modulation and demodulation process are represented only by the choice of constellation vectors and the addition of noise.

When the vector $\mathbf{Y}[t]$ is received, Maximum Likelihood (ML) symbol detection is used to estimate which symbol the transmitter most likely sent. The presence of noise from the channel introduces a non-zero probability that the receiver makes an incorrect decision based on the observed vector. The primary goal of the communication system is to maximize the rate at which information can be sent from the transmitter to the receiver while minimizing the probability that information is transferred incorrectly.

It is known that in order to achiveve the capacity bound, that is to maximize the amount of information transferred during the communication process, $\mathbf{1}X[t]$ should be statistically equivalent to a Gaussian random vector. Unfortunately, this is easier said than done[2].

At first glance, this could be accomplished simply by using microwave engineering to directly construct a constellation set that behaves like a set of Gaussian random vectors. However, the microwave engineering required for this strategy is usually infeasible. A second approach is to use a simple geometric construction for the constellation and to transmit each constellation vector with a probability such that over the finite alphabet the behavior appears Gaussian. By appropriately selecting the transmission probability and incorporating coding, it is possible to arbitrarily approach the capacity bound[3]. A third approach is to use coding to establish Gaussian codebooks, which exhibit the same potential for achieving capacity as Gaussian random vectors for an AWGN channel in the limit as the number of transmitted vectors approach infinity[4].

Wireless communication systems involving a single transmitting antenna and $Q$ receiving antennas usually satisfy the more general equation:

$$\mathbf{Y}[t] = \mathbf{H}[t]X[t] + \mathbf{N}[t], \tag{1.2}$$

where the channel $\mathbf{H}[t]$ is a random vector that represents the interaction of the time domain electromagnetic waveform with the environment. If it is assumed that the channel exhibits static Rayleigh fading (SRF), then $\mathbf{H}[t]$ is constant for a fixed time interval, is stationary for all time and contains i. i. d. circularly complex Gaussian entries with $\mathcal{N}(0, 1)$. This fading model can be considered a piecewise continuous approximation to slow fading over a small time-interval.

In Media Based Modulation (MBM), tunable mirrors are used to change the electromagnetic properties of the signal $X[t]$ as it leaves the transmitter[1]. Small perturbations

in the near field of the transmitter can result in noticeable, independent changes in the channel $\mathbf{H}$. Multiple i. i. d. realizations of $\mathbf{H}$ can be established by controlling the configurations of the tunable mirrors. It is therefore possible to encode information by selecting which channel realization is used at a particular time.

This method of modifying the channel or embedding information in channel selection is different from prior work. See the references in [1] for more details. The closest existing modulation scheme to MBM is Generalized Spatial Modulation (GSM), which involves transmitting information symbols using subsets of multiple transmitting antennas[5]. However, all of the transmitting antennas in GSM send information over a single channel realization whereas MBM involves sending information by selecting from multiple independent channel realizations.

Also, if $X[t] = 1$, then all the information is carried in the selection of $\mathbf{H}$, which is a Gaussian random vector. Therefore, it is expected that MBM can achieve the capacity of the AWGN channel model without requiring the use of coding. Indeed, it can be shown that the capacity of MBM over an SRF channel is equivalent to that of an AWGN channel[6].

This thesis investigates issues that arise with SIMO wireless communication systems as a result of using MBM instead of SBM. It is assumed that the MBM system communicates over an SRF channel. In particular, the following issues are discussed:

- Unlike constellations used in SBM, constellations used in MBM are at least partially random. The simulated performance of MBM constellations is discussed in Chapter 2. Performance in the context of this thesis usually refers to the Symbol Error Rate (SER) of a system as a function of the constellation size, the number of receiving antennas and the quantity $\frac{E_b}{N_0}$ measured in dB.

- When coding is considered, the geometry of MBM constellations results in fundamentally different behavior from the classical results expected from SBM. The impact of this geometry on coding is discussed in Chapter 3.

- It is necessary to develop new algorithms to solve the ML symbol detection problem in MBM due to the inherent randomness of the MBM constellations and the reality that both the constellation size and $Q$ are potentially large. This is discussed in Chapter 4.

Moving forward, one obvious question is how the use of MBM changes when multiple transmitting antennas are used. In SBM, there are nontrivial opportunities to improve communication system performance that are introduced specifically through the use of multiple

antennas. Consider for example, the use of space-time coding[7] to improve performance in Multi-Input Multi-Output (MIMO) systems or the leveraging of the structure of MIMO channels to improve the quality of channel estimation for time-varying environments[8]. The scope of this thesis is restricted to SIMO systems, and hence the majority of the topics introduced by using MIMO systems are left for future research. The issue of multi-user communication and interference management is also beyond the scope of this thesis, however it is unlikely that MBM will introduce any complications beyond what is already present with SBM.

MBM represents a paradigm shift in the theory of modulation for wireless communication. By using MBM, it is possible to significantly outperform the SBM systems that are currently being used. As shown in subsequent chapters, the performance of MBM constellations constructed using a SRF channel with respect to error rate, the effectiveness of coding and the expected complexity of the ML symbol detection problem all vary strongly with the number of antennas $Q$. From a system design perspective, there are significant differences between operating in the low dimensional regime ($Q < 4$) and the high dimensional regime $Q > 8$, and $4 \leq Q \leq 8$ appears to provide a reasonable range for current practical applications.

# Chapter 2

# MBM Performance

This chapter presents theoretical and simulated results to characterize MBM systems and to compare them to SBM systems under similar conditions.

## 2.1 Background

The performance of an SIMO wireless communication system can be characterized by using simulation to estimate the received data error rate as a function of transmission power, noise and the number of receiving antennas. In general, the decoder structure used at the receiver has a noticeable impact on the error rate. However in this chapter no coding is used and the channel is memoryless, so there is no advantage to using any technique that differs from ML symbol detection.

One common parameter in the performance analysis of communication systems is the Signal to Noise Ratio (SNR), which is the ratio of the average signal energy $E_s$ to the noise power $\sigma_n^2$. Let $E_b$ be the average energy per bit of the signal constellation and $N_0 = \frac{\sigma_n^2}{2}$ by convention. The definition of SNR (or equivalently $\frac{E_b}{N_0}$) in SBM remains unchanged regardless of whether the transmitter or receiver is used as a reference. In practice, the SNR is usually calculated with respect to the receiver since that is where the SNR is usually measured. In MBM however the choice of reference makes a difference and it is assumed that SNR measurements are always made with respect to the transmitter. Adding additional antennas at the receiver does not change the amount of transmitted signal power and hence does not impact the calculation of $\frac{E_b}{N_0}$. When transmitting information by changing the channel realization, it is more practical to define SNR at the transmitter,

since at a given time different realizations will cause different levels of received signal energy at the receiver even if the average transmission energy is constant.

## 2.2 Simulation Results

Simulation results are shown below to characterize the error rate as a function of $\frac{E_b}{N_0}$ of various MBM systems communicating over an SRF with perfect CSI at the receiver. Symbol Error Rate (SER) rather than Bit Error Rate (BER) is shown since the relationship between the two is affected by bit labeling, which in this case is random. Unlike SBM, it is not feasible to implement any form of Gray labeling to minimize the BER since the transmitter does not know the constellation structure.

The MBM constellations are constructed by fixing an SBM constellation and multiplying the set of constellation vectors by different independent channel realizations of an SRF channel. The SER is calculated for multiple different blocks of $P2^n$ symbols, each of which contains $M$ symbols for each constellation vector and each block corresponds to a different channel realization. The average SER over many blocks and sufficiently large $M$ characterizes the expected SER for communication over a block Rayleigh fading channel with stationary statistics. Since the channel is memoryless and stationary over the block interval, the order in which information symbols are sent within a block is irrelevant and an accurate SER can be found using ML symbol detection. The SER is an ensemble average over many different realizations, and in this characterization there is no notion of outages.

Consider two comparable SBM and MBM systems. The SBM system transmits information using a constellation $\mathcal{C}_1$ that involves $2^N = 256$ equally spaced rectangular constellation vectors. Each constellation vector has 2 dimensions which are generated from the set:

$$x_i = \pm 1, \pm 3, \ldots, \pm(\sqrt{2^N} - 1), \tag{2.1}$$

multiplied by a normalization factor $\sqrt{\frac{3}{(2^N-1)}}$ per dimension to account for the total average energy of the signal set. The set is extended to include $Q$ receiving antennas according to the SRF channel Equation 1.2, where the total number of dimensions received is $K = 2Q$.

The MBM system transmits information using a constellation $\mathcal{C}_2$ that is constructed from 256 independent realizations of an SRF channel. The receiver uses ML symbol detection to search amongst the constellation vectors to determine which constellation vector was most likely sent in both the SBM and MBM systems. The SER of the SBM system with $\mathcal{C}_1$ and $Q$ receiving antennas communicating over an AWGN channel and an SRF

channel is shown in Figure 2.1. The SER of the MBM system with $\mathcal{C}_2$ and $Q$ receiving antennas communicating over an SRF channel is shown in Figure 2.2. A comparison of the SER for the SBM and MBM systems with $Q = 4$ and $Q = 8$ receiving antennas is shown in Figure 2.3.

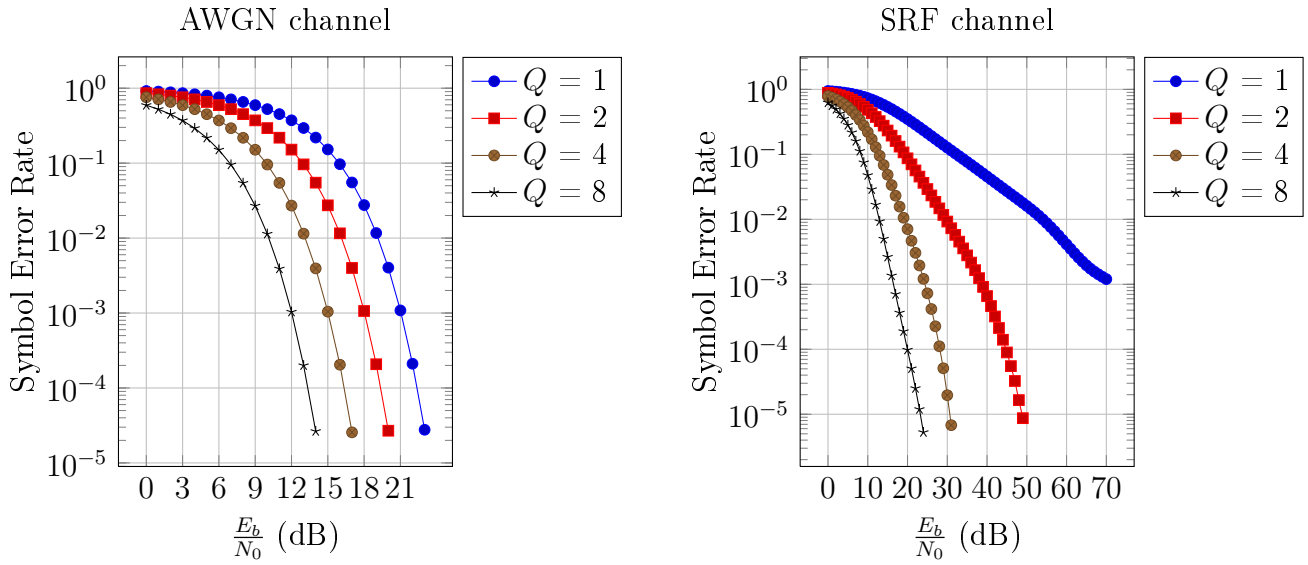


Figure 2.1: SER vs $\frac{E_b}{N_0}$ for an SBM system using a 256 symbol rectangular constellation, $\mathcal{C}_1$ and $Q$ receiving antennas communicating over an AWGN channel or SRF channel.

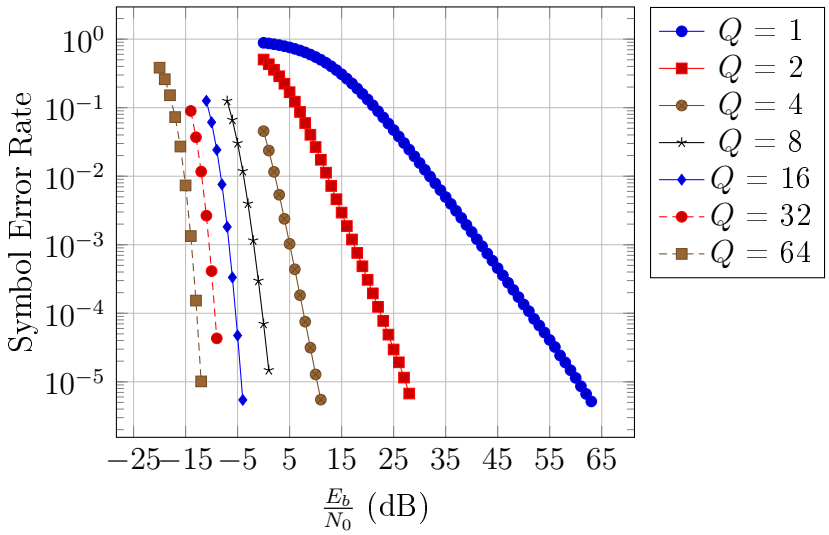


Figure 2.2: SER vs $\frac{E_b}{N_0}$ for an MBM system using 256 realizations of an SRF channel, $\mathcal{C}_2$ and $Q$ receiving antennas.

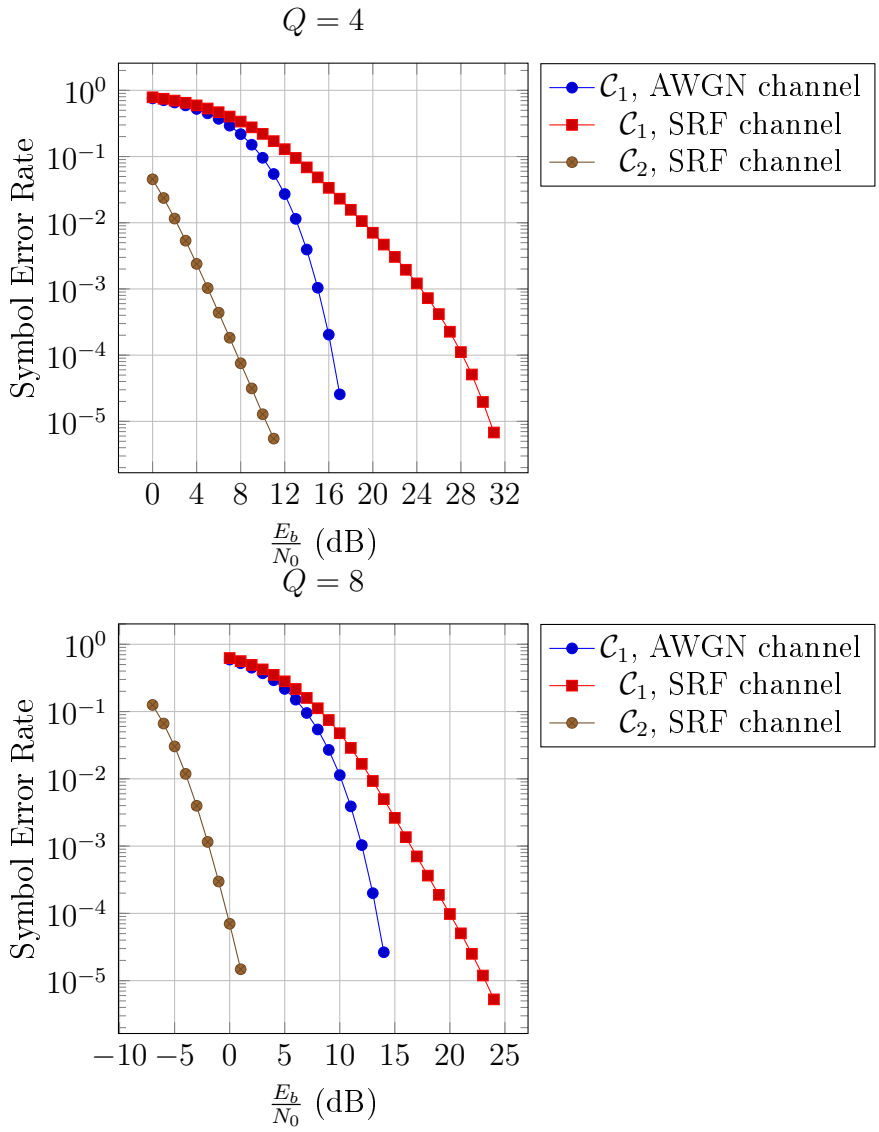


Figure 2.3: A comparison of SER vs $\frac{E_b}{N_0}$ for an SBM system with $\mathcal{C}_1$ and an MBM system with $\mathcal{C}_2$ and $Q$ receiving antennas communicating over an AWGN channel or an SRF channel.

Figure 2.1 demonstrates the fact that SRF significantly degrades the performance of an SBM system compared to when an AWGN channel is used. The presence of fading changes the geometry of the SBM constellations and a system with few receiving antennas is not necessarily comparable to a system with many receiving antennas.

As shown in Figure 2.2, fading statistics also result in this performance scaling property in MBM. An MBM system with at least 4 receiving antennas can exceed the performance of the comparable SBM system even if the SBM system is communicating over an AWGN channel When the systems are compared in Figure 2.3. It is therefore of significant interest to consider MBM systems with at least 4 receiving antennas in order to capture the majority of the potential performance advantages.

To show how the SER changes with constellation size, the average SER for an MBM system with a 65536 symbol constellation with 65536 channel realizations respectively, one transmitting antenna and $Q$ receiving antennas is shown Figure 2.4 and a comparison of the 65536 and 256 symbol MBM systems is shown in Figure 2.5.

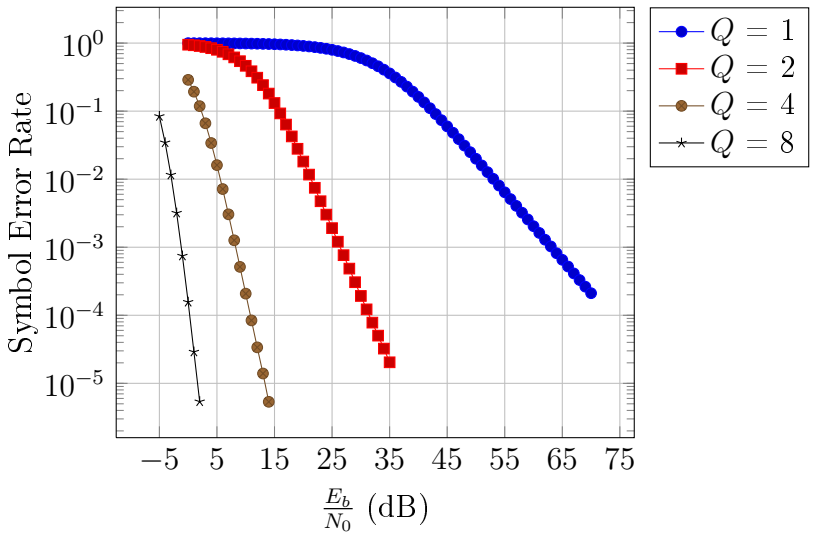


Figure 2.4: SER vs $\frac{E_b}{N_0}$ for an MBM system using 65536 independent realizations of an SRF channel and $Q$ receiving antennas.

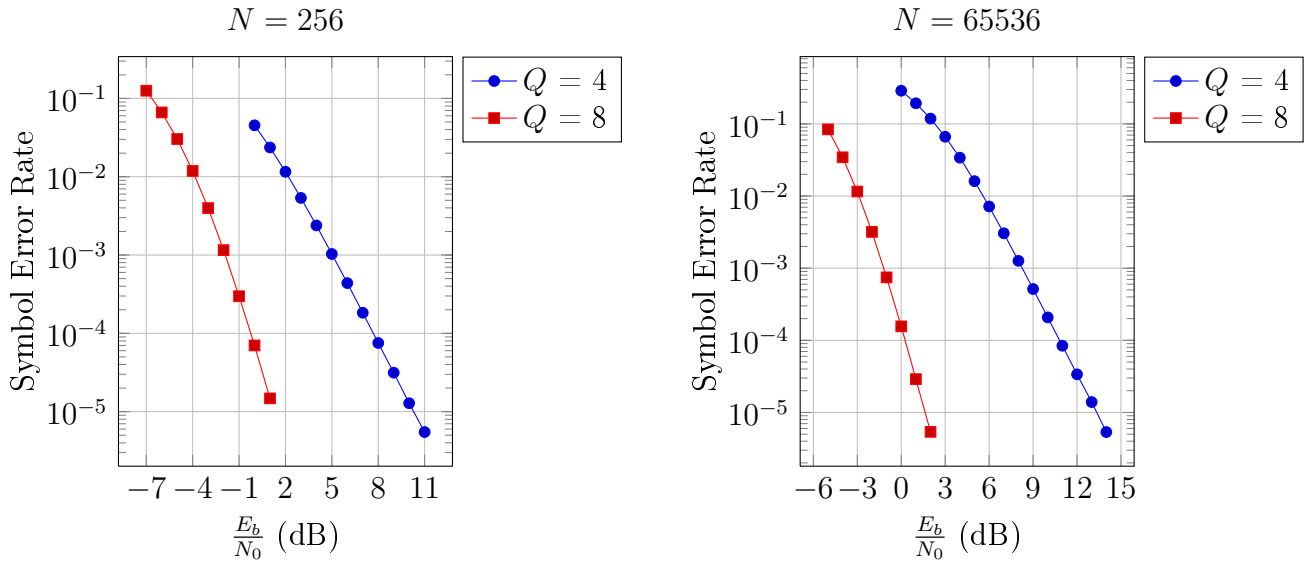


Figure 2.5: SER vs $\frac{E_b}{N_0}$ for an MBM system using $N$ independent realizations of an SRF channel and $Q$ receiving antennas.

As the number of receiving antennas increases, the energy loss experienced when doubling the number of bits/antenna in the constellation decreases, indicating that there is an opportunity to use larger constellation sizes with MBM without significant energy loss compared to SBM when sufficient receiving antennas are present.

Note that for an MBM system that encodes all of its information in the selection of channel realization, the components of each constellation vector are independent. This implies that the pairwise differences between the constellation vectors are also independent. Therefore, as the number of dimensions increases, the probability that any two constellation vectors will be close together approaches zero. The independence of the distances results is one of the primary properties that distinguishes SBM constellations from MBM constellations. The fact that this distance is large when many receiving antennas are used is responsible for MBM achieving the capacity of an AWGN channel.

# Chapter 3

# Coding

This chapter presents results relating channel coding to MBM. Simulation results for a simple Single Parity Check symbol code applied to MBM are presented.

## 3.1 Background

Channel coding is used to reduce error rates and provide energy gains at a cost of reduced communication rate. It is particularly important in scenarios where the channel statistics would normally prevent the reliable transmission of information[9]. In the context of wireless communication, channel coding is either designed independent of the modulation scheme or in conjunction with it. For example, many block and convolutional codes are designed assuming that the constellation size is 2 even though the actual constellation size may be much larger. Trellis Coded Modulation (TCM)[10][11][12] on the other hand relies on set partitioning and the explicit structure of the constellation to achieve coding gains.

SIMO MBM has several features that complicate the design and analysis of codes. In particular:

- The behavior of MBM depends strongly on the number of receiving antennas. This dependence is fundamentally different from SBM systems transmitting over an AWGN channel and complicates the use of common channel models used in conjunction with coding such as the Binary Symmetric Channel (BSC) or Discrete Memoryless Channel (DMC).

- The transmitter does not know the geometry of the constellation a priori. As such set partitioning and TCM is completely infeasible. Moreover, the exact transition probability is not known either for the purposes of code design and at best only a statistical model is known.

Together these features significantly limit the possibilities of the code designer, despite the desire to design codes specifically for MBM constellations. With this in mind, the simplest way to establish a base level of performance for coding with MBM is to design and implement a symbol code that does not account for the geometry of the constellation and compare its performance against a comparable SBM system. It might be expected that since the code does not take into account constellation structure, its coding gain would be roughly the same with SBM and MBM. Simulation results for a Single Parity Check (SPC) symbol code applied to an MBM system are shown in Section 3.2. These results demonstrate that the coding gains with MBM can be dramatically different than with SBM despite the generic nature of the code design.

## 3.2 Single Parity Check Symbol Code

Suppose a transmitter using MBM sends a block of $M$ information symbols to a receiver with $Q$ receiving antennas, and hence $K = 2Q$ receiving dimensions. The sequence of information symbols in the block is $i_1, i_2, \cdots, i_M$. The constellation size is $2^N$. The transmission of the block can be expressed in matrix form by the equation:

$$\mathcal{C} = \left(\mathbf{c}^{i_1}, \mathbf{c}^{i_2}, \cdots, \mathbf{c}^{i_M}\right) = \begin{pmatrix} c_1^{i_1} & c_1^{i_2} & \cdots & c_1^{i_M} \\ c_2^{i_1} & c_2^{i_2} & \cdots & c_2^{i_M} \\ \vdots & \vdots & \ddots & \vdots \\ c_K^{i_1} & c_K^{i_2} & \cdots & c_K^{i_M} \end{pmatrix} \tag{3.1}$$

where $\mathcal{C}$ is the block of constellation vectors that is sent, $\mathbf{C}^{i_m}$ is the constellation vector corresponding to information symbol $i_m$ for $1 \leq m \leq 2^N$, and $c_k^{i_m}$ is the component of the constellation vector $\mathbf{C}^{i_m}$ along dimension $1 \leq k \leq K$.

Consider a Single Parity Check (SPC) symbol code with block size $M$. The transmitter sends $M-1$ information symbols and selects the last information symbol such that:

$$\sum_{m=1}^{M} i_m \text{ modulo } 2^N = 0. \tag{3.2}$$

Ideally, the receiver would use a trellis based decoder to realize ML sequence detection. To reduce the complexity of this decoder, the receiver could instead use the following linear-complexity soft-decision decoder:

1. Perform ML symbol detection on each symbol. Check if the parity of the received block is 0. If true, then assume the block is correct.

2. If the parity is nonzero, assume exactly 1 error has occurred. For each symbol within the block, determine the correction candidate such that the block parity is 0 and store the minimum distance associated with this candidate. Select the replacement candidate that has the minimum distance among the $M$ candidates. Replace the ML candidate with the selected replacement and then assume the block is correct.

It can be shown that the probability of multiple errors occurring in a single block is negligible in the limit as the symbol error rate approaches 0. This implies that this suboptimal decoder will closely approximate the optimal decoder performance when the uncoded symbol error rate is sufficiently low, which occurs when $\frac{E_b}{N_0}$ is sufficiently high.

The SPC code can correct 1 error per block reliably. Suppose an error event $E$ occurs after the decoder has attempted error correction. In this case, suppose $i_n$ and $i_m$ are sent and $i'_n$ and $i'_m$ are received for $n \neq m$. This corresponds to error events $E_n$ and $E_m$. The decoder attempts to perform error correction by selecting some index $i_k$ and modifying it to $i'_k$ such that the parity check is satisfied.

If $k \neq n$ and $k \neq m$, then the following relation holds:

$$((i_n - i'_n) + (i_m - i'_m) = (i'_k - i_k)) \text{ modulo } 2^N. \tag{3.3}$$

The Euclidean distances $\|\mathbf{c}^{i_l} - \mathbf{c}^{i'_l}\|^2$ for some $l = n, m, k$ depends on the specific messages $i_l$ and $i'_l$, not the difference modulo $2^N$ between the messages. Since the constellation consists of arbitrarily labeled vectors rather than Gray labeled vectors, restricting the difference $(i_l - i'_l)$ modulo $2^N$ does not constrain $\|\mathbf{c}^{i_l} - \mathbf{c}^{i'_l}\|^2$ when $1 \leq i_l \leq 2^N$ is arbitrary. Consequently, the Euclidean distance corresponding to error event $E$ consists of the sum of at least two independent Euclidean distances corresponding to error events $E_n$ and $E_m$. Recall that the coding gain for an arbitrary code is a function of the total distance between codewords.

Consider an error event in which message $i_n$ is mistakenly decoded as message $i'_n$ and the constellation has only a single dimension for simplicity. For a fixed set of channel realizations, each of the constellation values have a Gaussian distribution with $\mathcal{N}(0, E_s)$

due to the fading statistics of a static Rayleigh fading channel. The distance between two Gaussian random variables would follow a Gaussian distribution with $\mathcal{N}(0, 2E_s)$. Since this distance is random, the probability of the error event is a function of this random distance and is also random. The probability of error for the two constellation values would be:

$$P(E_n) = \frac{1}{2} erfc\left(\sqrt{\frac{E_s}{N_o}} X\right), \tag{3.4}$$

where $X$ is a Half-Normal random variable with unit variance.

The average probability of error for the constellation is:

$$\begin{aligned}
\mathbb{E}[P(E_n)] &= \frac{\sqrt{2}}{2\sqrt{\pi}} \int_0^\infty erfc\left(\sqrt{\frac{E_s}{N_o}} x\right) e^{\frac{-x^2}{2}} dx \\
&= \frac{1}{2} - \frac{\sqrt{2}}{2\sqrt{\pi}} \int_0^\infty erf\left(\sqrt{\frac{E_s}{N_o}} x\right) e^{\frac{-x^2}{2}} dx \\
&= \sqrt{\frac{2}{\pi}} \tan^{-1}\left(\sqrt{\frac{N_o}{2E_s}}\right) \\
&\approx \mathcal{O}\left(\left(\frac{N_o}{E_s}\right)^{\frac{1}{2}}\right), \qquad (3.5) \\
& \qquad (3.6)
\end{aligned}$$

where the last equality is obtained using an integral table[13] and the approximation is obtained using a power series expansion of $\tan^{-1}$.

In MBM, the constellation vectors have independent components. For a SIMO MBM system with $Q$ antennas and $K = 2Q$ dimensions, the average probability of error for two constellation vectors is about:

$$\mathbb{E}[P(\mathbf{E}_n)] = \mathcal{O}\left(\left(\frac{N_o}{E_s}\right)^{Q}\right). \tag{3.7}$$

Since an error event after applying the code involves the sum of at least two independent errors, the average probability of error after decoding is:

$$\mathbb{E}[P(\mathbf{E}_n, \mathbf{E}_m)] = \mathcal{O}\left(\left(\frac{N_o}{E_s}\right)^{2Q}\right) = \mathcal{O}\left(\left(\frac{E_s}{N_o}\right)^{-2Q}\right). \tag{3.8}$$

This implies that as a result of applying the code, there is an average diversity gain of at least 2 due to the independence of the pairwise distances in the MBM constellation structure. An SIMO MBM system has a diversity gain of $Q$ without the application of the code. Note that the assumption that the error event $E$ consists of 2 independent errors $E_n$ and $E_m$ is only valid if $\frac{E_b}{N_o}$ is sufficiently high, but the multiplexing gain of $Q$ is always present independent of $\frac{E_b}{N_o}$ due to the nature of SIMO MBM. If a symbol code that corrects $p$ errors is used with MBM, the code would result in a diversity gain of $p+1$ if any uncorrected error events have the same independence property as the SPC code.

Simulation results involving an SBM system using constellation $\mathcal{C}_1$ consisting of 256 equally spaced rectangular constellation vectors with $Q$ receiving antennas, a block size of $M$ and the suboptimal decoder communicating over an AWGN channel is shown in Figure 3.1. Simulation results involving an MBM system using constellation $\mathcal{C}_2$ consisting of 256 different channel realizations of a SRF channel with $Q$ receiving antennas, a block size of $M$ and the suboptimal decoder are shown in Figure 3.2 and Figure 3.3.

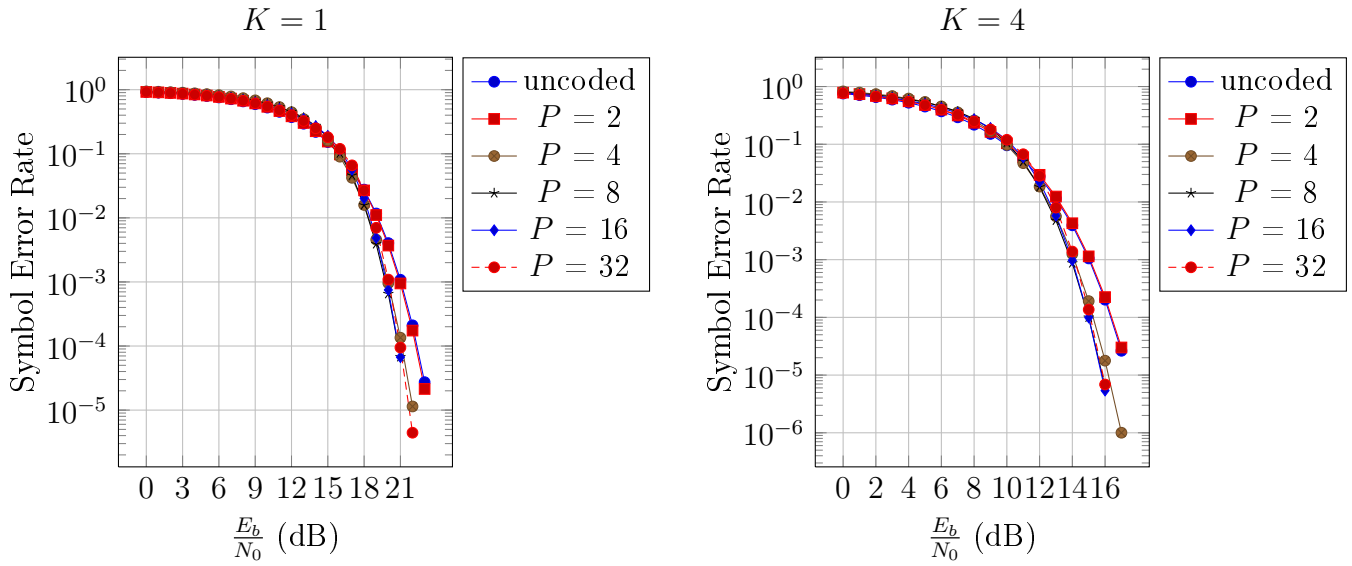


Figure 3.1: SER vs $\frac{E_b}{N_0}$ for an SBM system using a 256 symbol rectangular constellation, $K$ receiving antennas and an SPC symbol code with block size $P$ communicating over an AWGN channel.

Simulation results involving an MBM system using a constellation constructed from 65536 different channel realizations of an SRF channel with $K$ receiving antennas, a block size of $P$ and the suboptimal decoder are shown in Figure 3.4 and Figure 3.5.

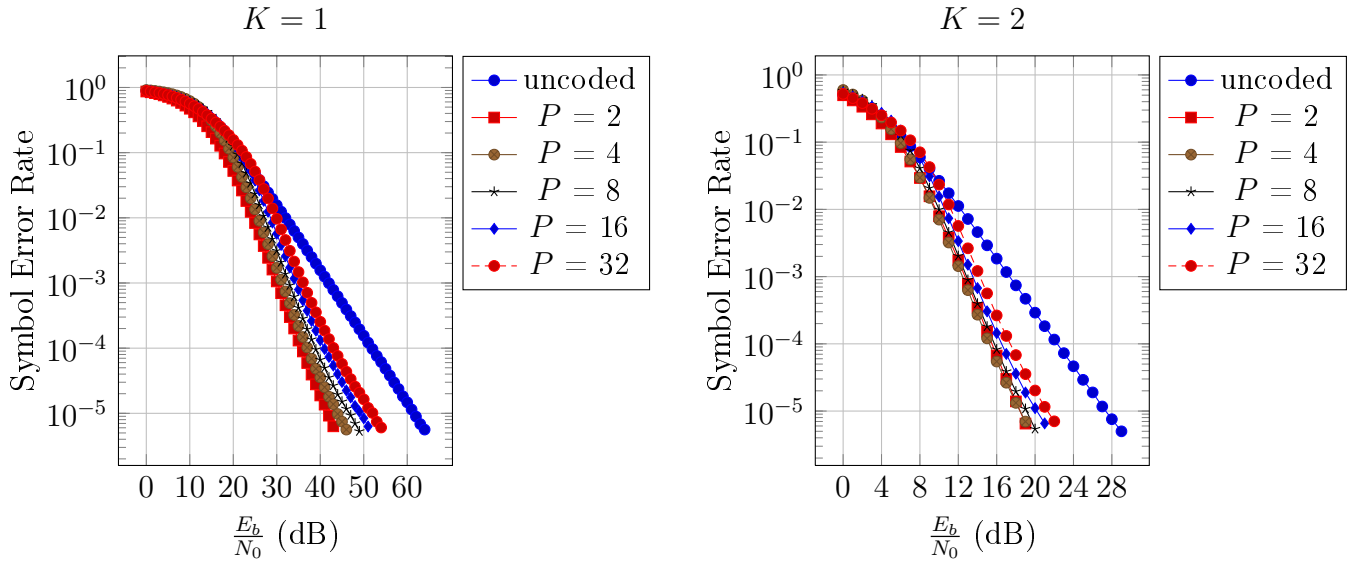


Figure 3.2: SER vs $\frac{E_b}{N_0}$ for an MBM system using 256 realizations of an SRF channel, $K = 1$ or $K = 2$ receiving antennas and an SPC symbol code with block size $P$.

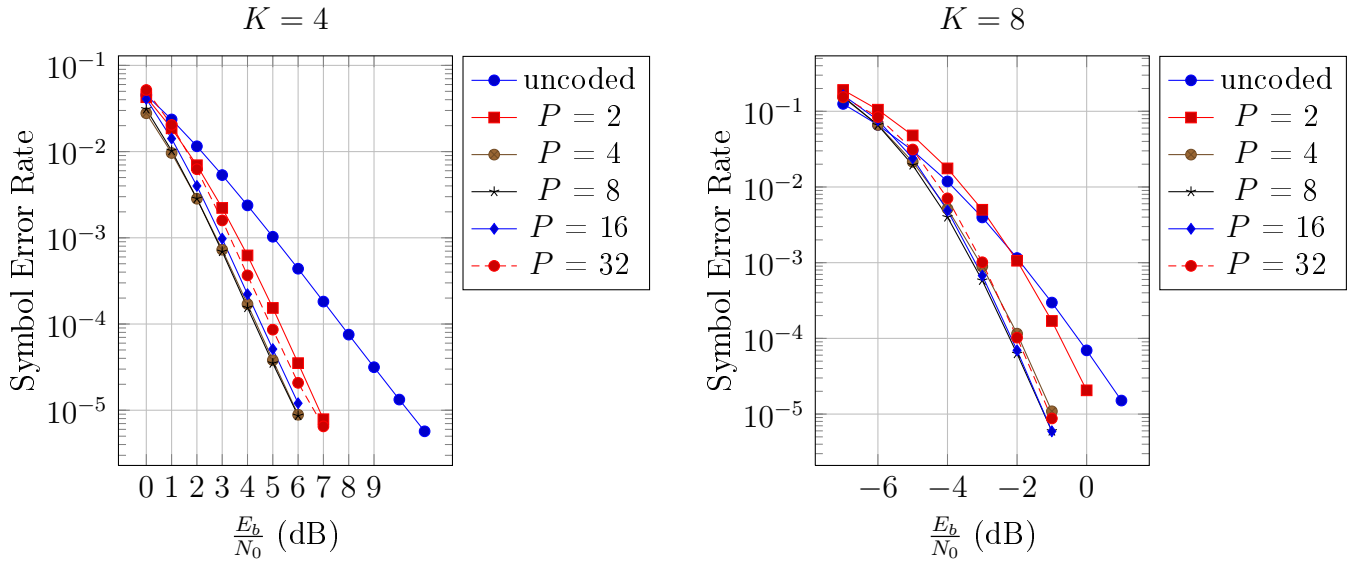


Figure 3.3: SER vs $\frac{E_b}{N_0}$ for an MBM system using 256 realizations of an SRF channel, $K = 4$ or $K = 8$ receiving antennas and an SPC symbol code with block size $P$.

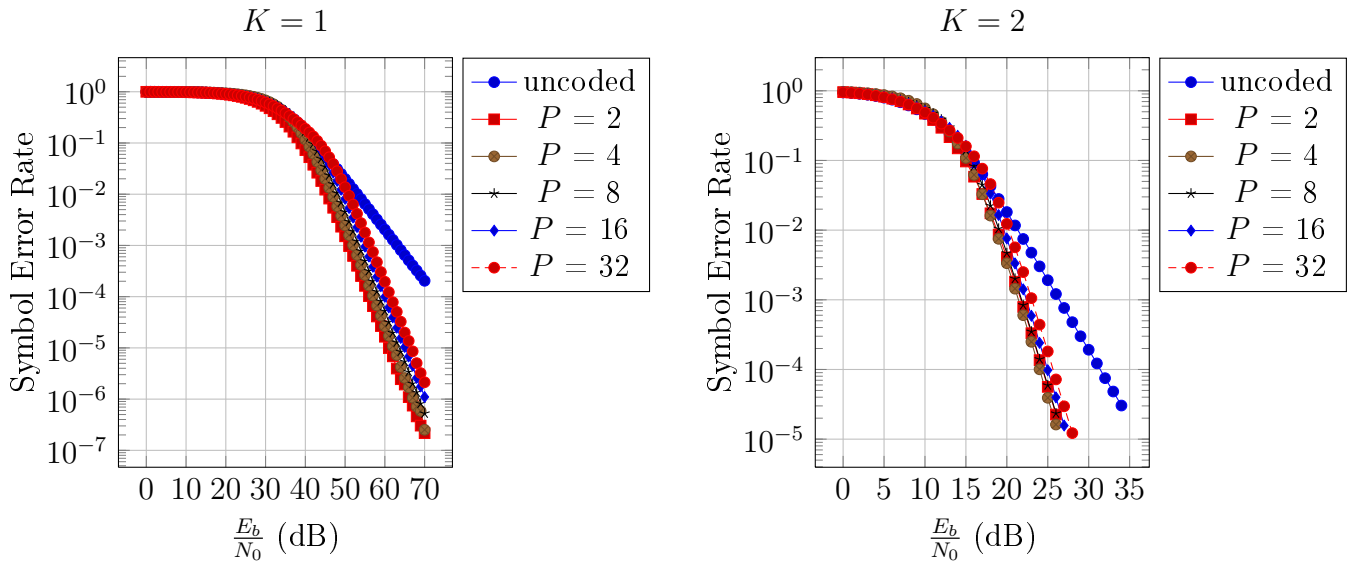


Figure 3.4: SER vs $\frac{E_b}{N_0}$ for an MBM system using 65536 realizations of an SRF channel, $K = 1$ or $K = 2$ receiving antennas and an SPC symbol code with block size $P$.

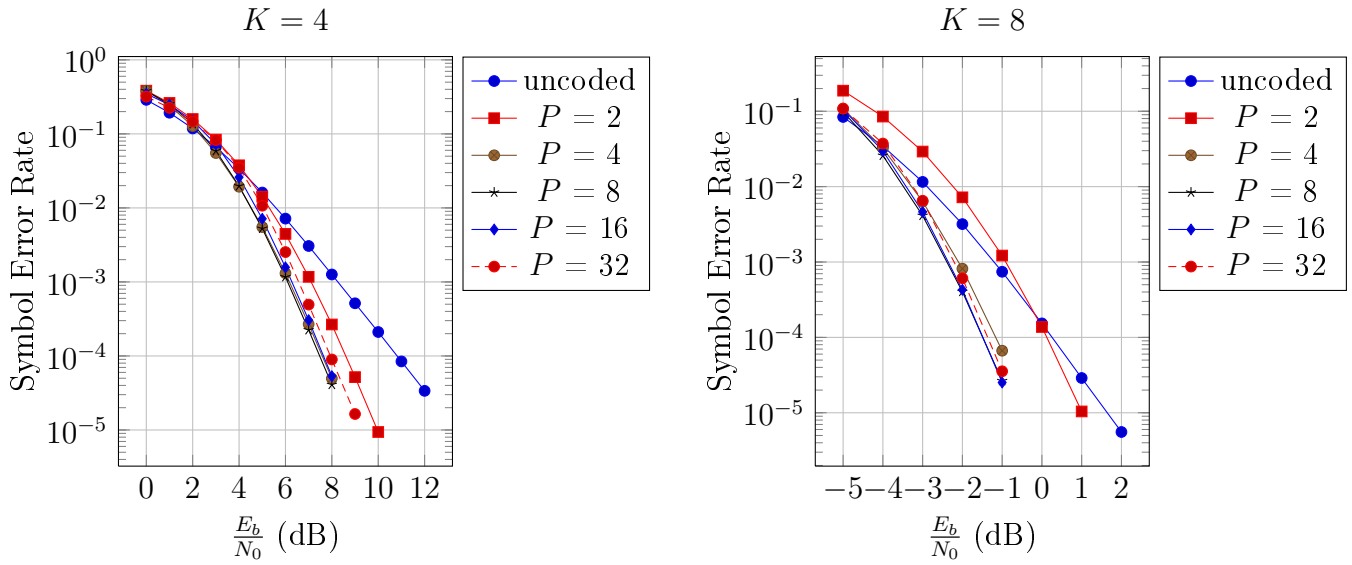


Figure 3.5: SER vs $\frac{E_b}{N_0}$ for an MBM system using 65536 realizations of an SRF channel, $K = 4$ or $K = 8$ receiving antennas and an SPC symbol code with block size $P$.

Coding gains are a function of the uncoded symbol error rate and the number of antennas. In MBM, the less antennas that are used, the greater the potential energy gains due to coding, regardless of the uncoded symbol error rate. As shown in Figure 3.1, the expected coding gain when an SBM system communicates over an AWGN channel is about $2-3$ dB regardless of the number of antennas. With MBM and $< 4$ antennas the coding gains are noticeably greater than 3 dB. In the regime when the number of antennas is $\geq 8$, the coding gains are similar to what would be expected with an SBM system communicating over an AWGN channel operating in the same regime of low uncoded symbol error rate. This is true for a 256 symbol MBM system and a 65536 symbol MBM system. The primary reasons for the difference in coding gains between MBM and SBM for this code are the fact that for MBM this code produces a diversity gain of at least 2 and that the incorporation of fading statistics into the constellation structure causes coding gains to depend on the number of antennas.

# Chapter 4

# ML Symbol Detection

This chapter presents an algorithm design that solves the Maximum Likelihood symbol detection problem for MBM constellations. Simulation results characterizing the algorithm performance are presented, as well as modifications that potentially improve performance at a cost of introducing a probability of incorrect decisions.

## 4.1 Background

Suppose a transmitter sends a constellation vector $\mathbf{x}$ with $K = 2Q$ real dimensions from a constellation set $\mathcal{X}$ of size $2^N$ over an AWGN channel to the receiver, which observers the vector $\mathbf{y}$. The ML decision problem is to solve:

$$\arg\max_i P\left(\mathbf{x}_i = \mathbf{x} \,|\, \mathbf{y}\right). \tag{4.1}$$

Analysis of this equation for an AWGN channel leads to the minimum distance rule, which states that the constellation vector that should be selected is the solution to the equation:

$$\arg\min_i \left(L_2(\mathbf{y} - \mathbf{x}_i)^2 = \|\mathbf{y} - \mathbf{x}_i\|^2\right). \tag{4.2}$$

It is assumed a priori that all vectors are equally likely to be transmitted and that there is no memory involved in the communication due to the channel, coding or the source data.

This minimization problem is equivalent to the Nearest Neighbor Search (NNS) problem in computer science, which is of fundamental importance in areas such as computational geometry.

When considering algorithm design, it is necessary to specify any computational constraints or requirements relevant to solving the problem. Consider the following block fading scenario:

- The transmitter sends training symbols for channel estimation and at the end of the training period the receiver has perfect knowledge of the channel.
- For a fixed and known block size, the transmitter sends a block of constellation vectors to the receiver and the receiver then solves the algorithmic tasks involved with ML decision making.

Any effort spent on preprocessing based on the channel matrix would be amortized over the block size, and any preprocessing based on the constellation would be performed once on initialization and amortized over many blocks. The objective of the algorithm design problem is to solve the ML symbol detection problem in a manner that minimizes the query complexity per symbol without requiring significant amounts of preprocessing.

The brute force solution to this problem is to calculate the Euclidean distance between the observed vector and all the constellation vectors in order to find the vector that minimizes the Euclidean distance, thereby spending $\mathcal{O}(Q2^N)$ effort per symbol with no preprocessing overhead. This approach is usually impractical to implement if the constellation size or the number of dimensions is large. In SBM, the constellation set $\mathcal{X}$ can be designed a priori to exhibit algebraic structure and it is common to assume that $\mathcal{X}$ is a lattice. A lattice is an algebraic structure in which all the elements of the lattice can be expressed as a linear combination of basis vectors that are in $\mathcal{R}^Q$ where the coefficients used in the linear combination are all integers. All lattices can be fully described by a generator matrix $\mathbf{G}$, which for this purpose is assumed to be known by the receiver. If the constellation is a lattice, then the ML symbol detection problem would be equivalent to the lattice decoding problem, which is of relevance to a variety of fields other than communication.

One main issue with lattice decoding is the fact that in general the problem is considered NP-Hard[14]. The worst-case difficulty of the problem varies depending on the particular SNR value of interest[15]. The average difficulty of the problem for various constellation sizes, fixed SNR ranges and a small number of antennas is polynomial however[16], implying that for many practical scenarios efficient lattice decoding algorithms will be much more computationally efficient than the worst-case analysis suggests.

At a high level, most approaches to solve the lattice decoding problem involves three main steps[17][18][19]:

- Left preprocessing, which involves performing a matrix decomposition matrix on $\mathbf{H}$ in order to simplify later calculation. Matrix decomposition is a computationally expensive procedure in general and usually involves $\mathcal{O}(Q^2)$ or $\mathcal{O}(Q^3)$ depending on the particular algorithm. This is considered negligible because it depends only on dimensionality $Q$ (which in practice is $< 16$), is independent of the constellation size, and is amortized over the block size.

- Right preprocessing, which involves transforming the generator matrix $\mathbf{G}$ into a lower dimensional matrix in a manner that preserves the optimality of the ML symbol detection problem in the lower dimensional space. This is done in practice by using algorithms such as LLL[20][21], which is polynomial time and would only need to be performed once initially as the constellation is known and fixed for all blocks.

- The actual solving of the lattice decoding problem can be viewed as searching over a tree for a leaf[17]. The branching nature of the tree is an consequence of working with lattices, and the fact that the tree has depth is due to the recursive nature of norm calculation. The majority of algorithms employ either Breadth First Search (BFS) or Depth First Search (DFS) to traverse the tree, and various heuristics are employed by exploiting algebra or number theory to dynamically reduce the size of the search space.

Alternative algorithms also exist that either solve the lattice decoding problem exactly[22] or produce approximate solutions[23].

The main challenge introduced by MBM is that the constellation is not a lattice. Even if the MBM system consists of an extension of an SBM system in which the underlying SBM constellation that is a lattice, the resulting constellation will at best be a union of different lattices, each with a different basis due to the different channel realizations. Lattice decoding algorithms with similar approaches to GSM[24][25] could be applied to this scenario, but this strategy in general is not applicable for all MBM constellations. Furthermore, in MBM it is advantageous to work with many receiving antennas ($> 8$), a constellation size that is large ($> 256$) and a low $\frac{E_b}{N_0}$ which is the among the most computationally difficult regimes to handle.

The randomness inherent in MBM constellations is such that lattice decoding algorithms that rely on algebraic structure are not feasible. Right preprocessing, or any technique that relies on information about the constellation, must be recalculated once per block and left preprocessing must be calculated once per channel realization. This causes the computational effort due to preprocessing to be fundamentally higher in an MBM context than what would be expected in SBM, and is especially problematic in the scenario

where the dimensionality and constellation size are large, which is of primary interest in MBM. There are at least three approaches to solve the ML symbol detection problem in this case:

- Design a tree-like data structure to represent the constellation[26][27]. Solving the ML symbol detection problem would then be equivalent to traversing the data structure and would therefore result in a small query complexity. Generally, this approach requires significant preprocessing effort to design the data structure and even if the effort is amortized over the block size it may remain too substantial to implement in a real-time communications setting.
- Use clustering (like [28]) or quantization based methods (like [29]) to partition the constellation set into groups of different vectors and search only over a particular group or set of groups. This usually only approximately solves the problem, but can be done with sufficiently low preprocessing computation to be implementable in practice.
- Apply search to identify possible candidate solutions and ultimately search less than the constellation size to find the optimal solution. Compared to brute force search, this approach reduces query complexity on average by reducing the number of points searched at a cost of increased preprocessing overhead. The query complexity may vary depending on the observed vector as well.

An algorithm design to solve the ML symbol decoding problem for MBM constellations based on searching a list of feasible candidates is presented in Chapter 4.2. The logic of the algorithm is inspired primarily from the search based algorithms used for lattice decoding. The algorithmic efficiency increases when the constellation size increases, but drops when the number of receiving antennas increases. The "curse of dimensionality" that results from high dimensionality is a fundamental issue in the NNS problem[30] and there is no known algorithm design that does not suffer from this to some extent regardless of the solution tactic used if the optimality of the final solution must be guaranteed. Modifications that can improve the algorithm speed at a cost of allowing erroneous solutions are presented in Chapter 4.3.

## 4.2 Exact ML Symbol Detection Algorithm

Suppose the transmitter sends a constellation vector $\mathbf{x}_i$, where $1 \leq i \leq 2^L$ to a receiver with $Q$ antennas. The receiver is assumed to know the coordinates of the MBM constellation

set. The goal of the ML symbol detector is to find:

$$\arg\min_i \left(L_2(\mathbf{y}-\mathbf{x}_i)^2 = \|\mathbf{y}-\mathbf{x}_i\|^2\right), \tag{4.3}$$

where $\mathbf{y} = \mathbf{x}_i + \mathbf{n}$ is the received vector and $\mathbf{n} \in \mathbb{R}^K$ is an additive white noise vector with i. i. d. Gaussian entries, $\mathcal{N}(0, \sigma_n^2)$. The $j$th component of vector $\mathbf{y}$ is denoted $y^{(j)}$.

In order to solve Equation 4.3, the brute force method would be to evaluate all the constellation vectors and select the one that minimizes $D_l$. The two main factors impacting query complexity are:

1. The number of candidates that are searched to find the optimal solution.
2. The amount of computation done to evaluate a candidate.

The first factor can be reduced by taking a sphere decoding approach, which involves defining a radius $R$ and solving:

$$\arg\min_i \left(L_2(\mathbf{y}-\mathbf{x}_i)^2 \leq R^2\right). \tag{4.4}$$

If $R^2$ is sufficiently large, then the solution of Equation 4.4 will be the same as the solution of Equation 4.3. By solving for the optimal solution within the sphere instead of simply the optimal solution, it is possible to use the following equation as a termination condition for the search process:

$$\begin{aligned} L_2(\mathbf{y}-\mathbf{x}_i)^2 &\leq R^2 \\ \sum_{j=1}^{K}(y^{(j)}-x_i^{(j)})^2 &\leq R^2 \\ (y^{(j)}-x_i^{(j)})^2 \leq R^2 &\text{ for each component } j \\ -R \leq y^{(j)}-x_i^{(j)} &\leq R \\ y^{(j)}-R \leq x_i^{(j)} &\leq y^{(j)}+R. \end{aligned} \tag{4.5}$$

The second factor can be reduced by noting that all $L_p^p$ norms including $L_\infty$ can be evaluated recursively when working with vectors. In particular, if a candidate $\mathbf{x}_i$ is within

the search radius $R$, then the following equation must be true:

$$\begin{aligned} L_2(\mathbf{y} - \mathbf{x}_i)^2 &\leq R^2 \\ \sum_{j=1}^{K} (y^{(j)} - x_i^{(j)})^2 &\leq R^2 \\ \sum_{j=1}^{p} (y^{(j)} - x_i^{(j)})^2 &\leq R^2 \qquad \forall\, 1 \leq p \leq K. \end{aligned} \tag{4.6}$$

In other words, a constellation vector can be rejected from further evaluation if any of the partial sums are outside the sphere. This property is related to the Depth First Search (DFS) algorithm for trees in the context of lattice decoding[17] and reduces the amount of work required to evaluate candidates when the number of dimensions is large.

Intuitively, since the goal is to find the constellation vector that is closest to the received vector, a third method to reduce the query complexity would be to search constellation vectors that appear to be in the neighborhood of the received vector with respect to one or more dimensions and use Equation 4.5 as the termination condition. Since the constellation vectors are randomly distributed, this would be done by creating sorted lists for each of the dimensions of the constellation and using a binary search to identify the candidates that are in the neighborhood of the received vector. The work required to create sorted lists will amortize over the block size, thereby becoming negligible if the block size is sufficiently large.

It is simpler to search a neighborhood in one list than multiple lists. Since each list corresponds to a particular dimension, it is necessary to determine which dimension should be selected when given the received vector. Suppose the following assumptions are true about the constellation vectors:

1. All the set of all the entries of all the constellation vectors represent a realization of an i. i. d. set of random variables with $\mathbb{E}[X] = 0$.

2. If the random variable used to generate the constellation vector entries is $X$, then $P(a - R \leq X \leq a + R) < P(b - R \leq x \leq b + R)$ for all $|a| < |b|$ and any fixed positive constant $R$ (i.e. the further away from the origin a component is, the less likely the component is to occur). This property is satisfied when $X$ is Gaussian.

Using only the first assumption, there is no advantage to selecting one dimension over another a priori. The second assumption implies that when considering only one dimension,

the number of candidates that need to be searched on average is minimized when the search radius is centered at a value with maximum absolute value. Therefore the dimensions should be evaluated in the order of decreasing absolute magnitude of the received vector and the list corresponding to the absolute largest component of the received vector should be used to specify the search candidates. Since the choice of search dimension is not known a priori, $K$ different sorted lists would need to be generated at the start of the block so that the appropriate list can be used when an information symbol is received. This would result in $\mathcal{O}(KN2^N)$ effort at the start of a block for sorting.

The proposed exact ML symbol detection algorithm is described below. It is assumed for simplicity that the received vector is in the convex hull of the constellation vectors. It is straightforward to extend the presented algorithm to handle the scenario in which the received vector is not on the convex hull of the constellation vectors.

1. Process the constellation by constructing an array of labels for each of the $K$ dimensions that correspond to the set constellation vectors if they had been sorted by increasing value along each dimension. This is done once before any observation vectors are received.

2. Select one of the arrays as the list of candidates by sorting the received vector components in order of decreasing absolute value. The list of candidates corresponds the the largest absolute value component of the received vector. The order of the sorted components defines the order in which the components of subsequent vector operations will be evaluated.

3. Define the initial search location by performing a binary search on the list of candidates. The goal of the search is to find the index in the array such that the appropriate component of the received vector could be inserted into that index position while maintaining the sorted order of the entries. The one dimensional distance between the received vector and the first constellation vector is the left search limit, and the difference between the received vector and the last constellation vector is the right search limit. The maximum of these two limits defines the search radius $d_{min}$.

4. Begin searching candidates until all the candidates within the search radius have been evaluated. Each candidate is evaluated by the following process:

   (a) Lift the candidate into the original $K$ dimensional space and calculate the Euclidean distance between the candidate and the received vector according to Equation 4.6. If the candidate is within the $K$-dimensional sphere specified by

radius $d_{min}$, then update $d_{min}$ to the new radius and reduce the left and right search limits accordingly. Note that since the candidate vectors have i. i. d. components, this lifting operation from a one dimensional component to its $K$ dimensional counterpart is a one-to-one operation with high probability. This is a departure from lattice decoding, in which the regularity of the lattice results in a tree-like search process.

(b) If $d_{min}$ was updated, then record the fact that evaluated candidate must be the current best candidate.

5. The search process terminates when there are no candidates within the search radius that have not already been evaluated. The current best candidate becomes the optimal candidate, as it is the only candidate within the $K$ dimensional sphere defined with radius $d_{min}$. Return the best candidate to terminate the algorithm.

For reference, code for a sample implementation of this algorithm written in the Julia programming language is shown in Appendix A.

The following remarks can be made about the proposed algorithm:

1. The average computational complexity is a function of $\frac{E_b}{N_0}$ due to its effect on the search radius size.

2. One advantage of evaluating dimensions in order of decreasing received vector absolute magnitude is that it usually produces a good quality ordering of partial sums for the depth first search. Different orderings of dimensions can lead to a noticeable speed reduction.

To characterize the efficiency of the algorithm, histograms involving an MBM constellation constructed using 256 realizations of an SRF channel, $Q$ receiving antennas and transmission at $\frac{E_b}{N_0} = 0$ dB are shown in Figure 4.1 and Figure 4.2. Figure 4.1 shows the relative proportion of the constellation that is searched before the algorithm terminates with an optimal solution. Figure 4.2 shows the number of add/multiply/comparisons operations, excluding the sorting done during preprocessing that the algorithm performs relative to what would be done when using brute force ML symbol detection. Brute force ML symbol detection involves about $(3K + 1)2^N$ operations.

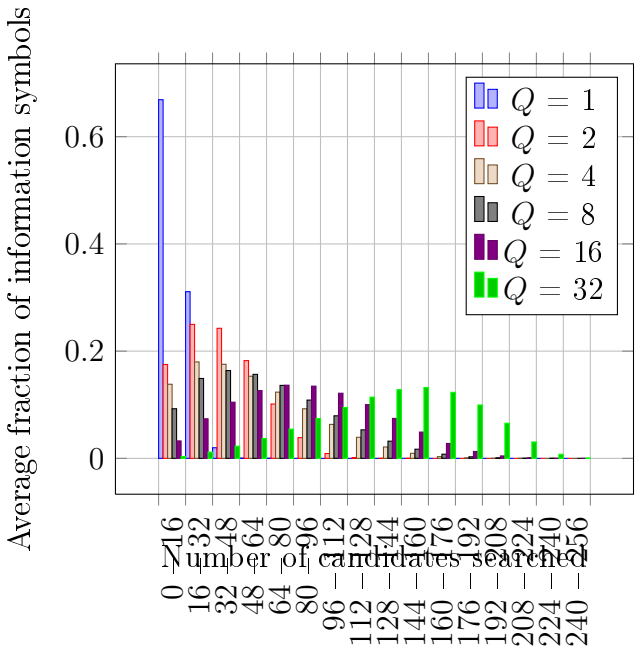


Figure 4.1: Histogram of the number of candidates searched by the exact ML symbol detection algorithm when an MBM constellation constructed using 256 realizations of an SRF channel is used with $Q$ receiving antennas and transmission at $\frac{E_b}{N_0} = 0$ dB.

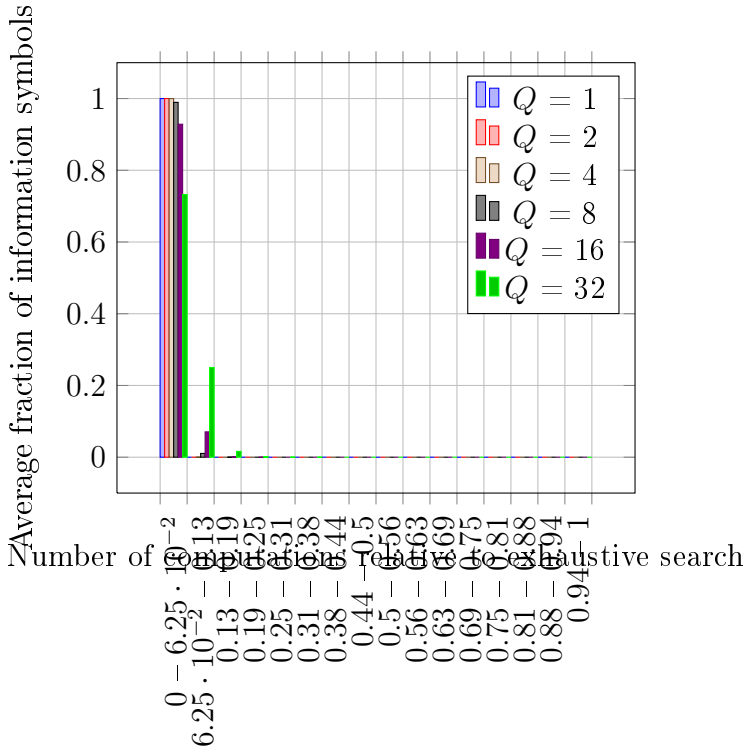


Figure 4.2: Histogram of the computational effort spent by the exact ML symbol detection algorithm relative to brute force search when an MBM constellation constructed using 256 realizations of an SRF channel is used with $Q$ receiving antennas and transmission at $\frac{E_b}{N_0} = 0$ dB.

Adding receiving antennas causes more candidates to be searched on average since Euclidean distance is additive. The relative amount of computational effort decreases when the number of receiving antennas increases since the computational savings obtained from recursively evaluating the norm becomes more apparent. Together these factors result in a net increase in computational efficiency as the number of dimensions approaches infinity.

A histogram involving an MBM constellation constructed using 65536 realizations of an SRF channel, $Q$ receiving antennas and transmission at $\frac{E_b}{N_0} = 0$ dB are shown in Figure 4.3.

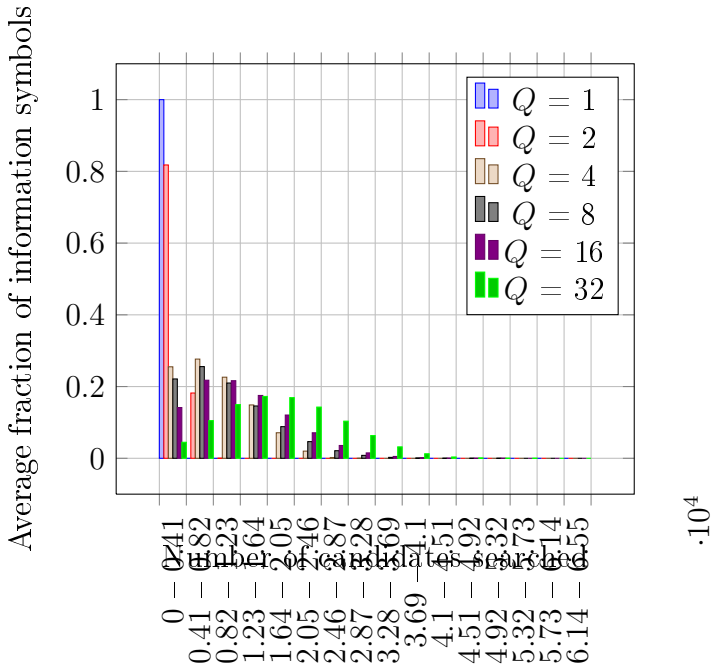


Figure 4.3: Histogram of the number of candidates searched by the exact ML symbol detection algorithm when an MBM constellation constructed using 65536 realizations of an SRF channel is used with $Q$ receiving antennas and transmission at $\frac{E_b}{N_0} = 0$ dB.

The efficiency of the algorithm relative to brute force search improves when the constellation size increases, as shown when Figure 4.1 and Figure 4.3 are compared. The difference in efficiency increases as the number of receiving antennas increases.

Suppose the receiver had access to an oracle which could verify whether a candidate solution is ML optimal. The algorithm produces a list of candidates to search and the search terminates once the oracle concludes the ML solution has been found. The total number of candidates searched with the oracle never exceeds the total number of candidates searched

without the oracle. A histogram showing the average minimum number of candidates that must be searched by the exact ML symbol detection algorithm using the previous MBM system as Figure 4.1 is shown in Figure 4.4.

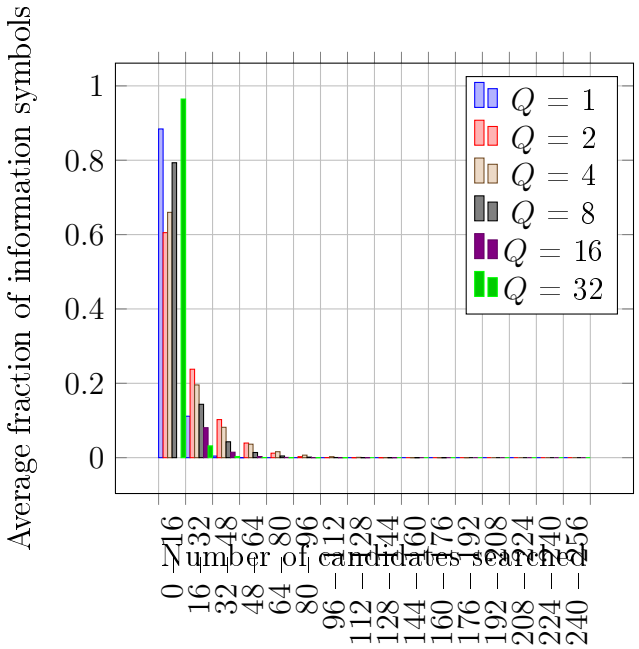


Figure 4.4: Histogram of the minimum number of candidates searched with the presence of an oracle by the exact ML symbol detection algorithm when an MBM constellation constructed using 256 realizations of an SRF channel is used with $Q$ receiving antennas and transmission at $\frac{E_b}{N_0} = 0$ dB.

Comparing Figure 4.1 and Figure 4.4, the exact ML symbol detection algorithm is searching more candidates than necessary when more dimensions are added in order to verify that a candidate solution is correct. Since the list of candidates is being selected based on the maximum absolute magnitude of the received vector, it is increasingly likely for the optimal candidate to be found earlier in the search process as the number of dimensions increases and is sufficiently large. This is due to fact that the number of points searched in the neighborhood of a value reduces as the absolute magnitude of the value increases and the sorting process causes this value to increase as the dimensionality increases. While there is little that can be done while retaining optimality, this fact justifies the use of several different strategies for approximate ML symbol detection.

## 4.3 Approximate ML Symbol Detection Algorithm

The ML symbol detection involves minimizing $L_2^2(\mathbf{y}-\mathbf{x}_i)$ for all $i$ where $L_2$ is the Euclidean norm. The number of candidates searched is based on the size of the optimal search radius, which if a correct decision is made has an expected size of:

$$\mathbb{E}[L_2(\mathbf{y}-\mathbf{x})] = \mathbb{E}[L_2(\mathbf{n})] = \sqrt{2}\sigma_n \frac{\Gamma(Q+\frac{1}{2})}{\Gamma(Q)} \propto \sqrt{Q}, \tag{4.7}$$

where $Q$ is the number of receiving antennas. This is a major source of inefficiency when $Q$ is large.

Consider an algorithm that minimizes:

$$L_\infty(\mathbf{y}-\mathbf{x}_i) = \max_{1\leq j\leq K} |y^{(j)} - x_i^{(j)}|,$$

for all $i$, where $L_\infty$ is the infinity norm. As $\frac{E_b}{N_0} \to \infty$, the error rate will approach zero regardless of whether the Euclidean or infinity norm is used, implying that there will not be any steady-state error rate due to the use of the infinity norm. Note that if a correct decision is made, it can be shown that:

$$\lim_{Q\to\infty} \mathbb{E}[L_\infty(\mathbf{y}-\mathbf{x}_i)] < \sqrt{2\sigma_n^2 \ln(Q)}). \tag{4.8}$$

Since both features of the exact algorithm that minimize the $L_2$ norm are also applicable for the $L_\infty$, the logic of the exact ML symbol detection algorithm can be easily adapted to find the candidate that minimizes $L_\infty(\mathbf{y}-\mathbf{x}_i)$ with no significant change in implementation complexity. If the $L_\infty$ norm is used to define the search radius and the $L_2$ norm is used to select the optimal candidate, then the number of candidates searched by the algorithm will be significantly smaller when $Q$ is large compared to the exact algorithm. This will also introduce approximation error since the new $L_\infty$ search radius may be too small, resulting in suboptimal candidate selection.

More specifically, when a candidate is visited, the $L_2$ norm is used to determine if it is a better candidate then the current best solution. If so, the corresponding $L_\infty$ norm for this new optimal candidate is calculated and compared against the current search radius, and the search radius is adjusted accordingly. Since the radius update only occurs when a new current best candidate is found, the overhead of calculating the $L_\infty$ norm is reduced compared to calculating the $L_\infty$ norm for every candidate that is searched.

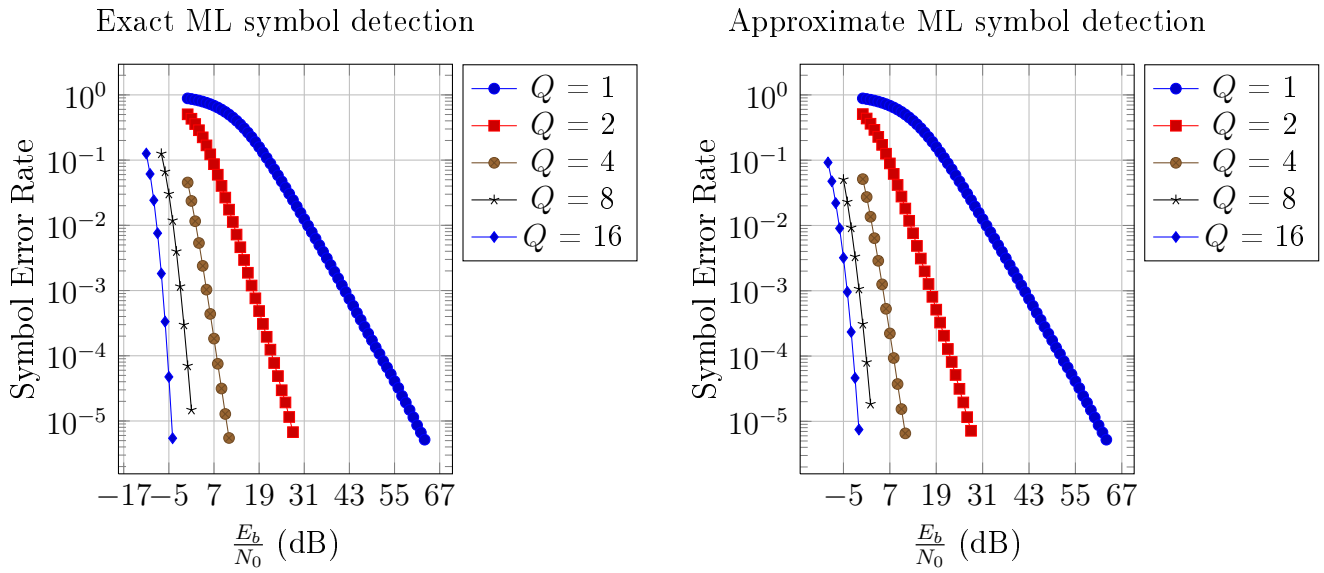


Figure 4.5: SER vs $\frac{E_b}{N_0}$ for an MBM system using 256 independent realizations of an SRF channel and $Q$ receiving antennas using the exact or approximate ML symbol detection algorithm.

Consider an MBM system with $Q$ receiving antennas that uses a constellation that is constructed using 256 realizations of an SRF channel. The SER for this MBM system using the exact ML symbol detection algorithm and the approximate ML symbol detection algorithm is shown in Figure 4.5. Histograms for a fixed $\frac{E_b}{N_0} = 0$ dB showing the number of computations relative to brute force search excluding preprocessing performed by the algorithm and the proportion of the constellation that is searched are shown in Figure 4.6 and Figure 4.7 respectively. Recall that brute force ML symbol detection involves about $(3(K) + 1)2^N$ operations.

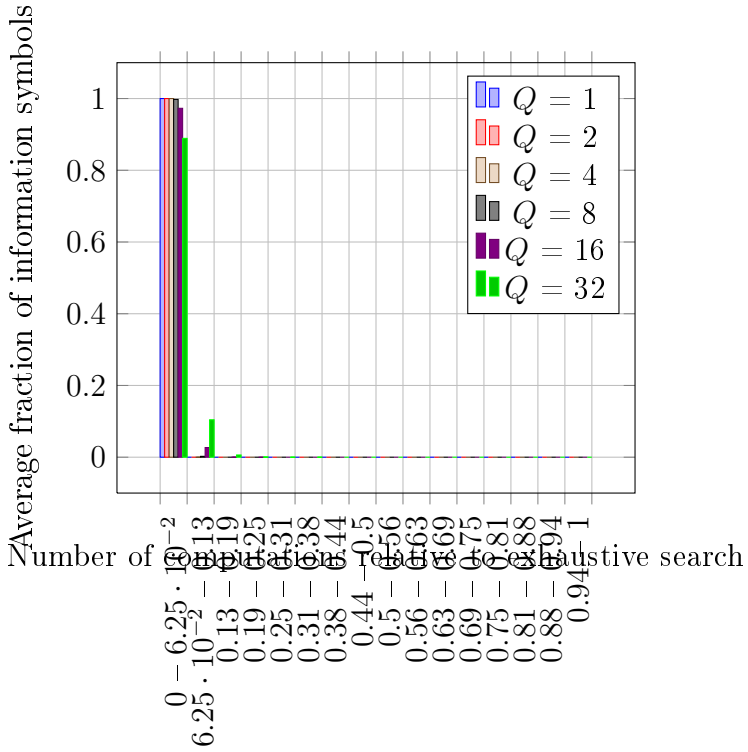


Figure 4.6: Histogram of the computational effort spent by the approximate ML symbol detection algorithm relative to brute force search when an MBM constellation constructed using 256 realizations of an SRF channel is used with $Q$ receiving antennas and transmission at $\frac{E_b}{N_0} = 0$ dB.

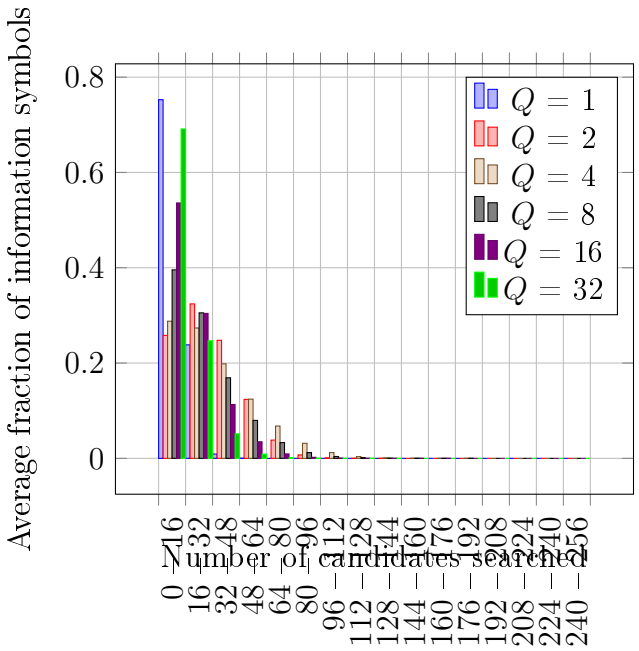


Figure 4.7: Histogram of the number of candidates searched by the approximate ML symbol detection algorithm when an MBM constellation constructed using 256 realizations of an SRF channel is used with $Q$ receiving antennas and transmission at $\frac{E_b}{N_0} = 0$ dB.

When comparing Figure 4.1 and Figure 4.7, the approximate algorithm searches noticeably less candidates when the number of receiving antennas is sufficiently high. The difference in the number of candidates searched only increases as the number of receiving antennas increases. Unfortunately, this reduction in computational effort is partially offset by the complexity of updating the radius selection increasing due to the $L_\infty$ norm calculation, which increases as the number of receiving antennas increases. As a result, the main the computational savings introduced in this exact algorithm are only obvious when the number of receiving antennas is sufficiently high. For a fixed $\frac{E_b}{N_0}$ and in the limit as both the constellation size and number of receiving antennas approach infinity, the approximate ML symbol detection algorithm will search an increasingly small percentage of the constellation and relative to the brute force search will save an increasingly greater amount of computational effort. Note that although the percentage of the constellation searched decreases, the quantity of candidates searched increases. As $\frac{E_b}{N_0}$ increases, the number of candidates that must be searched will decrease, resulting in substantial computational savings.

Based on Figure 4.5, the amount of additional $\frac{E_b}{N_0}$ required for the approximate ML symbol detection algorithm to achieve the same symbol error rate as the optimal ML symbol detection algorithm is approximately constant in dB for the evaluated MBM constellation, assuming $\frac{E_b}{N_0}$ is sufficiently high and $\geq 4$ receiving antennas are used. Unfortunately, as the number of receiving antennas increases, so does this energy loss. Note that since the list of candidates being searched and the ordering of candidates within the list is the same for both the approximate and optimal algorithms, they differ on only how many candidates along the list are evaluated. The constant factor energy loss is therefore because the ML solution is sometimes outside the reduced search radius.

To mitigate this issue, one solution is to multiply the $L_\infty$ search radius by a multiplicative factor $\beta > 1$. For example, if $\beta = 1.2$, then the energy loss when using 32 receiving antennas is about $< 1.5$ dB with the MBM system that involves 256 realizations of an SRF channel and the overall computational time due to the increased number of search candidates is also about 20% higher. By using the $\beta$ factor, it is possible to directly trade a reduction in energy loss for an increase in computational effort. Depending on the constellation size, number of receiving antennas and amount of energy loss that is acceptable, the use of a small $\beta$ value can allow for an arbitrarily accurate approximation to the ML symbol detection algorithm with less computational effort.

To understand the computational complexity of the optimal and approximate ML symbol detection algorithms, it is helpful to consider the size of the search radius as a function of the number of receiving antennas, $Q$. The search process consists of searching candidates along a one dimensional list in a neighborhood around a received component. Since the

noise is Gaussian and the received component is selected from maximizing the absolute value of a vector, the average value of this received component will grow at least as fast as $\mathcal{O}(\sqrt{Q})$. The average size of the search radius for the exact algorithm is also at least as fast as approximately $\mathcal{O}(\sqrt{Q})$. The average size of the search radius for the approximate algorithm is approximately $\mathcal{O}(\sqrt{\log(Q)})$. In light of the energy loss that results from using the approximate algorithm, it can be argued that an $\mathcal{O}(\sqrt{\log(Q)})$ search radius is most likely too small to accurately solve the ML symbol detection problem when $Q$ is large. In contrast, a search radius that is $\mathcal{O}(\sqrt{Q})$ is most likely too large, indicating that an optimal search radius would most likely lie within these two bounds.

For reference, code for a sample implementation of the approximate ML symbol detection algorithm is shown in Appendix A.

# Chapter 5

# Conclusion and Future Work

## 5.1 Conclusions

An SIMO MBM system communicating over an SRF is equivalent to SIMO SBM system communicating over an AWGN channel with the appropriate constellation. With that in mind, this thesis compared comumunication with SIMO MBM systems over an SRF channel to SIMO SBM systems over an AWGN channel and presented the following results:

- Generally, MBM constellations constructed from an SRF channel outperform SBM constellations communicating over an AWGN channel if enough receiving antennas are used. The energy gains introduced by MBM are substantial relative to comparable SBM systems. MBM also faciilitates the use of significantly higher constellation sizes than what is used in SBM without significant energy loss. The geometric properties of MBM over an AWGN channel are fundamentally different than that of SBM constellations and they result in different performance scaling properties both in theory and in simulation.

- Existing coding schemes which are designed and analyzed for SBM constellations perform differently with MBM due to differences in constellation geometry. The coding gains expected with MBM systems can be significantly greater than the coding gains expected with SBM systems mainly because the constellation geometry can cause additional diversity gains due to coding.

- The random nature of MBM constellations prevents the use of many existing ML symbol detection algorithms. Using ideas from sphere decoding, it is possible to

implement an exact ML symbol detection algorithm that usually results in at least a factor of 5 reduction in query complexity for MBM constellations if $\frac{E_b}{N_0} > 0$ dB, the constellation size is $> 256$ and the number of receiving antennas is $< 8$. There is algorithmic inefficiency when the number of receiving antennas is high. To combat this issue, an approximate ML symbol detection was presented that could improve search efficiency at a cost of a constant factor energy loss.

- When working with $< 4$ antennas, the energy gains resulting from coding or from adding antennas is substantial. Moreover, the proposed optimal ML symbol detection algorithm is near optimal in this regime due to the low dimensionality. When working with $> 8$ antennas, the energy gain in MBM due to adding antennas is approximately equal to what would be expected in SBM. The proposed optimal ML symbol detection is also inefficient in high dimensions, but due to the nature of the geometry of the problem there are opportunities to resolve this issue if optimality is sacrificed.

If sufficient receiving antennas are used, MBM has the potential to significantly outperform SBM both in communication speed and reliability.

## 5.2 Future Work

There are several major issues that need to be investigated in future work:

- The channel estimation problem in MBM is complicated by the fact that only one realization can be observed at a given time during the training process. This fundamentally degrades the effectiveness of the training process and will result in higher estimation error. The complete characterization of this fact and the design of estimation algorithms that can mitigate this issue is necessary for the use of MBM in practical situations.

- In this thesis, MBM was always used in conjunction with an SRF channel. If the channel is assumed to exhibit slow or fast fading, it is not clear how this will impact the average performance. Taking into account estimation error and the re-sending of training symbols, there may ultimately be significant reductions in rate when using MBM relative to SBM.

- Given the differences between SIMO SBM and SIMO MBM, it is clear that adding antennas in MBM is fundamentally different than in SBM. Extending the results

to the MIMO scenario, and in particular determining how space-time coding should be applied to MBM is another future direction. Given the performance gains that MIMO SBM systems have relative to SISO SBM systems, there is a high probability that MIMO MBM will also have opportunities for substantial performance gains.

- The impact of channel coding on MBM was investigated only in a preliminary manner. While the results are most likely applicable to block codes, it is not obvious if they will extend to more sophisticated coding schemes. Even without leveraging the properties of MIMO systems, the issue of how best to design a code knowing the constellation is Gaussian has not been thoroughly investigated.

- At the level of abstraction discussed in this thesis, bandwidth and communication rate is only indirectly addressed. In practice, changing realizations at the transmitter will result in switching overhead that will cause bandwidth expansion. The investigation of how significant these effects are, how they can be mitigated if mitigation is necessary at all and how these second-order effects degrade performance is also an open question.

- There can be substantial challenges when working with large constellation sizes both in terms of RF complexity and channel estimation. More elegent approaches to realize MIMO-MBM with large constellations is an area for future research.

# Appendix

# Appendix A

# Supplementary Code

## A.1 Code for Exact ML Symbol Detection Algorithm

The following is sample code written in the Julia programming language to implement the exact ML symbol detection algorithm described in Chapter 4.

```
% idata = input observation vector
% csize = constellation size
% dimr = number of receiving dimensions
% ctablep = table containing the sorted lists of the constellation vectors generated during preprocessing
% dtmp = array used during temporary calculation
% osymbol = symbol index that will be returned as an output
% oval = corresponding minimum distance value that will be returned as an output

% isort = Insertion Sort function that returns the permutation of indices required to sort the input
%         from least to greatest

function exactmlsymdetect(idata::Array{Float64, 1}, iconst::Array{Float64, 2}, csize::Int64, dimr::Int64,
                          ctablep::Array{Int64, 2}, dtmp::Array{Float64, 1}, osymbol::Int64, oval::Float64)
    const q = dimr - 1

    % Calculate -(abs(idata))
    for m = 1:dimr
        if (idata[m] <= 0.0)
            dtmp[m] = idata[m]
        else
            dtmp[m] = -idata[m]
        end
    end

    % Sort the temporary array to determine the order in which dimensions should be evaluated
    yperm = isort(dtmp, dimr)

    % Use binary search to find the position in the appropriate list of candidates to start the search
    low = 1
```

```
high = csize
tmp = idata[yperm[1]]

if (iconst[ctablep[high, yperm[1]], yperm[1]] <= tmp)
    low = csize
else
    while (low < (high - 1))
        mid = div((low + high), 2)

        if (tmp <= iconst[ctablep[mid, yperm[1]], yperm[1]])
            high = mid
        else
            low = mid
        end
    end
end

if (low < csize)
    k = low
else
    k = csize - 1
end
t = k + 1

% Initialize the bounds
lbd = idata[yperm[1]] - iconst[ctablep[1, yperm[1]], yperm[1]]
rbd = iconst[ctablep[csize, yperm[1]], yperm[1]] - idata[yperm[1]]

dmin = Inf
dleft = 0.0
dright = 0.0

% Explicitly check the edge cases and search them first
if (lbd < 0)
    didx = ctablep[k, yperm[1]]
    dleft = iconst[didx, yperm[1]] - idata[yperm[1]]
    dmin = dleft^2
    for l = 2:dimr
        dmin += (idata[yperm[l]] - iconst[didx, yperm[l]])^2
    end
    osymbol = didx
    rd = sqrt(dmin)
    if (rd < rbd)
        rbd = rd
    end
    lbd = dleft
    if (rd < lbd)
        lbd = rd
    end
    k -= 1
else
    if (rbd < 0)
        didx = ctablep[t, yperm[1]]
        dright = (idata[yperm[1]] - iconst[didx, yperm[1]])
        dmin = dright^2
        for l = 2:dimr
            dmin += (idata[yperm[l]] - iconst[didx, yperm[l]])^2
```

```
        end
        osymbol = didx
        rd = sqrt(dmin)
        rbd = dright
        if (rd < rbd)
            rbd = rd
        end
        if (rd < lbd)
            lbd = rd
        end
        t += 1
    end
end

% Search candidates one at a time from the left and right directions
while ((dleft <= lbd) && (dright <= rbd) && (0 < k) && (t <= csize))
    didx = ctablep[k, yperm[1]]
    dleft = idata[yperm[1]] - iconst[didx, yperm[1]]
    dtable = dleft^2
    for l = 2:q
        dtable += (idata[yperm[l]] - iconst[didx, yperm[l]])^2
        if (dmin < dtable)
            dtable = 0.0
            break
        end
    end

    if (dtable > 0.0)
        dtable += (idata[yperm[dimr]] - iconst[didx, yperm[dimr]])^2
        if (dtable <= dmin)
            dmin = dtable
            osymbol = didx
            rd = sqrt(dmin)
            if (rd < rbd)
                rbd = rd
            end
            if (rd < lbd)
                lbd = rd
            end
        end
    end
    k -= 1

    didx = ctablep[t, yperm[1]]
    dright = (iconst[didx, yperm[1]] - idata[yperm[1]])
    dtable = dright^2
    for l = 2:q
        dtable += (idata[yperm[l]] - iconst[didx, yperm[l]])^2
        if (dmin < dtable)
            dtable = 0.0
            break
        end
    end

    if (dtable > 0.0)
        dtable += (idata[yperm[dimr]] - iconst[didx, yperm[dimr]])^2
        if (dtable <= dmin)
```

```
            dmin = dtable
            osymbol = didx
            rd = sqrt(dmin)
            if (rd < rbd)
                rbd = rd
            end
            if (rd < lbd)
                lbd = rd
            end
        end
    end
    t += 1
end

% The loop terminates when either the left or right bound has been reached.
% The remaining bound must be searched explicitly.

while ((dleft <= lbd)  && (0 < k))
    didx = ctablep[k, yperm[1]]
    dleft = idata[yperm[1]] - iconst[didx, yperm[1]]
    dtable = dleft^2
    for l = 2:q
        dtable += (idata[yperm[l]] - iconst[didx, yperm[l]])^2
        if (dmin < dtable)
            dtable = 0.0
            break
        end
    end

    if (dtable > 0.0)
        dtable += (idata[yperm[dimr]] - iconst[didx, yperm[dimr]])^2
        if (dtable <= dmin)
            dmin = dtable
            osymbol = didx
            rd = sqrt(dmin)
            if (rd < lbd)
                lbd = rd
            end
        end
    end
    k -= 1
end

while ((dright <= rbd) && (t <= csize))
    didx = ctablep[t, yperm[1]]
    dright = (iconst[didx, yperm[1]] - idata[yperm[1]])
    dtable = dright^2
    for l = 2:q
        dtable += (idata[yperm[l]] - iconst[didx, yperm[l]])^2
        if (dmin < dtable)
            dtable = 0.0
            break
        end
    end

    if (dtable > 0.0)
        dtable += (idata[yperm[dimr]] - iconst[didx, yperm[dimr]])^2
```

```
            if (dtable <= dmin)
                dmin = dtable
                osymbol = didx
                rd = sqrt(dmin)
                if (rd < rbd)
                    rbd = rd
                end
            end
        end
        t += 1
    end
    osymbol -= 1
    oval = dmin

    return (osymbol, oval)
end
```

## A.2 Code for Approximate ML Symbol Detection Algorithm

The following is sample code written in the Julia programming language to implement the approximate ML symbol detection algorithm described in Chapter 4.

```
% idata = input observation vector
% csize = constellation size
% dimr = number of receiving dimensions
% ctablep = table containing the sorted lists of the constellation vectors generated during preprocessing
% dtmp = array used during temporary calculation
% osymbol = symbol index that will be returned as an output
% oval = corresponding minimum distance value that will be returned as an output

% isort = Insertion Sort function that returns the permutation of indices required to sort the input
%         from least to greatest

function approxmlsymdetect(idata::Array{Float64, 1}, iconst::Array{Float64, 2}, csize::Int64, dimr::Int64,
                           ctablep::Array{Int64, 2}, dtmp::Array{Float64, 1}, osymbol::Int64, oval::Float64)
    const q = dimr - 1
    const beta = 1.2

    % Calculate -(abs(idata))
    for m = 1:dimr
        if (idata[m] <= 0.0)
            dtmp[m] = idata[m]
        else
            dtmp[m] = -idata[m]
        end
    end

    % Sort the temporary array to determine the order in which dimensions should be evaluated
    yperm = isort(dtmp, dimr)
```

```
% Use binary search to find the position in the appropriate list of candidates to start the search
low = 1
high = csize
tmp = idata[yperm[1]]

if (iconst[ctablep[high, yperm[1]], yperm[1]] <= tmp)
    low = csize
else
    while (low < (high - 1))
        mid = div((low + high), 2)

        if (tmp <= iconst[ctablep[mid, yperm[1]], yperm[1]])
            high = mid
        else
            low = mid
        end
    end
end

if (low < csize)
    k = low
else
    k = csize - 1
end
t = k + 1

% Initialize the bounds
lbd = idata[yperm[1]] - iconst[ctablep[1, yperm[1]], yperm[1]]
rbd = iconst[ctablep[csize, yperm[1]], yperm[1]] - idata[yperm[1]]

dmin = Inf
dmax = max(lbd, rbd)
dleft = 0.0
dright = 0.0

% Explicitly check the edge cases and search them first
if (lbd < 0)
    didx = ctablep[k, yperm[1]]
    dleft = iconst[didx, yperm[1]] - idata[yperm[1]]
    dmin = dleft^2
    for l = 2:dimr
        dmin += (idata[yperm[l]] - iconst[didx, yperm[l]])^2
    end
    osymbol = didx

    curmax = dleft
    lbd = dleft
    for l = 2:dimr
        dcur = abs(iconst[didx, yperm[l]] - idata[yperm[l]])
        if (curmax < dcur)
            curmax = dcur
        end
        if (curmax > dmax)
            break
        end
    end
    curmax *= beta
```

```
    if (curmax < rbd)
        dmax = curmax
        rbd = dmax
    end
    k -= 1
else
    if (rbd < 0)
        didx = ctablep[t, yperm[1]]
        dright = (idata[yperm[1]] - iconst[didx, yperm[1]])
        dmin = dright^2
        for l = 2:dimr
            dmin += (idata[yperm[l]] - iconst[didx, yperm[l]])^2
        end
        osymbol = didx
        curmax = dright
        rbd = dright
        for l = 2:dimr
            dcur = abs(idata[yperm[l]] - iconst[didx, yperm[l]])
            if (curmax < dcur)
                curmax = dcur
            end
            if (curmax > dmax)
                break
            end
        end
        curmax *= beta
        if (curmax < lbd)
            dmax = curmax
            lbd = dmax
        end
        t += 1
    end
end

% Search candidates one at a time from the left and right directions
while ((dleft <= lbd) && (dright <= rbd) && (0 < k) && (t <= csize))
    didx = ctablep[k, yperm[1]]
    dleft = idata[yperm[1]] - iconst[didx, yperm[1]]
    dtable = dleft^2
    for l = 2:q
        dtable += (idata[yperm[l]] - iconst[didx, yperm[l]])^2
        if (dmin < dtable)
            dtable = 0.0
            break
        end
    end

    if (dtable > 0.0)
        dtable += (idata[yperm[dimr]] - iconst[didx, yperm[dimr]])^2
        if (dtable <= dmin)
            dmin = dtable
            osymbol = didx
            curmax = dleft
            for l = 2:dimr
                dcur = abs(idatayperm[l]] - iconst[didx, yperm[l]])
                if (curmax < dcur)
                    curmax = dcur
```

```
            end
            if (curmax > dmax)
                break
            end
        end
        curmax *= beta
        if (curmax < dmax)
            dmax = curmax
            if (dmax < lbd)
                lbd = dmax
            end
            if (dmax < rbd)
                rbd = dmax
            end
        end
    end
end
k -= 1

didx = ctablep[t, yperm[1]]
dright = (iconst[didx, yperm[1]] - idata[yperm[1]])
dtable = dright^2
for l = 2:q
    dtable += (idata[yperm[l]] - iconst[didx, yperm[l]])^2
    if (dmin < dtable)
        dtable = 0.0
        break
    end
end

if (dtable > 0.0)
    dtable += (idata[yperm[dimr]] - iconst[didx, yperm[dimr]])^2
    if (dtable <= dmin)
        dmin = dtable
        osymbol = didx
        curmax = dright
        for l = 2:dimr
            dcur = abs(idata[yperm[l]] - iconst[didx, yperm[l]])
            if (curmax < dcur)
                curmax = dcur
            end
            if (curmax > dmax)
                break
            end
        end
        curmax *= beta
        if (curmax < dmax)
            dmax = curmax
            if (dmax < lbd)
                lbd = dmax
            end
            if (dmax < rbd)
                rbd = dmax
            end
        end
    end
end
```

```
    t += 1
end

% The loop terminates when either the left or right bound has been reached.
% The remaining bound must be searched explicitly.

while ((dleft <= lbd)  && (0 < k))
    didx = ctablep[k, yperm[1]]
    dleft = idata[yperm[1]] - iconst[didx, yperm[1]]
    dtable = dleft^2
    for l = 2:q
        dtable += (idata[yperm[l]] - iconst[didx, yperm[l]])^2
        if (dmin < dtable)
            dtable = 0.0
            break
        end
    end

    if (dtable > 0.0)
        dtable += (idata[yperm[dimr]] - iconst[didx, yperm[dimr]])^2
        if (dtable <= dmin)
            dmin = dtable
            osymbol = didx
            curmax = dleft
            for l = 2:dimr
                dcur = abs(idata[yperm[l]] - iconst[didx, yperm[l]])
                if (curmax < dcur)
                    curmax = dcur
                end
                if (curmax > dmax)
                    break
                end
            end
            curmax *= beta
            if (curmax < lbd)
                dmax = curmax
                lbd = dmax
            end
        end
    end
    k -= 1
end

while ((dright <= rbd) && (t <= csize))
    didx = ctablep[t, yperm[1]]
    dright = (iconst[didx, yperm[1]] - idata[yperm[1]])
    dtable = dright^2
    for l = 2:q
        dtable += (idata[yperm[l]] - iconst[didx, yperm[l]])^2
        if (dmin < dtable)
            dtable = 0.0
            break
        end
    end

    if (dtable > 0.0)
        dtable += (idata[yperm[dimr]] - iconst[didx, yperm[dimr]])^2
```

```
            if (dtable <= dmin)
                dmin = dtable
                osymbol = didx
                curmax = dright
                for l = 2:dimr
                    dcur = abs(idata[yperm[l]] - iconst[didx, yperm[l]])
                    if (curmax < dcur)
                        curmax = dcur
                    end
                    if (curmax > dmax)
                        break
                    end
                end
                curmax *= beta
                if (curmax < rbd)
                    dmax = curmax
                    rbd = dmax
                end
            end
        end
        t += 1
    end
    osymbol -= 1
    oval = dmin

    return (osymbol, oval)
end
```

# References


[1] A.K. Khandani. Media-based modulation: A new approach to wireless transmission. In *Information Theory Proceedings (ISIT), 2013 IEEE International Symposium on*, pages 3050–3054, July 2013.

[2] G.D. Forney, R.G. Gallager, G. Lang, F.M. Longstaff, and S.U. Qureshi. Efficient modulation for band-limited channels. *Selected Areas in Communications, IEEE Journal on*, 2(5):632–647, Sep 1984.

[3] J. Bellorado, S. Ghassemzadeh, and A. Kavcic. Approaching the capacity of the mimo rayleigh flat-fading channel with qam constellations, independent across antennas and dimensions. *Wireless Communications, IEEE Transactions on*, 5(6):1322–1332, June 2006.

[4] C.E. Shannon. A mathematical theory of communication. *Bell System Technical Journal, The*, 27(3):379–423, July 1948.

[5] A. Younis, N. Serafimovski, R. Mesleh, and H. Haas. Generalised spatial modulation. In *Signals, Systems and Computers (ASILOMAR), 2010 Conference Record of the Forty Fourth Asilomar Conference on*, pages 1498–1502, Nov 2010.

[6] A.K. Khandani. Media-based modulation: Converting static rayleigh fading to awgn. In *Information Theory (ISIT), 2014 IEEE International Symposium on*, pages 1549–1553, June 2014.

[7] Vahid Tarokh, N. Seshadri, and A.R. Calderbank. Space-time codes for high data rate wireless communication: performance criterion and code construction. *Information Theory, IEEE Transactions on*, 44(2):744–765, Mar 1998.

[8] T. Svantesson and A.L. Swindlehurst. A performance bound for prediction of mimo channels. *Signal Processing, IEEE Transactions on*, 54(2):520–529, Feb 2006.

[9] Ezio Biglieri, J. Proakis, and S. Shamai. Fading channels: information-theoretic and communications aspects. *Information Theory, IEEE Transactions on*, 44(6):2619–2692, Oct 1998.

[10] G. Ungerboeck. Channel coding with multilevel/phase signals. *Information Theory, IEEE Transactions on*, 28(1):55–67, Jan 1982.

[11] G. Yilmaz, E. Basar, and U. Aygolu. Trellis coded space-shift keying modulation. In *Vehicular Technology Conference (VTC Spring), 2014 IEEE 79th*, pages 1–5, May 2014.

[12] You Zhou, Dongfeng Yuan, Xiaotian Zhou, and Haixia Zhang. Trellis coded generalized spatial modulation. In *Vehicular Technology Conference (VTC Spring), 2014 IEEE 79th*, pages 1–5, May 2014.

[13] E. W. Ng and M. Geller. A table of integrals of the error functions. *Journal of Research of the National Bureau of Standards - B. Mathematical Sciences*, 73B(1), Jan-Mar 1969.

[14] D. Micciancio. The hardness of the closest vector problem with preprocessing. *Information Theory, IEEE Transactions on*, 47(3):1212–1215, Mar 2001.

[15] J. Jalden and B. Ottersten. On the complexity of sphere decoding in digital communications. *Signal Processing, IEEE Transactions on*, 53(4):1474–1484, April 2005.

[16] B. Hassibi and H. Vikalo. On the sphere-decoding algorithm i. expected complexity. *Signal Processing, IEEE Transactions on*, 53(8):2806–2818, Aug 2005.

[17] A.R. Murugan, H. El Gamal, M.O. Damen, and G. Caire. A unified framework for tree search decoding: rediscovering the sequential decoder. *Information Theory, IEEE Transactions on*, 52(3):933–953, March 2006.

[18] M.O. Damen, H. El Gamal, and G. Caire. On maximum-likelihood detection and the search for the closest lattice point. *Information Theory, IEEE Transactions on*, 49(10):2389–2402, Oct 2003.

[19] E. Agrell, T. Eriksson, A. Vardy, and K. Zeger. Closest point search in lattices. *Information Theory, IEEE Transactions on*, 48(8):2201–2214, Aug 2002.

[20] A.K. Lenstra, H.W.jun. Lenstra, and Lászlo Lovász. Factoring polynomials with rational coefficients. *Math. Ann.*, 261:515–534, 1982.

[21] C. P. Schnorr and M. Euchner. Lattice basis reduction: Improved practical algorithms and solving subset sum problems. In *Math. Programming*, pages 181–191, 1993.

[22] Jiaxian Pan, Wing-Kin Ma, and J. Jalden. Mimo detection by lagrangian dual maximum-likelihood relaxation: Reinterpreting regularized lattice decoding. *Signal Processing, IEEE Transactions on*, 62(2):511–524, Jan 2014.

[23] R. Gowaikar and B. Hassibi. Statistical pruning for near-maximum likelihood decoding. *Signal Processing, IEEE Transactions on*, 55(6):2661–2675, June 2007.

[24] A. Younis, S. Sinanovic, M. Di Renzo, R. Mesleh, and H. Haas. Generalised sphere decoding for spatial modulation. *Communications, IEEE Transactions on*, 61(7):2805–2815, July 2013.

[25] J.A. Cal-Braz and R. Sampaio-Neto. Low-complexity sphere decoding detector for generalized spatial modulation systems. *Communications Letters, IEEE*, 18(6):949–952, June 2014.

[26] Sunil Arya, David M. Mount, Nathan S. Netanyahu, Ruth Silverman, and Angela Y. Wu. An optimal algorithm for approximate nearest neighbor searching fixed dimensions. *J. ACM*, 45(6):891–923, November 1998.

[27] Alexandr Andoni and Piotr Indyk. Near-optimal hashing algorithms for approximate nearest neighbor in high dimensions. *Commun. ACM*, 51(1):117–122, January 2008.

[28] David Arthur and Sergei Vassilvitskii. K-means++: The advantages of careful seeding. In *Proceedings of the Eighteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '07, pages 1027–1035, Philadelphia, PA, USA, 2007. Society for Industrial and Applied Mathematics.

[29] Y. Linde, A. Buzo, and R.M. Gray. An algorithm for vector quantizer design. *Communications, IEEE Transactions on*, 28(1):84–95, Jan 1980.

[30] Piotr Indyk and Rajeev Motwani. Approximate nearest neighbors: Towards removing the curse of dimensionality. In *Proceedings of the Thirtieth Annual ACM Symposium on Theory of Computing*, STOC '98, pages 604–613, New York, NY, USA, 1998. ACM.