SPACE–TIME BLOCK CODES AND THE COMPLEXITY OF SPHERE DECODING

Miia Mäki

Master's Thesis
July 2008

DEPARTMENT OF MATHEMATICS
UNIVERSITY OF TURKU
FIN-20014 TURKU
FINLAND

Contents

1 Introduction

In wireless communications the transmitted signals may be affected by noise. The receiver must decode the received message, which can be mathematically modelled as a search for the closest lattice point to a given vector. This problem is known to be NP-hard in general, but for communications applications there exist algorithms that, for a certain range of system parameters, offer polynomial expected complexity.

The purpose of this thesis is to study the sphere decoding algorithm introduced in [6] and especially its computational complexity when used in space–time coding. Computer simulations are used to study how different system parameters affect the computational complexity of the algorithm. The aim is to find ways to improve the algorithm from the complexity point of view. The limited battery life of mobile devices provides a motivation for finding algorithms with lower complexity and thus lower power consumption.

We start by reviewing the theory of lattices in Chapter 2 for some necessary background knowledge. In Chapter 3 the principles of space–time coding are explained and criteria for designing space–time block codes are introduced. In Chapter 4 the sphere decoding algorithm is discussed in detail.

Chapter 5 contains an analysis of the complexity of sphere decoding with computer simulation results. In Chapter 6 the concept of collapsing lattices is introduced and its effect on the complexity of sphere decoding is explained.

The main contribution of this thesis is in Chapter 7, where we search for ways to improve the sphere decoding algorithm and develop two new modifications to the algorithm, that are shown to perform faster than the original within a range of system parameters.

1.1 Notations

The following notation is used throughout this thesis.

$\mathbb{R}$	Field of real numbers
$\mathbb{C}$	Field of complex numbers
$\mathbb{Z}$	Ring of rational integers
i	$\sqrt{-1}$
$\mathbb{Z}[i]$	Ring of Gaussian integers
ω	$\frac{1}{2}(-1+i\sqrt{3})$
$\mathbb{Z}[\omega]$	Ring of Eisenstein integers
$\Re(x)$	Real part of x
$\Im(x)$	Imaginary part of x
x^*	Complex conjugate
$\mathbf{x}$	Vector
$\|\cdot\|$	Euclidean norm
$\langle\cdot,\cdot\rangle$	Inner product on $\mathbb{R}^n$
$\mathbf{X}$	Matrix
$\mathbf{X}^T$	Transpose
$\mathbf{X}^*$	Hermitian conjugate
$\det \mathbf{X}$	Determinant of $\mathbf{X}$
$\mathbf{I}_n$	$n \times n$ identity matrix
E	Expectation
$\lfloor\cdot\rceil$	Rounding to the closest integer

2 Lattices

In this chapter the basic concepts of lattices are reviewed for later use in this thesis. All of the results are given without proof. For more details see e.g. [5].

2.1 Bases

Let m and n be two positive integers such that $n \leq m$. A subset Λ of $\mathbb{R}^m$ is called a *lattice* of dimension n if there exist n linearly independent m-dimensional vectors $\mathbf{b}_1, \ldots, \mathbf{b}_n \in \mathbb{R}^m$ such that

$$\Lambda = \sum_{i=1}^{n} \mathbb{Z}\mathbf{b}_i = \{r_1\mathbf{b}_1 + \ldots + r_n\mathbf{b}_n | r_i \in \mathbb{Z}, 1 \leq i \leq n\}. \tag{1}$$

The set of vectors $\mathbf{b}_1, \ldots, \mathbf{b}_n$ is called the *basis* of the lattice Λ, and n is called the *rank* of Λ. The vectors of a lattice Λ form a group under addition: if $\mathbf{a} \in \Lambda$ then $-\mathbf{a} \in \Lambda$; and if $\mathbf{a}$, $\mathbf{b} \in \Lambda$ then $\mathbf{a} \pm \mathbf{b} \in \Lambda$.

The lattice can also be expressed in matrix form $\Lambda = \{\mathbf{x} | \mathbf{x} = \mathbf{Br}\}$ where $\mathbf{B}$ is an $m \times n$ matrix $\mathbf{B} = [\mathbf{b}_1, \ldots, \mathbf{b}_n]$ and $\mathbf{r}$ is an integer column vector. $\mathbf{B}$ is called the *basis matrix* of Λ.

The basis is not uniquely determined by the lattice, but the same lattice can be generated by several different bases. Let $\mathbf{b}'_i$ be the points

$$\mathbf{b}'_i = \sum_j v_{ij}\mathbf{b}_j, \qquad (1 \leq i, j \leq n),$$

where v_{ij} are any integers such that $\det(v_{ij}) = \pm 1$. Now

$$\mathbf{b}_i = \sum_j w_{ij}\mathbf{b}'_j$$

with integral w_{ij}. It then follows that (1) and the lattice

$$\Lambda' = r'_1\mathbf{b}'_1 + \ldots + r'_n\mathbf{b}'_n,$$

where $r'_1, \ldots, r'_n$ run through all integers, are precisely the same set of points, $\Lambda = \Lambda'$. That is, $\mathbf{b}_1, \ldots, \mathbf{b}_n$ and $\mathbf{b}'_1, \ldots, \mathbf{b}'_n$ are bases of the same lattice. Furthermore, every basis $\mathbf{b}'_i$ of a lattice Λ may be obtained from a given basis $\mathbf{b}_i$ in this way [5].

2.2 Lattices under Linear Transformation

Let us consider briefly the effect of a non-singular affine transformation $\mathbf{x} \rightarrow \mathbf{y} = \boldsymbol{\alpha}\mathbf{x}$ of n-dimensional space into itself. Let the transformation $\mathbf{y} = \boldsymbol{\alpha}\mathbf{x}$ be given by

$$y_i = \sum_{1 \leq j \leq n} \alpha_{ij}x_j \qquad (1 \leq i \leq n),$$

where $\mathbf{y} = (y_1, \ldots, y_n)$ and $\mathbf{x} = (x_1, \ldots, x_n)$ are corresponding points in the transformation and α_{ij} are real numbers such that

$$\det(\boldsymbol{\alpha}) = \det(\alpha_{ij}) \neq 0.$$

Let Λ be a lattice and denote by $\boldsymbol{\alpha}\Lambda$ the set of points $\boldsymbol{\alpha}\mathbf{x}$, $\mathbf{x} \in \Lambda$. If $\mathbf{b}_1, \ldots, \mathbf{b}_n$ is a basis for Λ, then the general lattice point $\mathbf{x} = r_1\mathbf{b}_1 + \ldots + r_n\mathbf{b}_n$ (with $r_1, \ldots, r_n$ integers) of Λ maps to

$$\boldsymbol{\alpha}\mathbf{x} = \boldsymbol{\alpha}(r_1\mathbf{b}_1 + \ldots + r_n\mathbf{b}_n) = r_1\boldsymbol{\alpha}\mathbf{b}_1 + \ldots + r_n\boldsymbol{\alpha}\mathbf{b}_n.$$

Hence $\boldsymbol{\alpha}\Lambda$ is a lattice with basis $\boldsymbol{\alpha}\mathbf{b}_1, \ldots, \boldsymbol{\alpha}\mathbf{b}_n$.

2.3 QR-decomposition

The basis vectors of a lattice are not necessarily orthogonal. One method for orthogonalizing the basis is the following Gram–Schmidt procedure. However, the resulting basis does not necessarily span the same lattice, since the coefficients are real numbers and not necessarily integers, but the basis vectors span the same vector space as the original basis.

Gram–Schmidt orthogonalization: Let $\mathbf{b}_i, \ldots, \mathbf{b}_n \in \mathbb{R}^n$ be a set of linearly independent vectors. The vectors $\mathbf{u}_i$ $(1 \leq i \leq n)$ are inductively defined by

$$\begin{aligned} \mathbf{u}_1 &= \mathbf{b}_1 \\ \mathbf{u}_i &= \mathbf{b}_i - \sum_{j=1}^{i-1} \phi_{i,j}\mathbf{u}_j, \end{aligned}$$

where the Gram–Schmidt coefficients $\phi_{i,j} \in \mathbb{R}$ are defined by

$$\phi_{i,j} = \frac{\langle \mathbf{b}_i, \mathbf{u}_j \rangle}{\langle \mathbf{u}_j, \mathbf{u}_j \rangle} = \frac{\langle \mathbf{b}_i, \mathbf{u}_j \rangle}{\|\mathbf{u}_j\|^2}.$$

In particular, $\phi_{i,i} = 1$ and $\phi_{i,j} = 0$ for $j > i$. Notice that $\mathbf{u}_i$ is the projection of $\mathbf{b}_i$ on the orthogonal complement of $\sum_{j=1}^{i-1} \mathbb{R}\mathbf{b}_j$, and that $\sum_{j=1}^{i-1} \mathbb{R}\mathbf{b}_j = \sum_{j=1}^{i-1} \mathbb{R}\mathbf{u}_j$, for $1 \leq i \leq n$. It follows that $\mathbf{u}_1, \ldots, \mathbf{u}_n$ is an orthogonal basis of $\mathbb{R}^n$.

With the help of the Gram–Schmidt procedure one can obtain the QR-decomposition of the matrix $\mathbf{B} = [\mathbf{b}_1, \ldots, \mathbf{b}_n]$, where the vectors $\mathbf{b}_i$ are the columns of the matrix. Let us denote $\mathbf{e}_k = \frac{\mathbf{u}_k}{\|\mathbf{u}_k\|}$ for $k = 1, 2, ..., n$. Since

$$\phi_{i,j}\mathbf{u}_j = \frac{\langle \mathbf{b}_i, \mathbf{u}_j \rangle}{\|\mathbf{u}_j\|^2}\mathbf{u}_j = \frac{\langle \mathbf{b}_i, \mathbf{e}_j\|\mathbf{u}_j\| \rangle}{\|\mathbf{u}_j\|^2}\mathbf{e}_j\|\mathbf{u}_j\| = \langle \mathbf{b}_i, \mathbf{e}_j \rangle \mathbf{e}_j,$$

by rearranging the equations above we have:

$$\begin{aligned}
\mathbf{b}_1 &= \mathbf{e}_1 \left\| \mathbf{u}_1 \right\|, \\
\mathbf{b}_2 &= \langle \mathbf{b}_2, \mathbf{e}_1 \rangle \mathbf{e}_1 + \mathbf{e}_2 \left\| \mathbf{u}_2 \right\|, \\
&\vdots \\
\mathbf{b}_n &= \sum_{j=1}^{n-1} \langle \mathbf{b}_n, \mathbf{e}_j \rangle \mathbf{e}_j + \mathbf{e}_n \left\| \mathbf{u}_n \right\|.
\end{aligned}$$

By writing the equations in matrix form we get the QR-decomposition of the matrix **B**:

$$\begin{aligned}
\mathbf{B} &= \mathbf{QR} \\
&= \begin{bmatrix} \mathbf{e}_1 & \mathbf{e}_2 & \dots & \mathbf{e}_n \end{bmatrix}
\begin{bmatrix}
\|\mathbf{u}_1\| & \langle \mathbf{b}_2, \mathbf{e}_1 \rangle & \langle \mathbf{b}_3, \mathbf{e}_1 \rangle & \dots & \langle \mathbf{b}_n, \mathbf{e}_1 \rangle \\
0 & \|\mathbf{u}_2\| & \langle \mathbf{b}_3, \mathbf{e}_2 \rangle & \dots & \langle \mathbf{b}_n, \mathbf{e}_2 \rangle \\
0 & 0 & \|\mathbf{u}_3\| & & \vdots \\
\vdots & \vdots & & \ddots & \langle \mathbf{b}_n, \mathbf{e}_{n-1} \rangle \\
0 & 0 & \dots & 0 & \|\mathbf{u}_n\|
\end{bmatrix}.
\end{aligned}$$

Any $m \times n$ matrix **A** with linearly independent columns can be factorized into a product $\mathbf{A} = \mathbf{QR}$, where the columns of **Q** are orthonormal and **R** is upper triangular and invertible [8].

2.4 Closest Vector Problem

The closest vector problem (CVP) is the problem of finding, for a given lattice Λ and a given input point $\mathbf{y} \in \mathbb{R}^m$, the lattice point that is closest to $\mathbf{y}$. More precisely, to find a vector $\hat{\mathbf{x}} \in \Lambda$ such that

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

For orthogonal lattices, such as the n-dimensional integer lattice $\mathbb{Z}^n$, the solution is simple. However, in the general case the lattice is not necessarily orthogonal, but skewed. It has been shown [1] that the general closest vector problem as a function of the dimension m is NP-hard. Thus, all known algorithms for solving the problem optimally have exponential complexity.

For example, the maximum-likelihood decoding of space–time codes can be reduced to a closest vector search. In the following chapters this will be explained in more detail.

3 Space–Time Block Codes

Space-–time coding is a method to improve the reliability of data transmission by using multiple antennas. In this chapter the MIMO (multiple-input multiple-output) channel model and two space–time lattice codes are introduced. In later chapters the properties of these two lattice codes are compared with each other.

3.1 Lattice Codes

A lattice codeword is a matrix from a constellation

$$\mathcal{L} = \left\{ \sum_{i=1}^{k} a_i \mathbf{X}_i | \mathbf{X}_i \in \mathcal{M}_{m\times l}(\mathbb{C}), a_i \in S \subset \mathbb{Z} \right\},$$

where $\mathbf{X}_i$, $(i = 1, \ldots, k)$, are constant basis matrices and a_i, $(i = 1, \ldots, k)$, are integer variables from some finite signal set S of size q. The information to be transmitted is coded in the coefficients $(a_1, \ldots, a_k) \in S^k$. Throughout the thesis, the terms "lattice code" and "block code" are used interchangeably to refer to a code used in space–time coding.

In this thesis we use the pulse amplitude modulation (PAM) signal set of size q, i.e.,

$$S = \{a = 2u - q + 1 | u \in \mathbb{Z}_q\}$$

with $\mathbb{Z}_q = \{0, 1, \ldots, q-1\}$.

The size of the signal set is normally some power of 2, i.e. $q = 2^b$ and b information bits are mapped to each symbol. A commonly used mapping is the Gray encoding [12], where the adjacent symbols differ from each other by only one binary digit (as illustrated in Figure 1). This mapping is preferred, because the most likely errors caused by noise involve the erroneous selection of an adjacent symbol to the one that was transmitted. In this case a symbol error results in only a single bit error in the b-bit sequence.

The following two examples of lattice codes were presented in the article [10]. In later chapters these codes will be used in the simulations and their performance with regard to decoding complexity will be analyzed.

The block code L_{NF}:

$$\mathbf{M}_{NF}(c_1, c_2, c_3, c_4) = \begin{pmatrix} c_1 & ic_4 & ic_3 & ic_2 \\ c_2 & c_1 & ic_4 & ic_3 \\ c_3 & c_2 & c_1 & ic_4 \\ c_4 & c_3 & c_2 & c_1 \end{pmatrix}$$

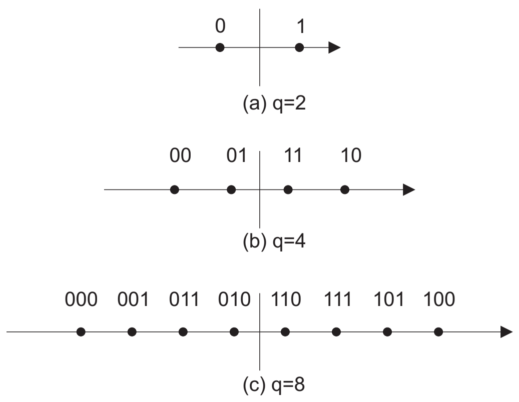


Figure 1: Gray encoding for PAM signals

and the block code L_{QNF}:

$$\mathbf{M}_{QNF}(c_1, c_2, c_3, c_4) = \begin{pmatrix} c_1 & ic_2 & -c_3^* & -c_4^* \\ c_2 & c_1 & ic_4^* & -c_3^* \\ c_3 & ic_4 & c_1^* & c_2^* \\ c_4 & c_3 & -ic_2^* & c_1^* \end{pmatrix},$$

where c_i $(1 \leq i \leq 4)$ are Gaussian integers; $c_i \in \mathbb{Z}[i] = \{a + bi | a, b \in \mathbb{Z}\}$.

Both of the above block codes have 8 basis matrices $\mathbf{X}_i$, which are $\mathbf{X}_1 = \mathbf{M}(1,0,0,0)$, $\mathbf{X}_2 = \mathbf{M}(i,0,0,0)$, $\mathbf{X}_3 = \mathbf{M}(0,1,0,0)$, $\mathbf{X}_4 = \mathbf{M}(0,i,0,0)$, $\mathbf{X}_5 = \mathbf{M}(0,0,1,0)$, $\mathbf{X}_6 = \mathbf{M}(0,0,i,0)$, $\mathbf{X}_7 = \mathbf{M}(0,0,0,1)$ and $\mathbf{X}_8 = \mathbf{M}(0,0,0,i)$.

For example, the codeword $\mathbf{M}(c_1, c_2, c_3, c_4)$ is generated by

$$\mathbf{M}(c_1, c_2, c_3, c_4) = \sum_{i=1}^{4} (\Re(c_i)\mathbf{X}_{2i-1} + \Im(c_i)\mathbf{X}_{2i})$$

3.2 MIMO Channel

Consider a multiple antenna system with m transmit and n receive antennas. The codewords $\mathbf{X}$ are $m \times l$ complex matrices from some constellation $\mathcal{L} \subset \mathcal{M}_{m \times l}(\mathbb{C})$, where l is the length of the code, $l \geq m$,

$$\mathbf{X} = \begin{pmatrix} \mathbf{x}_1 \\ \mathbf{x}_2 \\ \vdots \\ \mathbf{x}_m \end{pmatrix} \in \mathcal{M}_{m \times l}(\mathbb{C}).$$

The matrix rows $\mathbf{x}_i$ correspond to the different transmit antennas and the columns to the different time slots. At each time slot t, signals $x_{i,t}$ $(i = 1, \ldots, m)$ are transmitted simultaneously from the m transmit antennas, as depicted in Figure 2.

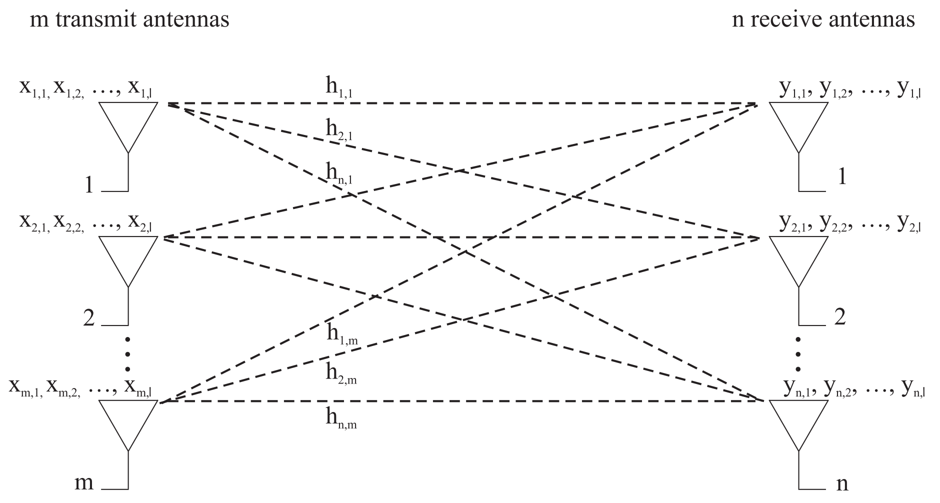

m transmit antennas
n receive antennas
$x_{1,1}, x_{1,2}, \ldots, x_{1,l}$
$x_{2,1}, x_{2,2}, \ldots, x_{2,l}$
$x_{m,1}, x_{m,2}, \ldots, x_{m,l}$
$y_{1,1}, y_{1,2}, \ldots, y_{1,l}$
$y_{2,1}, y_{2,2}, \ldots, y_{2,l}$
$y_{n,1}, y_{n,2}, \ldots, y_{n,l}$
$h_{1,1}$
$h_{2,1}$
$h_{n,1}$
$h_{1,m}$
$h_{2,m}$
$h_{n,m}$
1
2
m
1
2
n

Figure 2: Space–time coding in a quasi-static Rayleigh fading channel

In a general fading channel the receiver obtains the signal

$$\begin{aligned}
\mathbf{Y} &= \mathbf{HX} + \mathbf{N} \\
&= \begin{pmatrix} h_{1,1} & \ldots & h_{1,m} \\ \vdots & \ddots & \vdots \\ h_{n,1} & \ldots & h_{n,m} \end{pmatrix} \begin{pmatrix} x_{1,1} & \ldots & x_{1,l} \\ \vdots & \ddots & \vdots \\ x_{m,1} & \ldots & x_{m,l} \end{pmatrix} + \begin{pmatrix} \eta_{1,1} & \ldots & \eta_{1,l} \\ \vdots & \ddots & \vdots \\ \eta_{n,1} & \ldots & \eta_{n,l} \end{pmatrix}
\end{aligned}$$

where $h_{i,j}$ denotes the path gain from transmit antenna j to receive antenna i and $\mathbf{N} = (\eta_{i,j})$ represents additive white Gaussian noise. It is assumed that the antennas are sufficiently spaced such that the fadings are uncorrelated.

Here we use the Rayleigh fading channel model, where the path gains $h_{i,j}$ are modelled as samples of independent and identically distributed (i.i.d.) complex Gaussian random variables with zero mean and variance τ^2 per dimension, as it is assumed that signals received at different antennas experience independent fading. The fading channel is assumed to be quasi-static, i.e., the path gains are assumed to remain constant for the interval T of transmitting a single codeword after which they change and again remain constant for the next codeword.

The noise components $\eta_{i,t}$ at receive antenna i at time t are modelled as samples of i.i.d. complex Gaussian random variables with zero mean and variance σ^2 per dimension.

The component $y_{i,j}$ of the received matrix corresponds to the j-th signal received by antenna i, which is a superposition of the faded versions of all of the m transmitted signals, with additive noise. For $1 \leq i \leq n$, $1 \leq j \leq m$

$$y_{i,j} = h_{i,1}x_{1,j} + h_{i,2}x_{2,j} + \ldots + h_{i,m}x_{m,j} + \eta_{i,j}.$$

The *signal-to-noise ratio* (SNR) is the ratio of the expected energy of the transmitted signal to that of the additive noise per receive antenna.

$$SNR = \frac{E[P(\mathbf{HX})]}{E[P(\overline{\eta}_r)]},$$

where

$$E[P(\overline{\eta}_r)] = E\left[\sum_{i=1}^{l} \eta_{r,i}\eta_{r,i}^*\right] = 2l\sigma^2$$

is the expected energy of the noise affecting a single receive antenna r during the transmission of one codeword, and

$$E[P(\mathbf{HX})] = 2\tau^2 P_{\mathcal{L}}$$

is the expected energy of the corresponding transmitted signal, where $P_{\mathcal{L}}$ denotes the average energy of a codeword $\mathbf{X}$ from the constellation $\mathcal{L} = \{\sum_{i=1}^{k} a_i\mathbf{X}_i | a_i \in S\}$

$$\begin{aligned} P_{\mathcal{L}} &= \frac{1}{|\mathcal{L}|}\sum_{a_1\in S}\cdots\sum_{a_k\in S} P\left(\sum_{i=1}^{k} a_i\mathbf{X}_i\right) \\ &= \frac{1}{|\mathcal{L}|}\sum_{\mathbf{X}\in\mathcal{L}}\left(\sum_{i=1}^{m}\sum_{j=1}^{l}|x_{i,j}|^2\right). \end{aligned}$$

The size of the constellation is $|\mathcal{L}| = |S|^k = q^k$. Now

$$SNR = \frac{2\tau^2 P_{\mathcal{L}}}{2l\sigma^2} = \frac{P_{\mathcal{L}}}{l}\frac{\tau^2}{\sigma^2}.$$

3.3 Decoding

The receiver can be assumed to have perfect channel state information, i.e., the channel matrix $\mathbf{H}$ is known at the receiver. It is usually measured with so-called pilot signals. Then the set of possible messages is

$$\mathbf{H}\mathcal{L} = \left\{\sum_{i=1}^{k} a_i\mathbf{HX}_i\right\} \subseteq \mathcal{M}_{n\times l}(\mathbb{C}).$$

The problem of decoding is, given the received matrix $\mathbf{Y}$ and the matrices $\mathbf{HX}_1, \ldots, \mathbf{HX}_k$, to find the coefficients $a_1, \ldots, a_k$ for $\hat{\mathbf{X}} = \mathbf{HX} = \sum_{i=1}^{k} a_i\mathbf{HX}_i$ such that the metric

$$d(\mathbf{Y} - \hat{\mathbf{X}})^2 = \sum_{i=1}^{n}\sum_{j=1}^{l}|y_{i,j} - \hat{x}_{i,j}|^2 \tag{2}$$

is minimized. This problem can be transformed into the problem of finding the closest lattice point. In the next chapter an algorithm for finding the closest lattice point is presented.

The decision metric (2) is based on the *maximum likelihood* (ML) principle, in which the conditional probability $p(\mathbf{Y}|\mathbf{H},\mathbf{X})$ is maximized. In the following it is assumed that all the codewords from the constellation have an equal probability of being transmitted.

$$\begin{aligned}
p(\mathbf{Y}|\mathbf{H},\mathbf{X}) &= \prod_{i=1}^{n}\prod_{j=1}^{l} p(y_{i,j}|(\mathbf{HX})_{i,j}) \\
&= \prod_{i=1}^{n}\prod_{j=1}^{l} \left(\frac{1}{\sqrt{2\pi\sigma^2}} \exp \frac{-(y_{i,j}-(\mathbf{HX})_{i,j})^2}{2\sigma^2}\right) \\
&= \left(\frac{1}{\sqrt{2\pi\sigma^2}}\right)^{n\times l} \exp \frac{-\sum_{i=1}^{n}\sum_{j=1}^{l}(y_{i,j}-(\mathbf{HX})_{i,j})^2}{2\sigma^2} \\
&= \left(\frac{1}{\sqrt{2\pi\sigma^2}}\right)^{n\times l} \exp \frac{-d(\mathbf{Y}-\mathbf{HX})^2}{2\sigma^2}
\end{aligned}$$

The probability $p(\mathbf{Y}|\mathbf{H},\mathbf{X})$ is the probability of receiving $\mathbf{Y}$ after the codeword $\mathbf{X}$ has been sent through the channel $\mathbf{H}$. This probability (over all possible codewords) is maximum for the codeword $\mathbf{X}'$ for which the metric $d(\mathbf{Y}-\mathbf{HX}')^2$ of the matrices $\mathbf{Y}$ and $\mathbf{HX}'$ is minimized.

3.4 Criteria for Code Construction

The main priority for code design in space–time coding is the so-called *rank criterion* [14]. The rank criterion states that the difference of any two distinct codewords should have full rank. If the rank criterion is satisfied, the code is said to have *full diversity.* In that case $\mathbf{HX}_1 \neq \mathbf{HX}_2$ for all channel matrices $\mathbf{H} \neq \mathbf{0}$ and all codewords $\mathbf{X}_1$, $\mathbf{X}_2$ where $\mathbf{X}_1 \neq \mathbf{X}_2$.

Another criterion for code construction is the *determinant criterion* [14]. Let us denote the difference of two codewords by $\Delta_{12} = \mathbf{X}_1 - \mathbf{X}_2$ and let $\mathbf{E}_{12} = \Delta_{12}\Delta_{12}^*$. The determinant criterion sets as a design target that the minimum of the determinant of $\mathbf{E}_{12}$ taken over all pairs of distinct codewords $\mathbf{X}_1$ and $\mathbf{X}_2$ should be maximized. In the article [14] it was shown that for high SNR the probability of making an error between $\mathbf{X}_1$ and $\mathbf{X}_2$ is inversely proportional to a power of $\det \mathbf{E}_{12}/\sigma$. Furthermore, the rank criterion maximizes that exponent.

The *code rate* is the number of symbols transmitted in one code block divided by the number of time slots needed to send one block. In this context a symbol refers to an element of a 2-dimensional alphabet $S \subset \mathbb{C}$. Usually $S = \mathbb{Z}[i]$ or $\mathbb{Z}[\omega]$. The code rate is limited by the number of antennas, so that the code rate $\leq \min\{\#$ of Tx antennas, $\#$ of Rx antennas$\}$.

A block code is said to have *full rate* if its rate equals the number of transmit antennas. In that case there must be at least as many receive antennas in the setting, too.

Block codes constructed from orthogonal designs have been shown to possess many advantages [13]. They provide full diversity with maximal rate and have simple maximum-likelihood decoding algorithms with linear processing at the receiver.

Definition 3.1. A *complex orthogonal design* of size n is a $n \times n$ matrix $\mathbf{G}$ whose entries are the indetermined variables $\pm x_1, \pm x_2, ..., \pm x_n$, their conjugates $\pm x_1^*, \pm x_2^*, ..., \pm x_n^*$ and multiples of these by $i = \sqrt{-1}$ such that

$$\mathbf{G}^*\mathbf{G} = \left(|x_1|^2 + |x_2|^2 + \cdots + |x_n|^2\right) \mathbf{I}_n.$$

Example 3.1. The so-called Alamouti scheme is an example of a complex orthogonal design. It uses 2 transmit antennas and has rate 1. The transmitted 2×2 codeword is

$$\mathbf{A}(c_1, c_2) = \begin{pmatrix} c_1 & c_2 \\ -c_2^* & c_1^* \end{pmatrix} \qquad c_1, c_2 \in \mathbb{C}$$

and the receiver obtains the signal $\mathbf{y} = [y_1, y_2] = \mathbf{hA} + \mathbf{n}$.

$$\mathbf{A}^*\mathbf{A} = \begin{pmatrix} |c_1|^2 + |c_2|^2 & 0 \\ 0 & |c_1|^2 + |c_2|^2 \end{pmatrix}$$

Let

$$\mathbf{A}_1 = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}, \mathbf{A}_2 = \begin{pmatrix} i & 0 \\ 0 & -i \end{pmatrix}, \mathbf{A}_3 = \begin{pmatrix} 0 & 1 \\ -1 & 0 \end{pmatrix}, \mathbf{A}_4 = \begin{pmatrix} 0 & i \\ i & 0 \end{pmatrix}$$

denote the basis matrices. Then for every non-zero $\mathbf{h} \in \mathbb{C}^2 \backslash \{\mathbf{0}\}$ the vectors $\mathbf{hA}_i$ and $\mathbf{hA}_j$, $i \neq j$, are mutually orthogonal, since

$$\langle \mathbf{hA}_i \, , \mathbf{hA}_j \rangle_{\mathbb{R}} = \begin{cases} 0 & \text{if } i \neq j, \\ |h_1|^2 + |h_2|^2 & \text{if } i = j. \end{cases}$$

Due to this fact the Alamouti scheme has a very simple ML-decoding algorithm [2].

However, it has been shown that such full rate complex orthogonal designs do not exist for more than two transmit antennas [13]. Block codes derived from so-called *generalized complex orthogonal designs* still provide orthogonality and thus have simple ML-decoding, but they have smaller rate.

Definition 3.2. A generalized complex orthogonal design of size n is a $p \times n$ matrix $\mathbf{G}$ whose entries are $0, \pm x_1, \pm x_1^* \pm x_2, \pm x_2^*, ..., \pm x_k, \pm x_k^*$ or their product with i such that

$$\mathbf{G}^*\mathbf{G} = \kappa \left(|x_1|^2 + |x_2|^2 + \cdots + |x_k|^2\right) \mathbf{I}_n,$$

where κ is a constant. $\mathbf{G}$ has rate $R = k/p$.

Example 3.2. For four antennas the next example of a generalized complex orthogonal design provides rate 3/4, i.e., only three complex symbols are transmitted in four time slots.

$$\mathbf{M}_{OD}(c_1, c_2, c_3) = \begin{pmatrix} c_1 & c_2 & c_3 & 0 \\ -c_2^* & c_1^* & 0 & c_3 \\ c_3^* & 0 & -c_1^* & c_2 \\ 0 & c_3^* & -c_2^* & -c_1 \end{pmatrix} \qquad (c_1, c_2, c_3 \in \mathbb{C}).$$

It spans a 6-dimensional vector space over $\mathbb{R}$ with the basis

$$\{\mathbf{M}_{OD}(1,0,0), \mathbf{M}_{OD}(i,0,0), \ldots, \mathbf{M}_{OD}(0,0,i)\}$$

and it has the property

$$\mathbf{M}_{OD}^*\mathbf{M}_{OD} = (|c_1|^2 + |c_2|^2 + |c_3|^2)\mathbf{I}_4.$$

The two block codes L_{NF} and L_{QNF} that were introduced in Section 3.1 do not have the structure of a general orthogonal design. This means that the decoding at the receiver will be more complex using these codes. They do, however, have higher rate as both of them have rate 1. In the next chapters we will show that there exist algorithms that in practical situations provide a reasonable average decoding complexity for these codes.

4 Sphere Decoder

In general the problem of finding the closest lattice point is prohibitively complex. Indeed, certain cryptosystems are based on the difficulty of solving this and the related problem of finding the shortest vector in a lattice of a rank ranging in the hundreds. However, in communications applications the rank of the lattice remains moderate and the search is usually constrained among the finite set of points with coordinates within the prescribed signal set. Nevertheless, the number of valid combinations of coordinates is still too large for the simple approach of an exhaustive search. For example, with a lattice code of rank 8 and a signal set of size 8 there would be $8^8 \approx 16.8$ million different lattice points through which to search.

In communications applications the given vector is not arbitrary but rather an unknown lattice point that has been corrupted by an additive noise vector with known statistical properties. Therefore it is meaningful to consider the expected (average) complexity instead of the worst-case complexity. It has been shown that the sphere decoder algorithm, which is described in this chapter, has a polynomial time average complexity in many relevant cases, for a range of system parameters [6]. This finding is also supported by the simulation results in Chapter 5.

This chapter is based on the articles [6] and [9].

4.1 Introduction to Sphere Decoding

The basic idea of sphere decoding is to search for the closest lattice point only among points within a certain hypersphere of radius r around the given vector $\mathbf{y}$, instead of an exhaustive search over all lattice points. Clearly the closest lattice point inside the hypersphere is also the closest point in the whole lattice. Figure 3 illustrates this idea behind the sphere decoder in a 2-dimensional lattice.

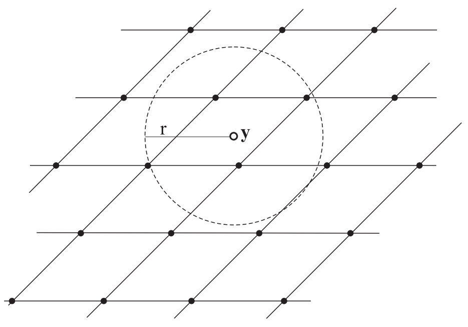


Figure 3: Closest lattice point in a 2-dimensional lattice

First of all, it is necessary to know which lattice points lie inside the hypersphere. Testing the distances between $\mathbf{y}$ and every other lattice point would still result in an exhaustive search and would therefore offer no reduction to the overall complexity of decoding. An efficient way of finding the points is explained in the following. Instead of considering a general m-dimensional hypersphere, we can start by considering the one-dimensional case where $m = 1$. A one-dimensional hypersphere is simply an interval and therefore the lattice points inside this hypersphere are the integer values lying in this interval. We can now use this fact to move from dimension k to $k+1$. Suppose that we have determined all of the lattice points in a k-dimensional hypersphere of radius r. Then for any such k-dimensional point, the set of admissible values for the $k+1$-th dimensional coordinate that lies in the higher-dimensional hypersphere of the same radius r forms an interval. In other words, we can determine all lattice points in an m-dimensional hypersphere of radius r by successively determining all lattice points in hyperspheres of lower dimensions $1, 2, \ldots, m$ and the same radius r.

Such an algorithm for determining the lattice points inside an m-dimensional hypersphere can be represented by a tree where the branches in the k-th level of the tree correspond to the lattice points inside the k-dimensional hypersphere of the same radius. Figure 4 below shows an example of such a tree generated by finding the lattice points inside a 4-dimensional hypersphere. Furthermore, the complexity of such an algorithm will depend on the size of the tree, i.e., on the number of lattice points visited by the algorithm in different dimensions.

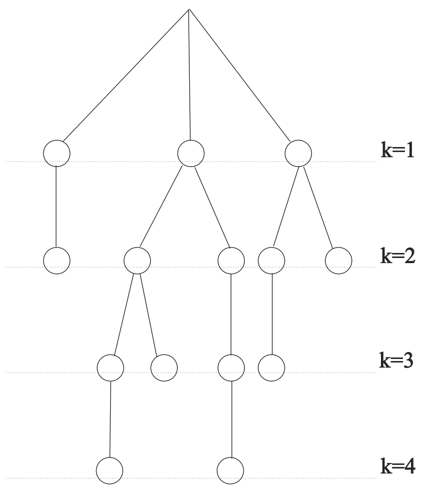


Figure 4: The search tree

4.2 The Pohst Enumeration

This strategy for enumerating all of the lattice points inside a sphere with a certain radius was first proposed by Pohst [11]. This so-called Pohst enumeration can be more specifically outlined as follows [6].

Assume that $\mathbf{B}$ is an $m \times n$ real matrix with $m \geq n$ and $\text{rank}(\mathbf{B}) = n$. For $\mathbf{B} \in \mathbb{R}^{m\times n}$ and $\mathbf{y} \in \mathbb{R}^m$, consider the minimization

$$\hat{\mathbf{x}} = \arg\min_{\mathbf{x}\in\mathbb{Z}^\mathbf{n}} |\mathbf{y} - \mathbf{B}\mathbf{x}|^\mathbf{2}.$$

The set $\Lambda = \{\mathbf{Bx}|\mathbf{x} \in \mathbb{Z}^n\}$ is an n-dimensional (infinite) lattice in $\mathbb{R}^m$. Let C_0 be the squared radius of an m-dimensional sphere $S(\mathbf{y}, \sqrt{C_0})$ centered at $\mathbf{y}$. A lattice point $\mathbf{Bx}$ lies inside a hypersphere of radius $\sqrt{C_0}$ if and only if

$$C_0 \geq |\mathbf{y} - \mathbf{Bx}|^2.$$

Now we wish to produce a list of all of these points of $\Lambda \cap S(\mathbf{y}, \sqrt{C_0})$. In order to break the problem into the subproblems of finding the points successively in each dimension, it is useful to consider the QR factorization of the matrix $\mathbf{B}$. By performing the Gram–Schmidt orthonormalization of the columns of $\mathbf{B}$, we get

$$\mathbf{B} = [\mathbf{Q}, \mathbf{Q}'] \begin{bmatrix} \mathbf{R} \\ \mathbf{0} \end{bmatrix}$$

where $\mathbf{R}$ is an $n \times n$ upper triangular matrix with positive diagonal elements, $\mathbf{0}$ is an $(m-n)\times n$ zero matrix, $\mathbf{Q}$ (resp. $\mathbf{Q}'$) is an $m \times n$ (resp. $m \times (m-n)$) matrix and $[\mathbf{Q}, \mathbf{Q}']$ is orthogonal. The condition $\mathbf{Bx} \in S(\mathbf{y}, \sqrt{C_0})$ can be written as

$$\begin{aligned} |\mathbf{y} - \mathbf{Bx}|^2 &\leq C_0 \\ \left| [\mathbf{Q}, \mathbf{Q}']^T \mathbf{y} - \begin{bmatrix} \mathbf{R} \\ \mathbf{0} \end{bmatrix} \mathbf{x} \right|^2 &\leq C_0 \\ \left| \mathbf{Q}^T\mathbf{y} - \mathbf{Rx} \right|^2 &\leq C_0 - \left| (\mathbf{Q}')^T \mathbf{y}) \right|^2 \\ |\mathbf{y}' - \mathbf{Rx}|^2 &\leq C_0' \end{aligned}$$

where $\mathbf{y}' \triangleq \mathbf{Q}^T\mathbf{y}$ and $C_0' \triangleq C_0 - |(\mathbf{Q}')^T\mathbf{y})|^2$. Because $\mathbf{R}$ has an upper triangular form, the last inequality implies the set of conditions

$$\sum_{j=i}^{n} \left| y_j' - \sum_{l=j}^{n} r_{j,l} x_l \right|^2 \leq C_0', \qquad i = 1, \ldots, n. \tag{3}$$

By considering the above conditions in the descending order from n to 1 (in the same way as in back-substitution when solving a linear upper triangular system), we obtain the set of suitable values of each symbol x_i for fixed values of previous symbols $x_{i+1}, \ldots, x_n$. More explicitly, let $\mathbf{x}_l^n \triangleq (x_l, x_{l+1}, \ldots, x_n)^T$ denote the last $n-l+1$ components of the vector $\mathbf{x}$. When $\mathbf{x}_{i+1}^n$ is fixed, the component x_i can take on integer values in the range $\mathcal{I}_i(\mathbf{x}_{i+1}^n) = [A_i(\mathbf{x}_{i+1}^n), B_i(\mathbf{x}_{i+1}^n)]$ where

$$A_i(\mathbf{x}_{i+1}^n) = \left\lceil \frac{1}{r_{i,i}} \left(y_i' - \sum_{j=i+1}^{n} r_{i,j} x_j - \sqrt{C_0' - \sum_{j=i+1}^{n} \left| y_j' - \sum_{l=j}^{n} r_{j,l} x_l \right|^2} \right) \right\rceil$$

and

$$B_i(\mathbf{x}_{i+1}^n) = \left\lfloor \frac{1}{r_{i,i}} \left(y_i' - \sum_{j=i+1}^{n} r_{i,j} x_j + \sqrt{C_0' - \sum_{j=i+1}^{n} \left| y_j' - \sum_{l=j}^{n} r_{j,l} x_l \right|^2} \right) \right\rfloor .$$

If for some i

$$\sum_{j=i+1}^{n} \left| y_j' - \sum_{l=j}^{n} r_{j,l} x_l \right|^2 \geq C_0'$$

or if $A_i(\mathbf{x}_{i+1}^n) > B_i(\mathbf{x}_{i+1}^n)$, then $\mathcal{I}_i(\mathbf{x}_{i+1}^n) = \emptyset$. In this case, there exists no such value of x_i that would satisfy the inequalities (3) and the points which have the same values of $x_{i+1}, \ldots, x_n$ as $\mathbf{x}_{i+1}^n$ do not lie inside the sphere $S(\mathbf{y}, \sqrt{C_0})$.

The Pohst enumeration starts from level $i = n$ and proceeds climbing up to levels $i = n-1, n-2, \ldots, 1$. At each level i, the variable values chosen earlier at lower levels determine the interval $\mathcal{I}_i(\mathbf{x}_{i+1}^n)$ for x_i. If $\mathcal{I}_1(x_2^n)$ is nonempty, the vectors $\mathbf{x} = (x_1, (\mathbf{x}_2^n)^T)^T$, for all $x_1 \in \mathcal{I}_1(x_2^n)$, correspond to lattice points $\mathbf{Bx} \in S(\mathbf{y}, \sqrt{C_0})$. The squared Euclidean distances between such points and $\mathbf{y}$ are given by

$$d^2(\mathbf{y}, \mathbf{Bx}) = \sum_{j=i}^{n} \left| y_j' - \sum_{l=j}^{n} r_{j,l} x_l \right|^2 .$$

The output of the algorithm is the point $\hat{\mathbf{x}}$ for which this distance is minimum. If no point is found inside the sphere it is declared empty and the search fails. In this case the squared radius C_0 must be increased and the search is performed again.

4.3 The Schnorr–Euchner Enumeration and the Babai Point

In the article [1] Agrell et al. proposed the use of Schnorr–Euchner refinement of the Pohst enumeration in the closest lattice point search. They concluded, based on numerical results, that their sphere decoding algorithm with Schnorr–Euchner enumeration is more efficient than the Viterbo–Boutros implementation that uses Pohst enumeration.

The Pohst enumeration is based on the so-called natural spanning of the intervals $\mathcal{I}_i(x_{i+1}^n)$ at each level i. In other words, x_i takes on values in the ascending order $A_i(\mathbf{x}_{i+1}^n), A_i(\mathbf{x}_{i+1}^n)+1, \ldots, B_i(\mathbf{x}_{i+1}^n)$. The Schnorr–Euchner enumeration is a variation of the Pohst strategy where the intervals are spanned in a zig-zag order, starting from the midpoint

$$S_i(\mathbf{x}_{i+1}^n) = \left\lfloor \frac{1}{r_{i,i}} \left(y_i' - \sum_{j=i+1}^{n} r_{i,j} x_j \right) \right\rceil. \tag{4}$$

Hence, the sequence of values produced by the Schnorr–Euchner enumeration at each level i is

$$x_i \in \{S_i(\mathbf{x}_{i+1}^n), S_i(\mathbf{x}_{i+1}^n)+1, S_i(\mathbf{x}_{i+1}^n)-1, S_i(\mathbf{x}_{i+1}^n)+2, S_i(\mathbf{x}_{i+1}^n)-2, \ldots\} \cap \mathcal{I}_i(x_{i+1}^n)$$

if

$$y_i' - \sum_{j=i+1}^{n} r_{i,j} x_j - r_{i,i} S_i(\mathbf{x}_{i+1}^n) \geq 0$$

or the sequence of values

$$x_i \in \{S_i(\mathbf{x}_{i+1}^n), S_i(\mathbf{x}_{i+1}^n)-1, S_i(\mathbf{x}_{i+1}^n)+1, S_i(\mathbf{x}_{i+1}^n)-2, S_i(\mathbf{x}_{i+1}^n)+2, \ldots\} \cap \mathcal{I}_i(x_{i+1}^n)$$

if

$$y_i' - \sum_{j=i+1}^{n} r_{i,j} x_j - r_{i,i} S_i(\mathbf{x}_{i+1}^n) < 0.$$

Similar to Pohst enumeration, when a given value of x_i results in a point segment $\mathbf{x}_i^n$ outside the sphere, the next value of x_{i+1} (at level $i+1$) is produced.

Every time a vector $\mathbf{x}' \in \mathbb{Z}^n$ is found such that $\mathbf{B}\mathbf{x}' \in S(\mathbf{y}, \sqrt{C_0})$, the squared radius of the sphere can be dynamically updated to $d^2(\mathbf{y}, \mathbf{B}\mathbf{x}')$, since this will obviously be an upper bound for the distance of the closest point. Note that with the Schnorr–Euchner enumeration one can set the squared radius to $C_0 = \infty$, in which case the event of declaring an empty sphere never occurs. The first point found in this case is by definition the Babai point [1]. The Babai point is not necessarily the closest point, but the error can be bounded, i.e., it is a nearby point. This point is also known as the *nulling and cancelling* or *zero-forcing decision-feedback equalization* (ZF-DFE) point. Explicitly, it is given by

$$\begin{aligned} x_i^{zf-dfe} &= S_i(x_{i+1}^{zf-dfe}, \ldots, x_n^{zf-dfe}) \\ &= \left\lfloor \frac{1}{r_{i,i}} \left(y_i' - \sum_{j=i+1}^{n} r_{i,j} x_j^{zf-dfe} \right) \right\rceil \end{aligned}$$

for $i = m, m-1, \ldots, 1$. The procedure begins by first finding the midpoint of the interval $\mathcal{I}_n$ and setting x_n^{zf-dfe} at it. Its effect is then cancelled out by back-substitution, which is enabled by the upper triangular form of $\mathbf{R}$. The same process is then repeated for x_{n-1}^{zf-dfe} and so on.

4.4 The Algorithm

The following algorithm was proposed by Damen et al. [6] and was shown to be very efficient in terms of receiver complexity when compared to other known sphere decoding algorithms. It is a modification of the Schnorr–Euchner enumeration in order to take into account a finite signal set boundary. Here the lattice is not infinite, but $\mathbf{x}$ takes on values from the set $\{0, 1, \ldots, Q-1\}$, i.e., $\Lambda = \{\mathbf{Bx} | \mathbf{x} \in \mathbb{Z}_Q^n\}$.

Algorithm II, Smart Implementation (Input C_0', $\mathbf{y}'$, $\mathbf{R}$, Output $\hat{\mathbf{x}}$):

Step 1 (Initialization) Set $i := n$, $T_n := 0$, $\xi_n := 0$, and $d_c := C_0'$ (current sphere squared radius).

Step 2 (DFE on x_i) Set $x_i := \lfloor (y_i' - \xi_i)/r_{i,i} \rceil$ and $\Delta_i := \text{sign}(y_i' - \xi_i - r_{i,i}x_i)$.

Step 3 (Main step) If $d_c < T_i + |y_i' - \xi_i - r_{i,i}x_i|^2$, then go to Step 4 (i.e., we are outside the sphere).

Else if $x_i \notin [0, Q-1]$ go to Step 6 (i.e., we are inside the sphere but outside the signal set boundaries).

Else (i.e., we are inside the sphere and signal set boundaries) if $i > 1$, then $\{$let $\xi_{i-1} := \sum_{j=1}^{n} r_{i-1,j}x_j$, $T_{i-1} := T_i + |y_i' - \xi_i - r_{i,i}x_i|^2$, $i := i-1$, and go to Step 2$\}$.

Else ($i = 1$) go to Step 5.

Step 4 If $i = n$, terminate, else set $i := i + 1$ and go to Step 6.

Step 5 (A valid point is found) Let $d_c := T_1 + |y_1' - \xi_1 - r_{1,1}x_1|^2$, save $\hat{\mathbf{x}} := \mathbf{x}$. Then, let $i := i + 1$ and go to Step 6.

Step 6 (Schnorr–Euchner enumeration of level i) Let $x_i := x_i + \Delta_i$, $\Delta_i := -\Delta_i - \text{sign}(\Delta_i)$, and go to Step 3.

□

Here T_{i-1} corresponds to the squared distance between the points $(x_i, \ldots, x_n)$ and $(y_i, \ldots, y_n)$. Note that only the coordinates $i, \ldots, n$ of the lattice points are considered. The variable ξ_i, for $i = n, \ldots, 1$, is the decision feedback of a ZF-DFE when the decisions on the symbols from $i + 1$ to n are the current values of $(x_{i+1}, \ldots, x_n)$. The variable Δ_i holds the information of which value of x_i follows next according to the Schnorr–Euchner enumeration.

Every time a new value for x_i is chosen, it corresponds to a new node in the tree representation of the algorithm – see the Example 4.1 and Figure 6 below. The first step is initialization and we start from the root of the tree by choosing a value for the coordinate x_n in step 2. We come to step 2 also every time we move to a lower level i in the tree. Here the midpoint of the interval is chosen as the first value of x_i, i.e., we choose the value of x_i with which the distance from $\mathbf{y}$ is the smallest when the previous values of x_j, $j = i+1, \ldots, n$ are known. In step 3 we calculate the distance of the point from $\mathbf{y}$ and compare it to the radius of the sphere. If we are outside the sphere, we continue to step 4. In this case we go up one level in the tree to the next value of x_{i+1} which is determined by Δ_{i+1}. In case we are at the upper most level of the tree, the search is terminated. If in step 3 we are inside the sphere but outside the signal set boundaries, we go to step 6 and move to the next value of x_i staying at the same level in the tree. If we are both inside the sphere and inside the signal set boundaries, a suitable value for x_i was found and we move down one level in the tree and go to step 2 again. We come to step 5 if a lattice point is found inside the sphere. The radius is then updated and we continue by moving up one level to the node determined by Δ_{i+1}. Step 6 takes care of the zig-zag order of the values of x_i determined by the Schnorr–Euchner enumeration by adjusting the variable Δ_i.

Figure 5 on the next page presents a flow chart of Algorithm II.

Let us next consider a numerical example:

Example 4.1. Let $\mathbf{x} = (2, 7, 3, 2)^T$ be the message to be transmitted (from the signal set $x_i \in \mathbb{Z}_8$) and

$$\mathbf{B} = \begin{pmatrix} 3.7 & 1.6 & -0.6 & 20.2 \\ 1.5 & 5.2 & 5.1 & 13.5 \\ 6.7 & 9.4 & 8.6 & -21.9 \\ -4.3 & -20.7 & -23.1 & -10.7 \end{pmatrix} \text{ and } \mathbf{N} = \begin{pmatrix} -0.8 \\ 0.1 \\ 1.2 \\ -0.9 \end{pmatrix}$$

be the channel and noise matrices. This corresponds to the case of 4 transmit and 4 receive antennas and block length $l = 1$. For simplicity we use real numbers in this example instead of complex numbers. The receiver obtains the signal

$$\mathbf{y} = \mathbf{B}\mathbf{x} + \mathbf{N} = \begin{pmatrix} 56.4 \\ 81.8 \\ 62.4 \\ -245.1 \end{pmatrix}.$$

The problem is now to determine the transmitted signal $\mathbf{x}$, when $\mathbf{y}$ and $\mathbf{B}$ are known. This corresponds to the problem of finding the closest lattice point to the point $\mathbf{y}$ in the lattice $\{\mathbf{B}\mathbf{x} | \mathbf{x} \in \mathbb{Z}_8^4\}$. For this purpose, the QR-decomposition

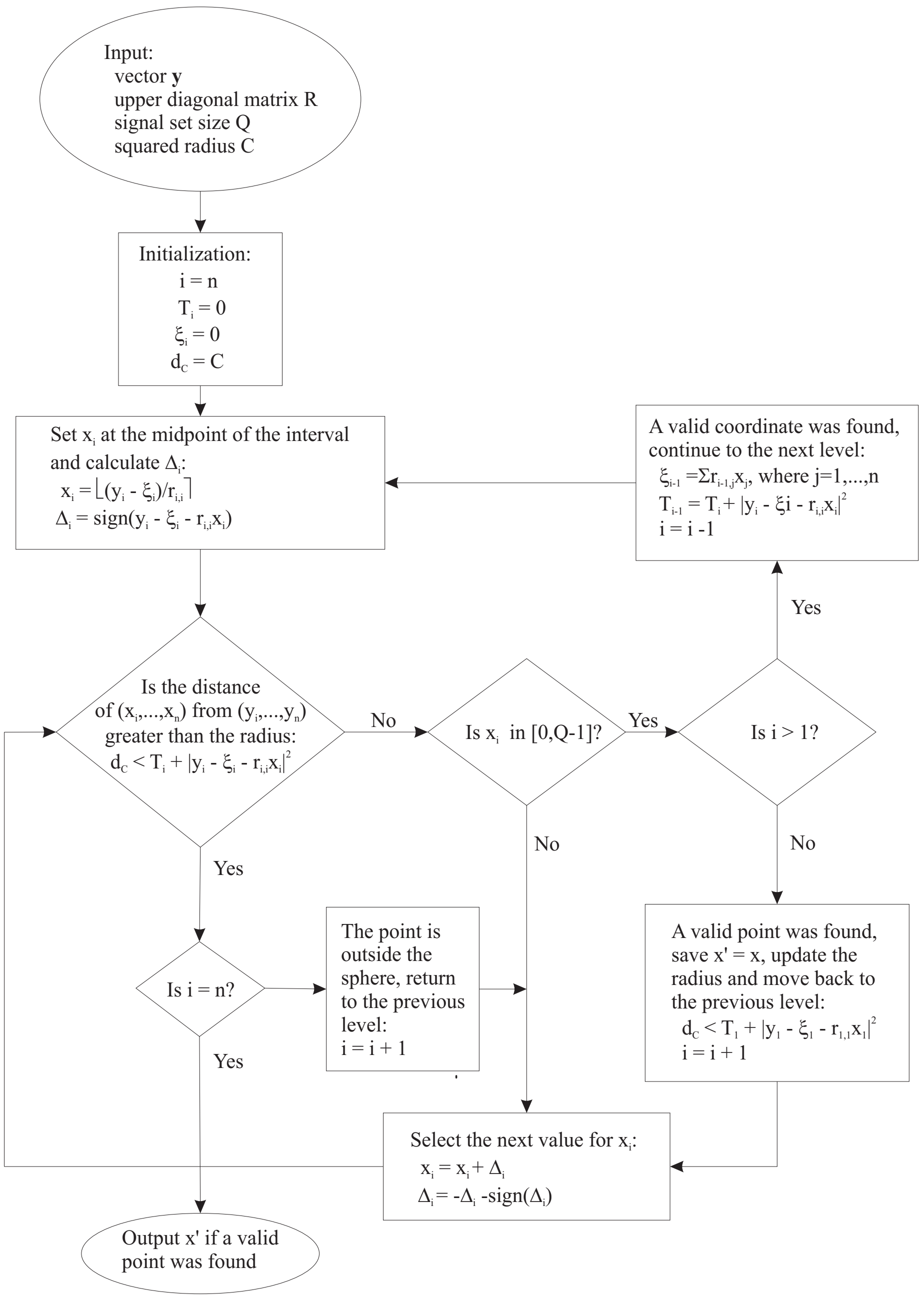

Input:
vector **y**
upper diagonal matrix R
signal set size Q
squared radius C
Initialization:
$i = n$
$T_i = 0$
$\xi_i = 0$
$d_C = C$
Set x_i at the midpoint of the interval and calculate Δ_i:
$x_i = \lfloor (y_i - \xi_i)/r_{i,i} \rceil$
$\Delta_i = \text{sign}(y_i - \xi_i - r_{i,i}x_i)$
A valid coordinate was found, continue to the next level:
$\xi_{i-1} = \Sigma r_{i-1,j}x_j$, where j=1,...,n
$T_{i-1} = T_i + |y_i - \xi i - r_{i,i}x_i|^2$
$i = i - 1$
Yes
Is the distance of $(x_i,...,x_n)$ from $(y_i,...,y_n)$ greater than the radius:
$d_C < T_i + |y_i - \xi_i - r_{i,i}x_i|^2$
No
Is x_i in [0,Q-1]?
Yes
Is i > 1?
No
No
Yes
Is i = n?
The point is outside the sphere, return to the previous level:
$i = i + 1$
A valid point was found, save x' = x, update the radius and move back to the previous level:
$d_C < T_1 + |y_1 - \xi_1 - r_{1,1}x_1|^2$
$i = i + 1$
Yes
Select the next value for x_i:
$x_i = x_i + \Delta_i$
$\Delta_i = -\Delta_i - \text{sign}(\Delta_i)$
Output x' if a valid point was found

Figure 5: Flow chart of Algorithm II

of **B** is first computed:

$$\begin{aligned}\mathbf{B} &= \mathbf{QR}\\ &= \begin{pmatrix} 0.42 & -0.43 & -0.68 & 0.43 \\ 0.17 & 0.15 & -0.51 & -0.83 \\ 0.75 & -0.32 & 0.53 & -0.23 \\ -0.48 & -0.83 & 0.06 & -0.28 \end{pmatrix} \begin{pmatrix} 8.91 & 18.61 & 18.23 & -0.64 \\ 0 & 14.15 & 17.34 & 9.19 \\ 0 & 0 & 0.98 & -32.72 \\ 0 & 0 & 0 & 5.38 \end{pmatrix}.\end{aligned}$$

The sphere decoding algorithm takes the upper triangular matrix **R** and the vector $\mathbf{y}' = \mathbf{Q}^T\mathbf{y} \begin{pmatrix} 202.5 \\ 170.2 \\ -61.4 \\ 10.3 \end{pmatrix}$ as input. The squared radius of the sphere is set to be $C_0 = 15$. It then proceeds as depicted in Figure 6 below. The nodes in the tree correspond to the different values of x_i that the algorithm goes through when searching for the closest lattice point, and the dashed arrows show how the sphere decoder traverses through the tree.

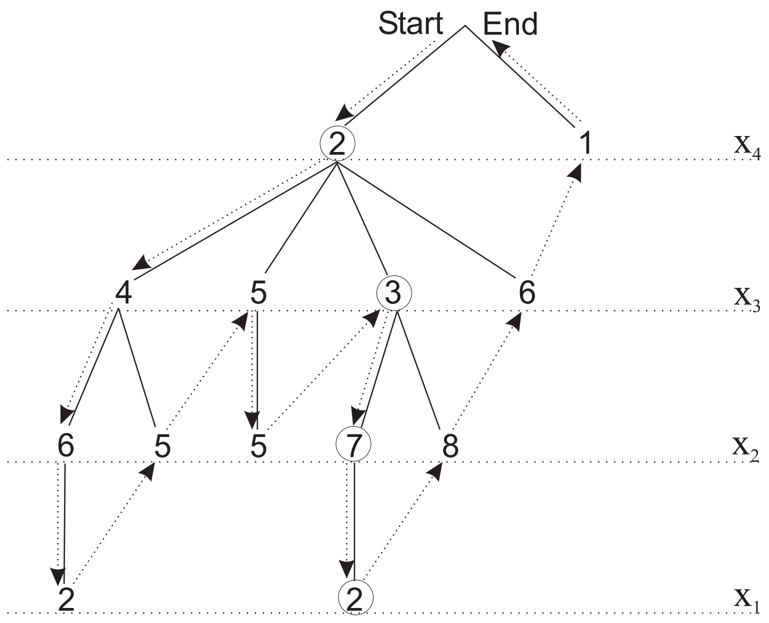


Figure 6: The search tree for the numerical example

The first value for x_4 is computed by $x_4 = \lfloor y'_4/r_{4,4} \rceil = \lfloor 10.3/5.38 \rceil = \lfloor 1.91 \rceil = 2$. This point is inside the sphere, which can be verified using the condition (3), and the algorithm moves to the next level. The same happens at the second and third levels after which a lattice point is found inside the sphere with coordinates $\mathbf{x} = (2, 6, 4, 2)^T$. This point is also referred to as the Babai point. Now the radius of the sphere is replaced by the distance between this point and $\mathbf{y}'$ in the lattice, since the closest point cannot be further away from $\mathbf{y}'$ than the one that was already found. The new value of the squared radius C_0 is 8.25.

Next, the algorithm jumps to the previous level with $x_2 = 5$. However, all of the points $\mathbf{x} = (x_1, 5, 4, 2)^T$ are outside the sphere and the algorithm moves up again to $x_3 = 5$. This is inside, but at the next level the points $\mathbf{x} = (x_1, 5, 5, 2)^T$ are again outside the sphere. The next value for x_3 is 3 and the algorithm proceeds all the way to the level x_1, where another lattice point is found at $\mathbf{x} = (2, 7, 3, 2)^T$. The radius of the sphere is again updated to 2.9, and the algorithm moves up in the tree. All of the following points at each level are then outside the sphere and the algorithm moves up and finally stops. The point which was found to be the closest one to $\mathbf{y}'$ is $\hat{\mathbf{x}} = (2, 7, 3, 2)^T$, which is exactly the transmitted point.

5 Complexity of Sphere Decoding

In e.g. [1], [6], [9], [10], [13] and [14] several factors that may affect the complexity of the sphere decoder have been suggested. In this and the next chapter the complexity of sphere decoding is studied using results from computer simulations. The complexity comparisons are necessarily statistical in nature. In most cases the average complexity is relevant (where the complexity is averaged over several channel realizations and several choices of a transmitted lattice point). In some cases the high complexity tail is more informative.

In the simulations the two block codes from Section 3.1 are used as codebooks. The number of nodes in the search tree is used as a measure of complexity, so that the implementation details or the environment where the simulation runs do not affect it.

5.1 System Model

In this section we describe the system model that was used in the simulations. The system takes the SNR, the numbers of transmit and receive antennas m and n, the initial squared radius C_0 for the sphere decoder, and the size of the signal set q as input parameters. The basis matrices $\mathbf{X}_1, ..., \mathbf{X}_k$, where $\mathbf{X}_i \in \mathcal{M}_{m\times l}(\mathbb{C})$, of the code to be used for transmitting are read from a file that is also given as an input parameter.

The transmission is simulated by first generating a random message $\mathbf{a} = (a_1, ..., a_k)$ to be sent, where the components are from the PAM signal set $a_i = \{2c_i - q + 1 | c_i \in \mathbb{Z}_q\}$. Then the channel matrix $\mathbf{H} \in \mathcal{M}_{n\times m}(\mathbb{C})$ and the noise matrix $\mathbf{N} \in \mathcal{M}_{n\times l}(\mathbb{C})$ are generated. The elements $\eta_{i,j}$ of $\mathbf{N}$ are i.i.d. complex Gaussian random variables with zero mean and variance $\sigma^2 = 1$ per dimension. The elements $h_{i,j}$ of $\mathbf{H}$ are also i.i.d. complex Gaussian random variables with zero mean and variance $\tau^2 = SNR\frac{l}{P_{\mathcal{L}}}$ per dimension, where $P_{\mathcal{L}}$ is the average energy of a codeword from the used constellation. The received matrix $\mathbf{Y} \in \mathcal{M}_{n\times l}(\mathbb{C})$ is computed by

$$\mathbf{Y} = \mathbf{H}\sum_{i=1}^{k} a_i \mathbf{X}_i + \mathbf{N}.$$

Next the system simulates the receiver by finding the closest point to the received one using knowledge of the channel state. A translation and scaling is applied to the received matrix so that instead of the coefficients a_i we have the coefficients $c_i \in \mathbb{Z}_q$ in order to use the sphere decoder. The matrix $\mathbf{Y}$ is then turned into a real valued vector $\mathbf{y}' \in \mathbb{R}^{2ln}$ by converting each row $\mathbf{y}_i$ to real valued vector $\mathbf{y}'_i$ and then combining these together:

$$\mathbf{y}'_i = (\Re(y_{i,1}), \Im(y_{i,1}), \Re(y_{i,2}), \Im(y_{i,2}), \ldots, \Re(y_{i,l}), \Im(y_{i,l})) \qquad \text{for } i = 1, ..., n$$

and

$$\mathbf{y}' = \begin{bmatrix} \mathbf{y}'_1 & \mathbf{y}'_2 & \cdots & \mathbf{y}'_n \end{bmatrix}.$$

The matrices $\mathbf{HX}_1, ..., \mathbf{HX}_k \in \mathcal{M}_{n\times l}(\mathbb{C})$ are computed and turned into vectors $\mathbf{b}_i \in \mathbb{R}^{2ln}$ in the same way as the received matrix. Then the QR-decomposition is computed for the matrix $\mathbf{B} = \begin{bmatrix} \mathbf{b}_1^T & \mathbf{b}_2^T & \cdots & \mathbf{b}_k^T \end{bmatrix}$ and $\mathbf{R}$, $\mathbf{y}'$ and the initial squared radius C_0 are given to the sphere decoding algorithm as input. The sphere decoder then returns the closest point $\mathbf{x}$, from which the transmitted point $\mathbf{a}$ can be obtained by computing $a_i = 2x_i - q + 1$ for $i = 1, ..., k$.

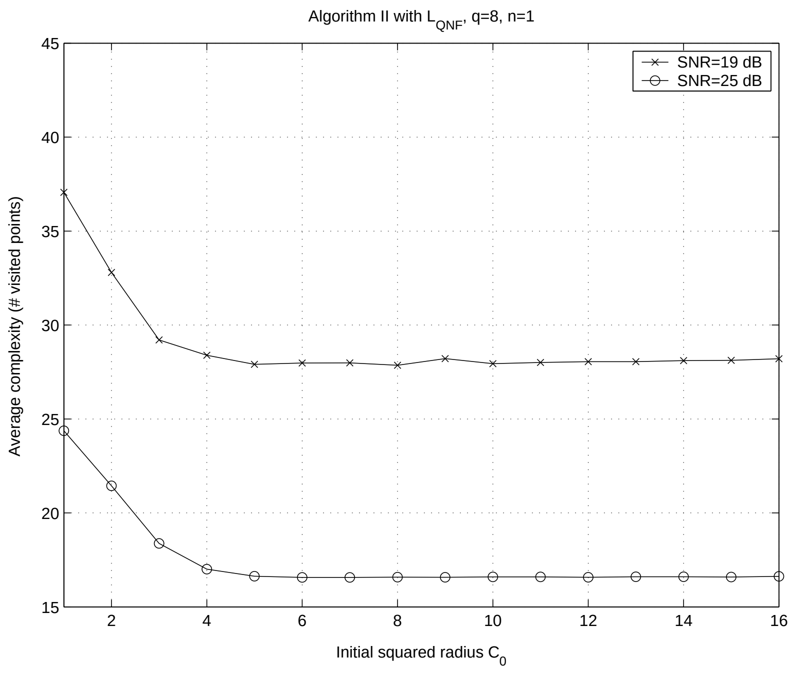
The system was implemented in C++ and the simulations were run on a Windows XP Professional environment with a 1.83 GHz dual-core processor and 2048MB RAM. In each simulation at least 1 000 000 channel realizations were created, unless otherwise stated. In the cases where the performances of different algorithms were being compared, the same channel realizations were used for both algorithms.

5.2 Choosing the Radius

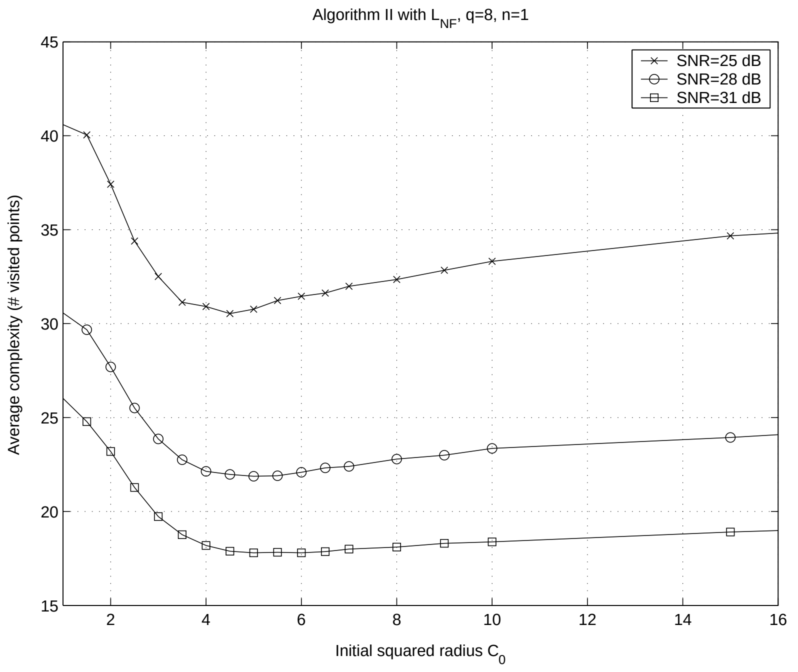
The selection of the initial radius affects the performance of the sphere decoding algorithm. Too small an initial radius causes no lattice points to be found inside the sphere and the sphere decoder must start the search again with an increased radius. On the other hand, too large a radius may result in an unnecessarily large search tree.

Algorithm II is in general not very sensitive to a large initial radius. This is because in most cases it finds the Babai point right away and updates the radius accordingly so that the large initial radius is not a problem. However, with finite lattices the Babai point may be outside the constellation, in which case the algorithm does not find it and may have to search for a long time before finding a valid point inside the sphere.

Figures 7(a) and 7(b) show the average complexity of Algorithm II as a function of C_0 in systems using the lattice codes L_{QNF} and L_{NF}. We observe that with L_{QNF} after a certain point a larger initial radius does not have much of an effect on the complexity. A system using the code L_{NF} on the contrary is much more sensitive to the selection of the initial radius, especially in the low SNR range.



(a) The code L_{QNF} is used for transmission



(b) The code L_{NF} is used for transmission

Figure 7: The average complexity of Algorithm II as a function of the initial squared radius C_0 in a system with one receive antenna and $q = 8$.

5.3 SNR and Complexity

A high SNR typically reduces the complexity of decoding. When the SNR increases, the influence of the noise gets smaller and the received point is expected to be nearer to a lattice point. It also means that the Babai point is more often the closest point. As the Babai point is the first lattice point that the sphere decoder finds, the algorithm finds the solution very quickly. Figure 8 shows the average complexity plotted against SNR for the code L_{NF}. It can be seen that as the SNR increases, the average complexity tends to 15. With an 8-level search tree, 15 is the minimum number points the algorithm needs to visit. It means that the sphere decoder goes down the search tree, finds a lattice point and then comes straight up without any zigzagging. We also notice, that the sphere decoder performs better with the lattice code L_{QNF} than with L_{NF}. We study the reasons for this in Chapter 6.

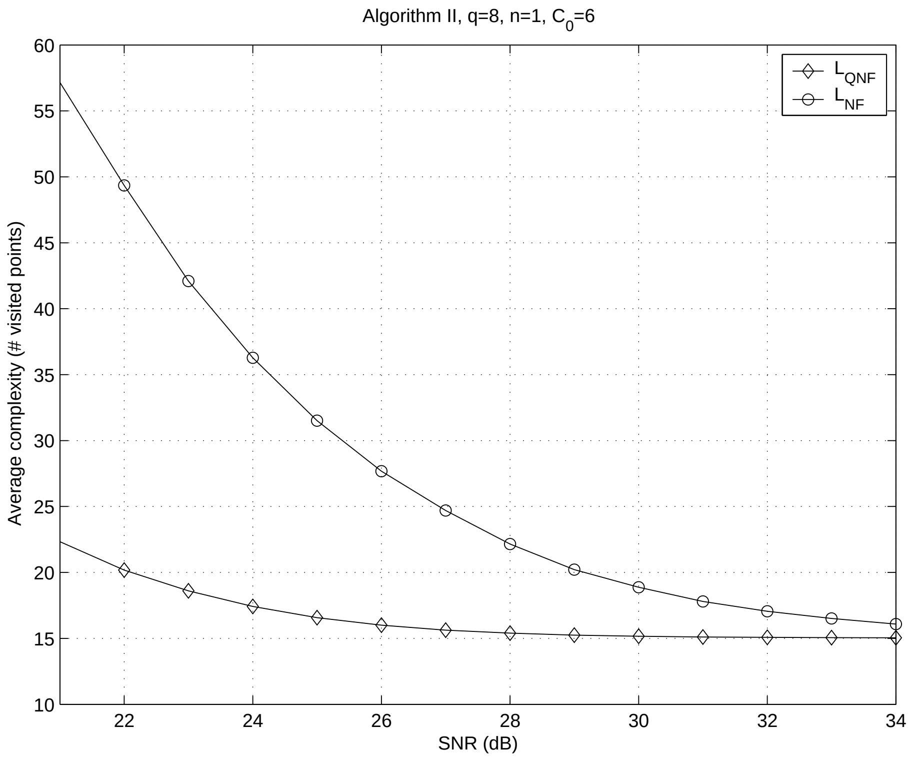


Figure 8: The average complexity of Algorithm II as a function of the SNR, using the codes L_{QNF} and L_{NF}

The complexity depends also on the data rate, i.e., the choice of q. A greater size of the signal set means that the codebook is larger and also the search tree will be larger, since on each level of the tree there will be more valid options for the coefficients. This effect is visible in Figure 9, which presents the average complexity against SNR when different data rates are used.

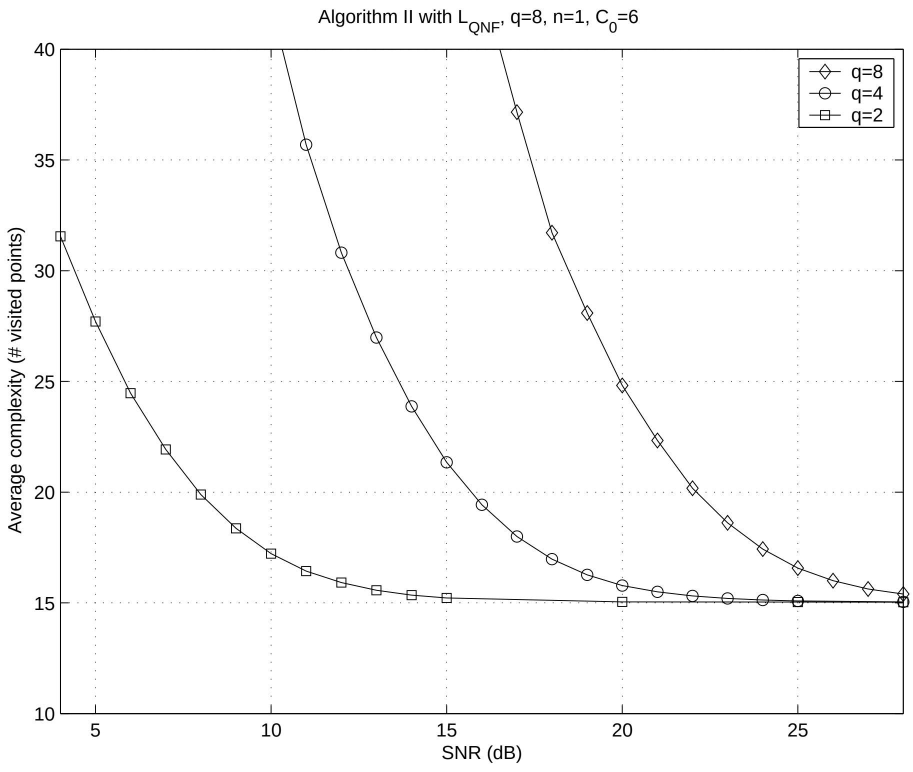


Figure 9: The average complexity of Algorithm II as a function of the SNR, using the code L_{QNF} and different data rates in the transmission

5.4 Preprocessing and Ordering

Preprocessing and the ordering in which the components of $\mathbf{x}$ are considered has a significant effect on the performance of the sphere decoder. The standard preprocessing and ordering consists of the QR-decomposition and the natural back-substitution ordering $x_k, x_{k-1}, \ldots, x_1$. But this is not necessarily the optimal ordering.

One example of a preprocessing approach is the vertical Bell Labs layered space–time (V-BLAST) optimal decision ordering [7]. Its purpose is to maximize the minimum value of the R-matrix's diagonal elements, since a large value of $r_{i,i}$ gives a short search interval on the corresponding level of the search tree. Ordering the diagonal elements so that $r_{k,k}$ is the largest and $r_{1,1}$ the smallest value also reduces the effect of error propagation. Since the sphere decoder starts from the level k, selecting wrong values in the beginning can have an adverse effect on the complexity.

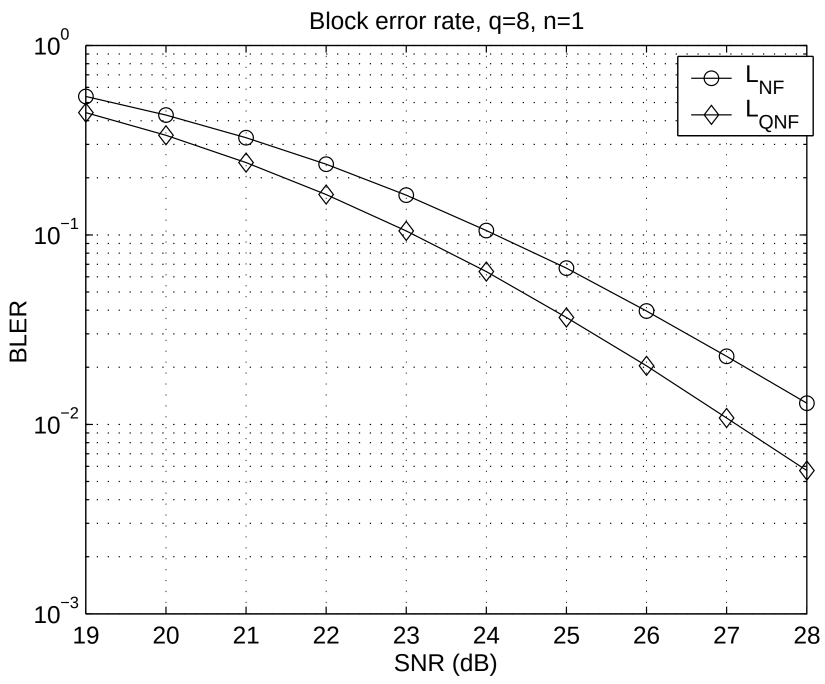
The lattice code L_{QNF} does not need this kind of preprocessing, because as we will see in Section 7.2, the default ordering gives its R-matrix an advantageous structure, which allows for the use of a faster, modified algorithm.

5.5 Error Probability

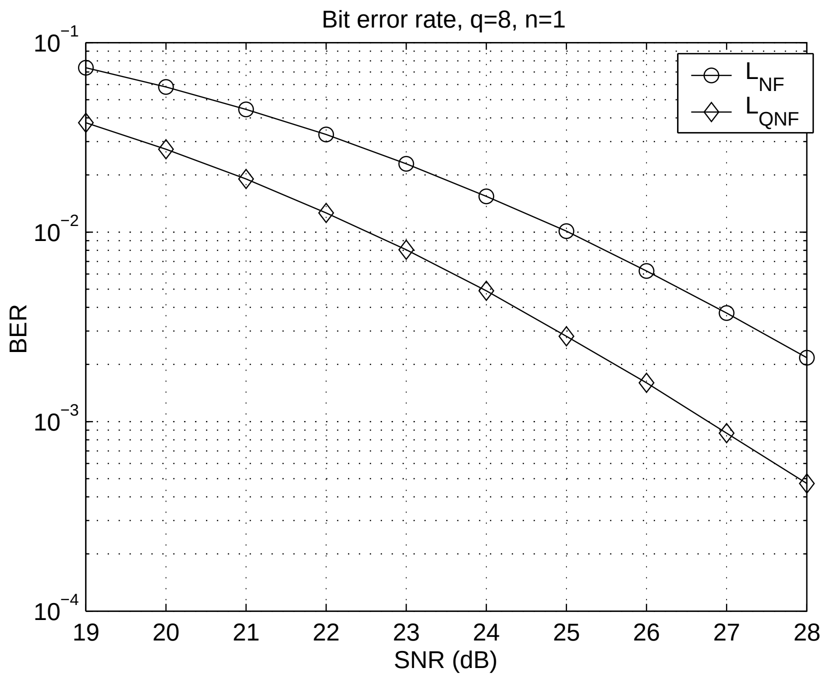
Since the sphere decoder finds the exact maximum-likelihood solution, the error rates show exactly how often the closest lattice point detected at the receiver has not been the point that was sent. The block error rate (BLER) and the bit error rate (BER) indicate how often there have been symbol errors and correspondingly bit errors in the detected message. In the simulations Gray encoding has been used for the bit encoding.

The error rates naturally decrease as the SNR increases and the effect of noise becomes smaller. Figures 10(a) and 10(b) display a comparison of the block and bit error rates for the codes L_{QNF} and L_{NF} with different values of SNR. It can be seen that the code L_{QNF} performs better than the code L_{NF} in the sense of error probability.

Other effects are more subtle. Some lattices are more adversely affected by certain anomalous channel realizations than others. In such cases the lattice is severely skewed and high complexity comes jointly with high error probability. This scenario will be explained in more detail in Chapter 6.



(a) Block error rates for L_{QNF} and L_{NF}



(b) Bit error rates for L_{QNF} and L_{NF}

Figure 10: Error rates for L_{QNF} and L_{NF}

6 Collapsing Lattices

In this section only the multiple-input single-output (MISO) channel setting with a single receive antenna is considered. In the MISO case the channel matrix $\mathbf{H}$ is a complex vector, which we here denote by $\mathbf{h}$.

6.1 The Defect

Several different 8-dimensional lattice codes of 4×4 complex matrices have been designed, but they all have the property that for some non-zero channel vector $\mathbf{h}$ the dimension of the lattice $\mathbf{h}L$ is smaller than the dimension of L. In the following we call this event the collapsing of the lattice. The closely related notion of a lattice's defect was presented in the article [10]:

Definition 6.1. Matrix lattice L has *defect* r, if its rank is m but the minimum positive real dimension of the span of $\mathbf{h}L$ is $m - r$. In other words, the lattice collapses by dimension r.

A lattice $\mathbf{h}L$ collapses if and only if there exists a vector $\mathbf{h} \in \mathbb{C}^n \setminus \{\mathbf{0}\}$ such that the set $\{\mathbf{hX}_1, \ldots, \mathbf{hX}_k\}$ is linearly dependent over $\mathbb{R}$, where $\mathbf{X}_1, \ldots, \mathbf{X}_k$ are the basis matrices of L.

Let us consider next the lattice code L_{NF} that was introduced in Section 3.1. The code matrices are of the form

$$\mathbf{M}_{NF}(c_1, c_2, c_3, c_4) = \begin{pmatrix} c_1 & ic_4 & ic_3 & ic_2 \\ c_2 & c_1 & ic_4 & ic_3 \\ c_3 & c_2 & c_1 & ic_4 \\ c_4 & c_3 & c_2 & c_1 \end{pmatrix} \qquad c_1, c_2, c_3, c_4 \in \mathbb{Z}[i].$$

Let the basis matrices $\mathbf{X}_1, \ldots, \mathbf{X}_8$ be as defined in Section 3.1. Now $L_{NF} = \mathbb{Z}\mathbf{X}_1 \oplus \mathbb{Z}\mathbf{X}_2 \oplus \ldots \oplus \mathbb{Z}\mathbf{X}_8$ and it has dimension 8.

Let us define $\mathbf{h}_j = (1, \zeta^j, \zeta^{2j}, \zeta^{3j})$ for $j \in \{1, 5, 9, 13\}$, where $\zeta = e^{2\pi i/16}$. The vectors $\mathbf{h}_j$ are the eigenvectors of all of the matrices of the lattice L_{NF}, as shown below. Note that $\zeta^4 = i$ and $\zeta^{4j} = i^j = i$.

$$
\begin{aligned}
\mathbf{h}_j\mathbf{M}_{NF}(c_1,c_2,c_3,c_4) &= (1,\zeta^j,\,\zeta^{2j},\,\zeta^{3j})\begin{pmatrix} c_1 & ic_4 & ic_3 & ic_2 \\ c_2 & c_1 & ic_4 & ic_3 \\ c_3 & c_2 & c_1 & ic_4 \\ c_4 & c_3 & c_2 & c_1 \end{pmatrix} \\
&= (\underbrace{c_1+\zeta^j c_2+\zeta^{2j}c_3+\zeta^{3j}c_4}_{=\lambda_j},\,\ldots,\,ic_2+\zeta^j ic_3+\zeta^{2j}ic_4+\zeta^{3j}c_1) \\
&= (\lambda_j,\,\ldots,\zeta^{4j}c_2+\zeta^{5j}c_3+\zeta^{6j}c_4+\zeta^{3j}c_1) \\
&= (\lambda_j,\,\ldots,\zeta^{3j}(c_1+\zeta^j c_2+\zeta^{2j}c_3+\zeta^{3j}c_4)) \\
&= (\lambda_j,\,\zeta^j\lambda_j,\,\zeta^{2j}\lambda_j,\,\zeta^{3j}\lambda_j) \\
&= \lambda_j\,(1,\,\zeta^j,\,\zeta^{2j},\,\zeta^{3j}) \\
&= \lambda_j\mathbf{h}_j
\end{aligned}
$$

The vectors $\{\mathbf{h}_1,\,\mathbf{h}_5,\,\mathbf{h}_9,\,\mathbf{h}_{13}\}$ form a basis of $\mathbb{C}^4$. The channel vector $\mathbf{h}$ can be written in terms of the basis vectors: $\mathbf{h}=\alpha_1\mathbf{h}_1+\alpha_5\mathbf{h}_5+\alpha_9\mathbf{h}_9+\alpha_{13}\mathbf{h}_{13}$, where $\alpha_j\in\mathbb{C}$. Now, suppose that the channel vector $\mathbf{h}$ is a scalar multiple of only one of the $\mathbf{h}_j$. Then only two of the vectors $\mathbf{hX}_1,\ldots,\mathbf{hX}_8$ are linearly independent over $\mathbb{R}$, and the lattice $\mathbf{h}L_{NF}$ has real dimension 2. Thus, the lattice L_{NF} has defect 6.

Theorem 6.1. *With the lattice code L_{NF} the image lattice $\mathbf{h}L_{NF}$ collapses $\Leftrightarrow\alpha_j=0$ for some $j\in\{1,5,9,13\}$.*

Proof. The basis vectors of the lattice $\mathbf{h}L_{NF}$ are

$$
\begin{aligned}
\mathbf{hX}_k &= \sum_j \alpha_j\mathbf{h}_j\mathbf{X}_k \\
&= \sum_j \alpha_j\lambda(\mathbf{X}_k,j)\mathbf{h}_j
\end{aligned}
$$

where $\lambda(\mathbf{X}_k,j)$ is the scalar in $\mathbf{h}_j\mathbf{X}_k=\lambda(\mathbf{X}_k,j)\mathbf{h}_j$. If $\alpha_j=0$ for some j, say $\alpha_9=0$, then $\forall k=1,\ldots,8$

$$
\begin{aligned}
\mathbf{hX}_k &= \beta_1\mathbf{h}_1+\beta_5\mathbf{h}_5+\beta_{13}\mathbf{h}_{13} \\
&\in \mathbb{C}\mathbf{h}_1+\mathbb{C}\mathbf{h}_5+\mathbb{C}\mathbf{h}_{13}\triangleq V_{\hat{9}}
\end{aligned}
$$

where $\dim_{\mathbb{C}}V_{\hat{9}}=3$ and $\dim_{\mathbb{R}}V_{\hat{9}}=6$.

Conversely, assume that $\alpha_j\neq 0\;\forall j\in\{1,5,9,13\}$. Since $\mathbf{hX}_{2k}=i\mathbf{hX}_{2k-1}$ for $k=1,\ldots,4$, it suffices to check that the vectors $\mathbf{hX}_{2k-1}$, $k=1,\ldots,4$ are linearly independent. For them $\lambda(\mathbf{X}_{2k-1},j)=\zeta^{(k-1)j}$.

$$
\mathbf{hX}_{2k-1}=\sum_{j\in\{1,5,9,13\}}\alpha_j\zeta^{(k-1)j}\mathbf{h}_j,\qquad k=1,\ldots,4.
$$

The determinant

$$\begin{vmatrix} \alpha_1 & \alpha_1\zeta & \alpha_1\zeta^2 & \alpha_1\zeta^3 \\ \alpha_5 & \alpha_5\zeta^5 & \alpha_5\zeta^{10} & \alpha_5\zeta^{15} \\ \alpha_9 & \alpha_9\zeta^9 & \alpha_9\zeta^{18} & \alpha_9\zeta^{27} \\ \alpha_{13} & \alpha_{13}\zeta^{13} & \alpha_{13}\zeta^{26} & \alpha_{13}\zeta^{39} \end{vmatrix} = \alpha_1\alpha_5\alpha_9\alpha_{13}|VM| \neq 0$$

where $|VM|$ is the determinant of a Van der Monde matrix, so the vectors $\mathbf{hX}_{2k-1}$, $k = 1, \ldots, 4$, are linearly independent. □

Let us consider now the code L_{QNF}, which was also introduced in Section 3.1. The code matrices are of the form

$$\mathbf{M}_{QNF}(c_1, c_2, c_3, c_4) = \begin{pmatrix} c_1 & ic_2 & -c_3^* & -c_4^* \\ c_2 & c_1 & ic_4^* & -c_3^* \\ c_3 & ic_4 & c_1^* & c_2^* \\ c_4 & c_3 & -ic_2^* & c_1^* \end{pmatrix} \qquad c_1, c_2, c_3, c_4 \in \mathbb{Z}[i].$$

The channel vector $\mathbf{h}$ can be written in terms of vectors $\mathbf{v}_i$, $(i = 1, \ldots, 4)$, which form a basis of $\mathbb{C}^4$:

$$\mathbf{h} = \mathbf{h}_+ + \mathbf{h}_- = \underbrace{h_1\mathbf{v}_1 + h_2\mathbf{v}_2}_{\mathbf{h}_+} + \underbrace{h_3\mathbf{v}_3 + h_4\mathbf{v}_4}_{\mathbf{h}_-}$$

where $\mathbf{v}_1 = (1, \xi, 0, 0)$, $\mathbf{v}_2 = (0, 0, 1, \xi)$, $\mathbf{v}_3 = (1, -\xi, 0, 0)$, $\mathbf{v}_4 = (0, 0, 1, -\xi)$, $h_j \in \mathbb{C}$ $(j = 1, \ldots 4)$ and $\xi = e^{2\pi i/8}$. We denote by $V_+ = L_{\mathbb{C}}(\mathbf{v}_1, \mathbf{v}_2)$ the vector space generated by $\mathbf{v}_1$ and $\mathbf{v}_2$ and by $V_- = L_{\mathbb{C}}(\mathbf{v}_3, \mathbf{v}_4)$ the vector space generated by $\mathbf{v}_3$ and $\mathbf{v}_4$.

Now

$$\begin{aligned} \mathbf{v}_1\mathbf{M}_{QNF}(c_1, c_2, c_3, c_4) &= (c_1 + \xi c_2)\mathbf{v}_1 - (c_3 + \xi c_4)^*\mathbf{v}_2, \\ \mathbf{v}_2\mathbf{M}_{QNF}(c_1, c_2, c_3, c_4) &= (c_3 + \xi c_4)\mathbf{v}_1 + (c_1 + \xi c_2)^*\mathbf{v}_2, \\ \mathbf{v}_3\mathbf{M}_{QNF}(c_1, c_2, c_3, c_4) &= (c_1 - \xi c_2)\mathbf{v}_3 - (c_3 - \xi c_4)^*\mathbf{v}_4 \text{ and} \\ \mathbf{v}_4\mathbf{M}_{QNF}(c_1, c_2, c_3, c_4) &= (c_3 - \xi c_4)\mathbf{v}_3 + (c_1 - \xi c_2)^*\mathbf{v}_4. \end{aligned}$$

Theorem 6.2. *With the lattice code L_{QNF} the image lattice $\mathbf{h}L_{QNF}$ collapses $\Leftrightarrow \mathbf{h}_+ = 0$ or $\mathbf{h}_- = 0$*

Proof. Consider the vector space

$$V = \{\mathbf{hM}_{QNF}(z_1, z_2, z_3, z_4) \mid z_i \in \mathbb{C}\}.$$

First we show that if $\mathbf{h}_+ \neq 0$ and $\mathbf{h}_- \neq 0$ then $\dim_{\mathbb{R}} V = 8$.

Now if $z_1 = \xi z_2$ and $z_3 = \xi z_4$ then $\mathbf{v}_3\mathbf{M}_{QNF}(z_1, z_2, z_3, z_4) = 0$ and $\mathbf{v}_4\mathbf{M}_{QNF}(z_1, z_2, z_3, z_4) = 0$, and the dimension of V depends only on $\mathbf{h}_+$ and not on $\mathbf{h}_-$.

Notice that $\xi = e^{2\pi i/8} = \frac{1+i}{\sqrt{2}}$, $\xi^2 = i$ and $\xi^* = -i\xi$.

i) If $z_1 = \frac{1}{2}$, $z_2 = \frac{1}{2}\xi^*$, $z_3 = z_4 = 0$ then $\mathbf{v}_1\mathbf{M}_{QNF} = \mathbf{v}_1$ and $\mathbf{v}_2\mathbf{M}_{QNF} = \mathbf{v}_2$

ii) If $z_1 = \frac{i}{2}$, $z_2 = \frac{1}{2}\xi$, $z_3 = z_4 = 0$ then $\mathbf{v}_1\mathbf{M}_{QNF} = i\mathbf{v}_1$ and $\mathbf{v}_2\mathbf{M}_{QNF} = -i\mathbf{v}_2$

iii) If $z_1 = z_2 = 0$, $z_3 = -\frac{1}{2}, z_4 = -\frac{1}{2}\xi^*$ then $\mathbf{v}_1\mathbf{M}_{QNF} = \mathbf{v}_2$ and $\mathbf{v}_2\mathbf{M}_{QNF} = -\mathbf{v}_1$

iv) If $z_1 = z_2 = 0$, $z_3 = \frac{i}{2}$, $z_4 = \frac{1}{2}\xi$ then $\mathbf{v}_1\mathbf{M}_{QNF} = i\mathbf{v}_2$ and $\mathbf{v}_2\mathbf{M}_{QNF} = i\mathbf{v}_1$

From these we see that the four $\mathbf{h}_+\mathbf{M}_{QNF}$ vectors below are all in V.

i) $\Rightarrow \mathbf{h}_+\mathbf{M}_{QNF} = h_1\mathbf{v}_1 + h_2\mathbf{v}_2 \in V$

ii) $\Rightarrow \mathbf{h}_+\mathbf{M}_{QNF} = ih_1\mathbf{v}_1 - ih_2\mathbf{v}_2 \in V$

iii) $\Rightarrow \mathbf{h}_+\mathbf{M}_{QNF} = -h_2\mathbf{v}_1 + h_1\mathbf{v}_2 \in V$

iv) $\Rightarrow \mathbf{h}_+\mathbf{M}_{QNF} = ih_2\mathbf{v}_1 + ih_1\mathbf{v}_2 \in V$

All of these belong to V_+ and by showing that these four vectors are linearly independent over $\mathbb{R}$ we prove that the whole V_+ is included in V. Let $h_1 = x_1 + y_1 i$ and $h_2 = x_2 + y_2 i$ where $x_j, y_j \in \mathbb{R}$ for $j = 1, 2$. In terms of the $\mathbb{R}$-basis $\{\mathbf{v}_1, i\mathbf{v}_1, \mathbf{v}_2, i\mathbf{v}_2\}$ of V_+, these vectors get the form

$$\begin{aligned}
&(x_1, y_1, x_2, y_2) \in V\\
&(-y_1, x_1, y_2, -x_2) \in V\\
&(-x_2, -y_2, x_1, y_1) \in V\\
&(-y_2, x_2, -y_1, x_1) \in V
\end{aligned}$$

By looking at the determinant of the matrix

$$A = \begin{pmatrix} x_1 & y_1 & x_2 & y_2 \\ -y_1 & x_1 & y_2 & -x_2 \\ -x_2 & -y_2 & x_1 & y_1 \\ -y_2 & x_2 & -y_1 & x_1 \end{pmatrix}$$

we see that if $h_1 \neq 0$ or $h_2 \neq 0$ the vectors are linearly independent. Since $AA^T = (x_1^2 + y_1^2 + x_2^2 + y_2^2)\mathbf{I}_4$ it follows that $\det A \neq 0$ if $h_1 \neq 0$ or $h_2 \neq 0$. Hence, if $\mathbf{h}_+ \neq 0$ then $V_+ \subseteq V$.

Similarly by considering the code matrices for which $z_1 = -\xi z_2$ and $z_3 = -\xi z_4$ we see that if $\mathbf{h}_- \neq 0$ then $V_- \subseteq V$.

Now $V_+ \oplus V_- \subseteq V \subseteq L_{\mathbb{C}}(\mathbf{v}_1, \mathbf{v}_2, \mathbf{v}_3, \mathbf{v}_4)$. Since $V_+ \oplus V_- = L_{\mathbb{C}}(\mathbf{v}_1, \mathbf{v}_2, \mathbf{v}_3, \mathbf{v}_4)$ it follows that $V = L_{\mathbb{C}}(\mathbf{v}_1, \mathbf{v}_2, \mathbf{v}_3, \mathbf{v}_4)$ and hence $\dim_{\mathbb{R}} V = 8$ if $\mathbf{h}_+ \neq 0$ and $\mathbf{h}_- \neq 0$.

Conversely, if $\mathbf{h}_+ = 0$ or $\mathbf{h}_- = 0$ then the resulting vector space $\mathbf{hM}_{QNF}$ only has terms of $\mathbf{v}_1$ and $\mathbf{v}_2$ or terms of $\mathbf{v}_3$ and $\mathbf{v}_4$ and thus has real dimension 4. This means that L_{QNF} has defect 4. □

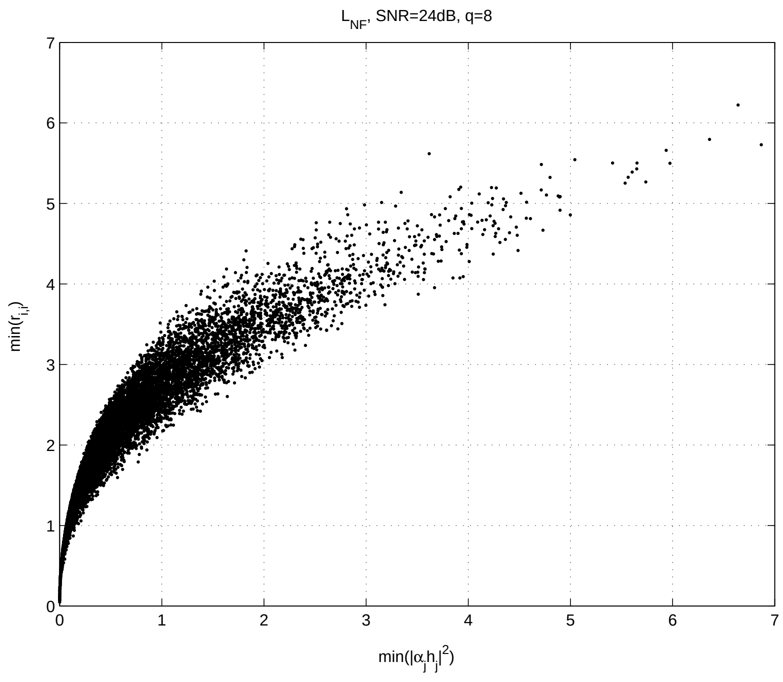
6.2 Collapsing Lattices and the Decoding Complexity

If the vectors $\{\mathbf{hX}_1, \ldots, \mathbf{hX}_k\}$ are orthogonal then the closest lattice point is the Babai point, which the sphere decoder finds right away without any zigzagging. It is natural to assume that the decoding complexity would be significantly higher if the lattice collapses or comes near to collapsing. In this section we study the effect of the collapsing lattice on the decoding performance using computer simulations.

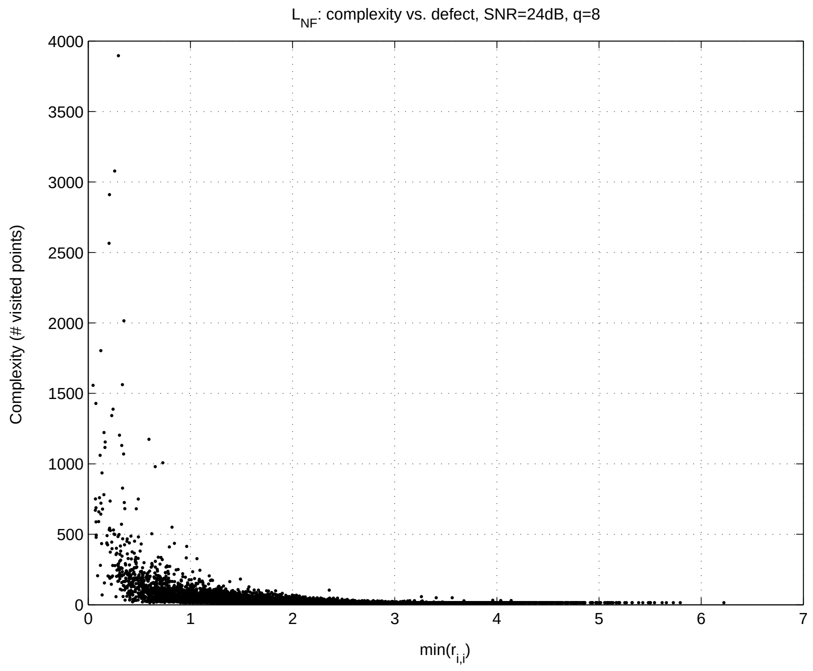
When the lattice collapses or is near collapsing, it causes the diagonal elements of $\mathbf{R}$ to be very small. This can be verified from Figure 11(a), which shows the correlation between the minimum diagonal element of $\mathbf{R}$ and the length of the shortest component $\alpha_j \mathbf{h}_j$ of $\mathbf{h}$ for the lattice code L_{NF}. Figure 13(a) shows the same for the lattice code L_{QNF}, where $\mathbf{h}$ has components $\mathbf{h}_+$ and $\mathbf{h}_-$.

A small diagonal element of $\mathbf{R}$ means that the corresponding interval for the sphere decoder on that level is large, which makes the search tree also large. It can be seen from Figures 11(b) and 13(b), that a small diagonal element very often corresponds to a high complexity.

Figures 12(a) and 12(b) show the complexity distributions of 10,000 transmissions for the lattice code L_{NF} with different SNRs. The horizontal axis indicates how close to collapsing the image lattice $\mathbf{h}L$ has been. Figures 14(a) and 14(b) show the same for the lattice code L_{QNF}. For both L_{NF} and L_{QNF}, the figures show that the smaller the minimum component of $\mathbf{h}$, the higher the complexity. We also notice that the lattice code L_{NF} comes near to collapsing more often than L_{QNF}. This can be explained by the higher defect of L_{NF}.

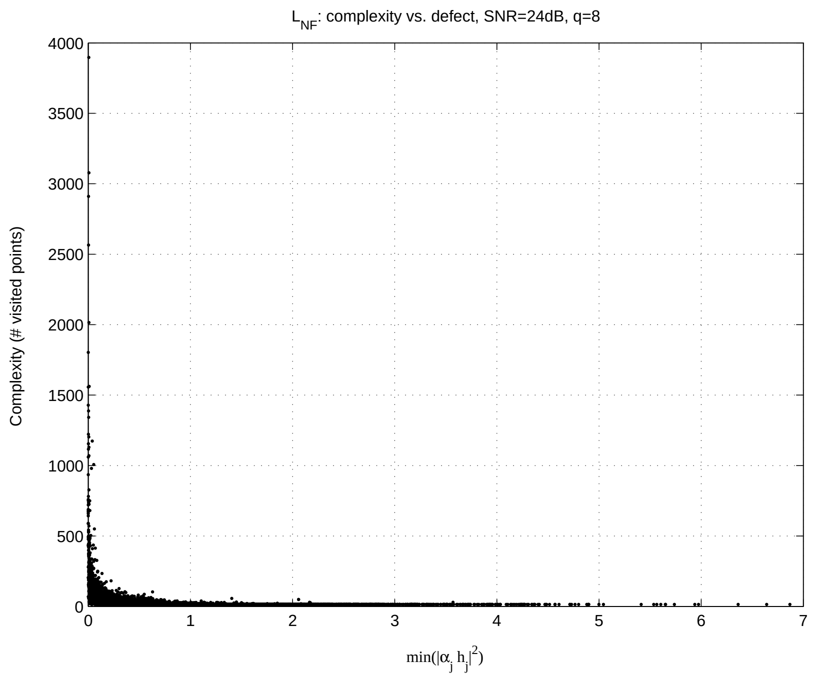


(a) Minimum diagonal element of R vs. minimum $|\alpha_j \mathbf{h}_j|^2$

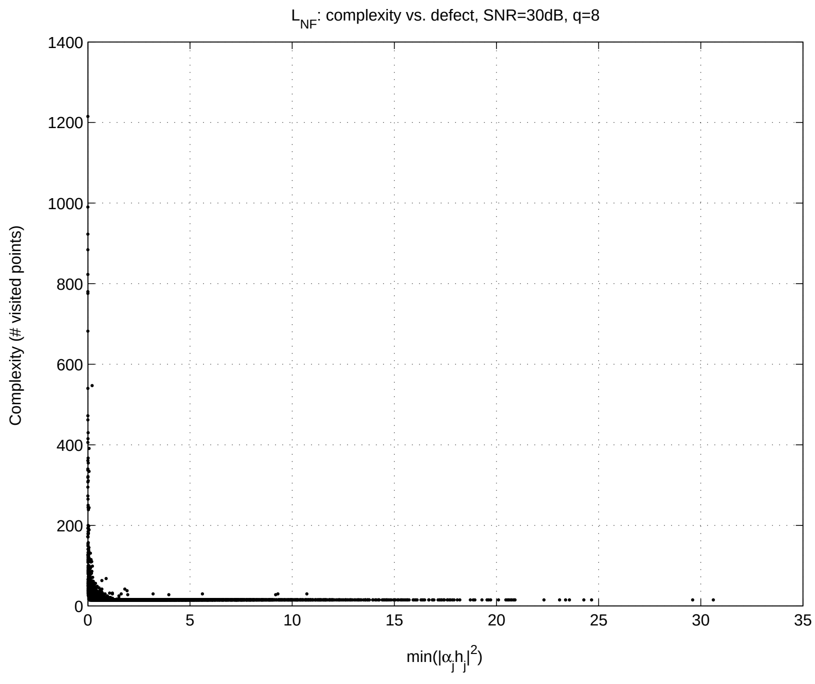


(b) Complexity vs. minimum diagonal element of R

Figure 11: Distributions of 10,000 transmissions using the code L_{NF}

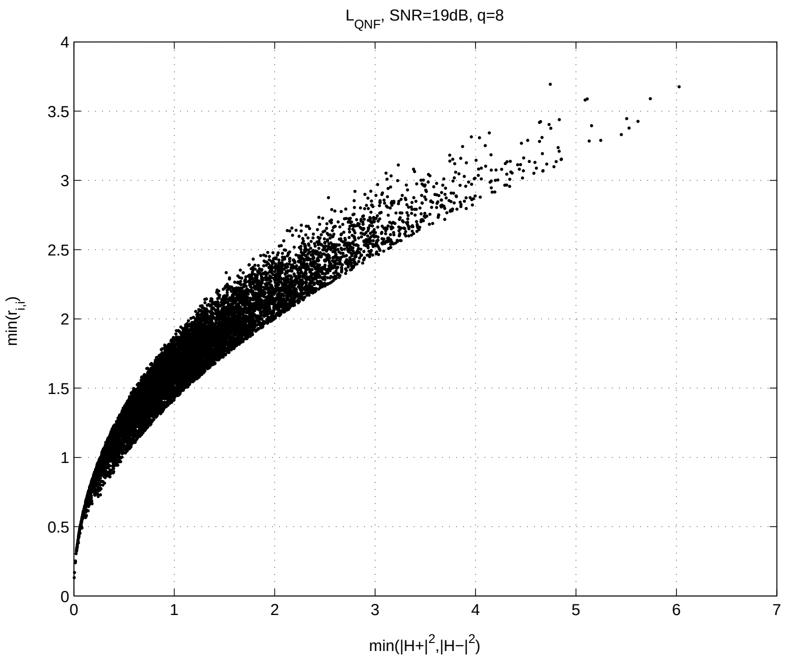


(a) Complexity vs. defect with L_{NF} and an SNR of 24dB

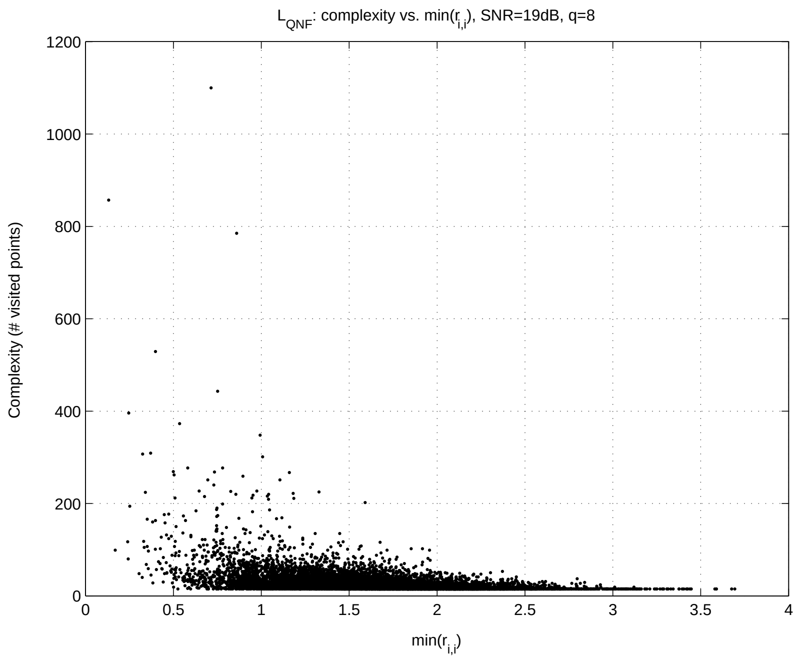


(b) Complexity vs. defect with L_{NF} and an SNR of 30dB

Figure 12: Complexity distributions of 10,000 transmissions using the code L_{NF} and different SNRs

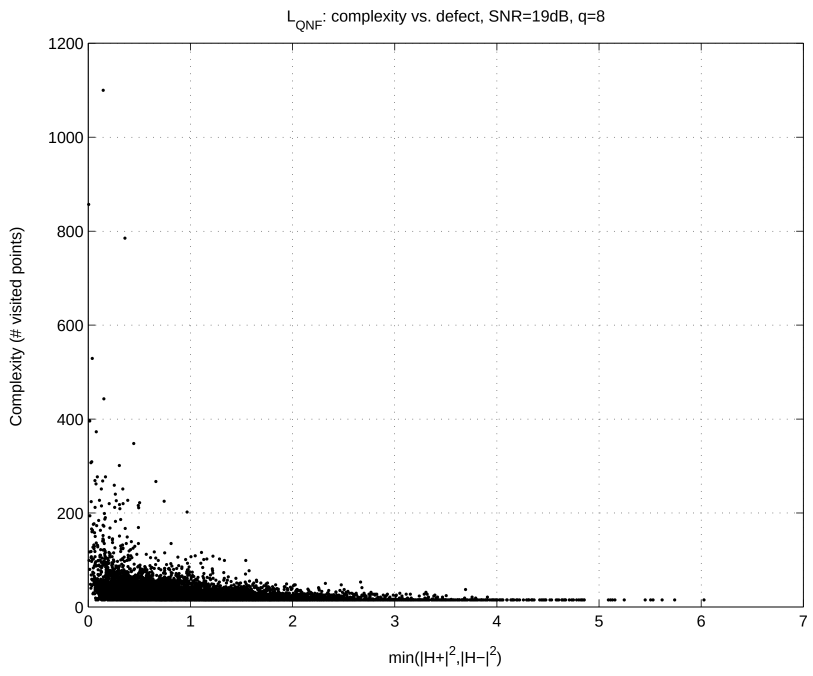


(a) Minimum diagonal element of R vs. $\min\{|\mathbf{h}_+|^2, \mathbf{h}_-|^2\}$ with L_{QNF} and an SNR of 19dB

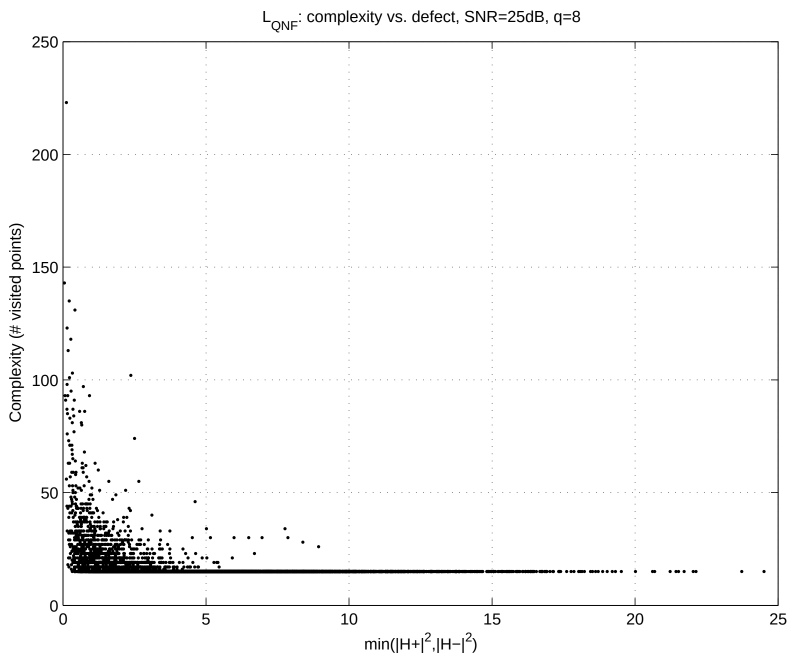


(b) Complexity vs. minimum diagonal element of R with L_{QNF} and an SNR of 19dB

Figure 13: Distributions of 10,000 transmissions using the code L_{QNF}



(a) Complexity vs. defect with L_{QNF} and an SNR of 19dB



(b) Complexity vs. defect with L_{QNF} and an SNR of 25dB

Figure 14: Complexity distributions of 10,000 transmissions using the code L_{QNF} and different SNRs

7 Improving the Performance of Sphere Decoding

In this chapter we look for ways to reduce the complexity of sphere decoding by two methods: first by modifying the algorithm and second, by taking advantage of the properties of the lattice.

7.1 Adding Constraints

The fact that we are using finite code sets gives us an opportunity to improve the algorithm. The idea is to avoid going through points that are outside the signal set. The original algorithm picks the midpoint of the interval on each level as the starting point and zigzags its way from there until it finds a valid point or is outside the sphere. If the midpoint is far below or above the signal set $[0, Q-1]$, the algorithm may have to go through a lot of points before reaching the signal set values. As we know that any points outside the signal set cannot be a part of the solution, we can improve the algorithm by jumping straight to the first possible value on each level. If the midpoint of the interval is smaller than 0, we can pick 0 as the starting point, as it is the first value belonging to the signal set that the algorithm would come across. Correspondingly, if the midpoint is greater than $Q-1$, we can pick $Q-1$ as the starting point.

Another case where Algorithm II may go through unnecessarily many points in the search tree, is when it has already processed all of the values that belong to the signal set on that level, but still has not gone outside the radius. After that point all of the following values that are considered cannot belong to the solution, since they are outside the signal set. We can now add another check to the algorithm: if all of the values from 0 to $Q-1$ have been processed, we can finish on that level and move up a level.

Example 7.1. Let us consider only the first level of the search tree and assume that the corresponding diagonal element of the R matrix $r_{n,n} = 0.1$, the received point $y'_n = 1.3$, signal set size $Q = 8$ and the squared radius $C'_0 = 5$. The interval for Algorithm II on this level has a lower boundary $A_n = \lceil (y'_n - \sqrt{C'_0})/r_{n,n} \rceil = -9$ and an upper boundary $B_n = \lfloor (y'_n + \sqrt{C'_0})/r_{n,n} \rfloor = 35$. Algorithm II starts zigzagging from the middle of the interval $x_n := \lfloor y'_n/r_{n,n} \rceil = 13$ and continues until it has gone through all of the values in the interval. Since the signal set is $[0, 7]$ the algorithm goes through points 13, 14, 12, 15, 11, 16, 10, 17, 9, 18, 8, 19 before reaching 7 as the first value belonging to the signal set. By adding the first check described above to the algorithm we could leave out the points in between and jump straight from 13 to 7 thus saving time. After 7 the next values assigned to x_n are 20, 6, 21, 5, 22, 4, 23, 3, 24, 2, 25, 1, 26, 0 and 27.

At $x_n := 27$, all of the values of the signal set have been processed, but x_n is still not outside the radius. The first value outside the radius is 36. By adding the second check to the algorithm we could now stop processing on this level without going through the rest of the points in the interval.

The amount of these unnecessary points in the search tree depends greatly on the R matrix and its diagonal elements. The smaller the diagonal element, the longer the interval that Algorithm II has to go through on that level.

In practice the first modification is added to step 2 of the algorithm, right after the part where the value for x_i is calculated. The second modification is added to step 3 after checking whether x_i belongs to the signal set. There is also a flow chart of the modified algorithm on page 41.

Modified algorithm (Input C_0', $\mathbf{y}'$, $\mathbf{R}$, Output $\hat{\mathbf{x}}$):

Step 1 (Initialization) Set $i := n$, $T_n := 0$, $\xi_n := 0$, and $d_c := C_0'$ (current sphere squared radius).

Step 2 (DFE on x_i) Calculate $x_{temp} := \lfloor (y_i' - \xi_i)/r_{i,i} \rceil$.

If $x_{temp} < 0$ set $x_i := 0$ and $\Delta_i := 1$.

Else if $x_{temp} > Q - 1$ set $x_i := Q - 1$ and $\Delta_i := -1$.

Else set $x_i := x_{temp}$ and $\Delta_i := \text{sign}(y_i' - \xi_i - r_{i,i}x_i)$.

Step 3 (Main step) If $d_c < T_i + |y_i' - \xi_i - r_{i,i}x_i|^2$, then go to Step 4 (i.e., we are outside the sphere).

Else if $x_i \notin [0, Q-1]$ (i.e., we are inside the sphere but outside the signal set boundaries) then {Set $x_{next} := x_i + \Delta_i$. If ($x_i < 0$ and $x_{next} > Q - 1$) or ($x_i > Q - 1$ and $x_{next} < 0$) then go to Step 4, otherwise go to Step 6 }.

Else (i.e., we are inside the sphere and signal set boundaries) if $i > 1$, then {let $\xi_{i-1} := \sum_{j=1}^{n} r_{i-1,j}x_j$, $T_{i-1} := T_i + |y_i' - \xi_i - r_{i,i}x_i|^2$, $i := i - 1$, and go to Step 2}.

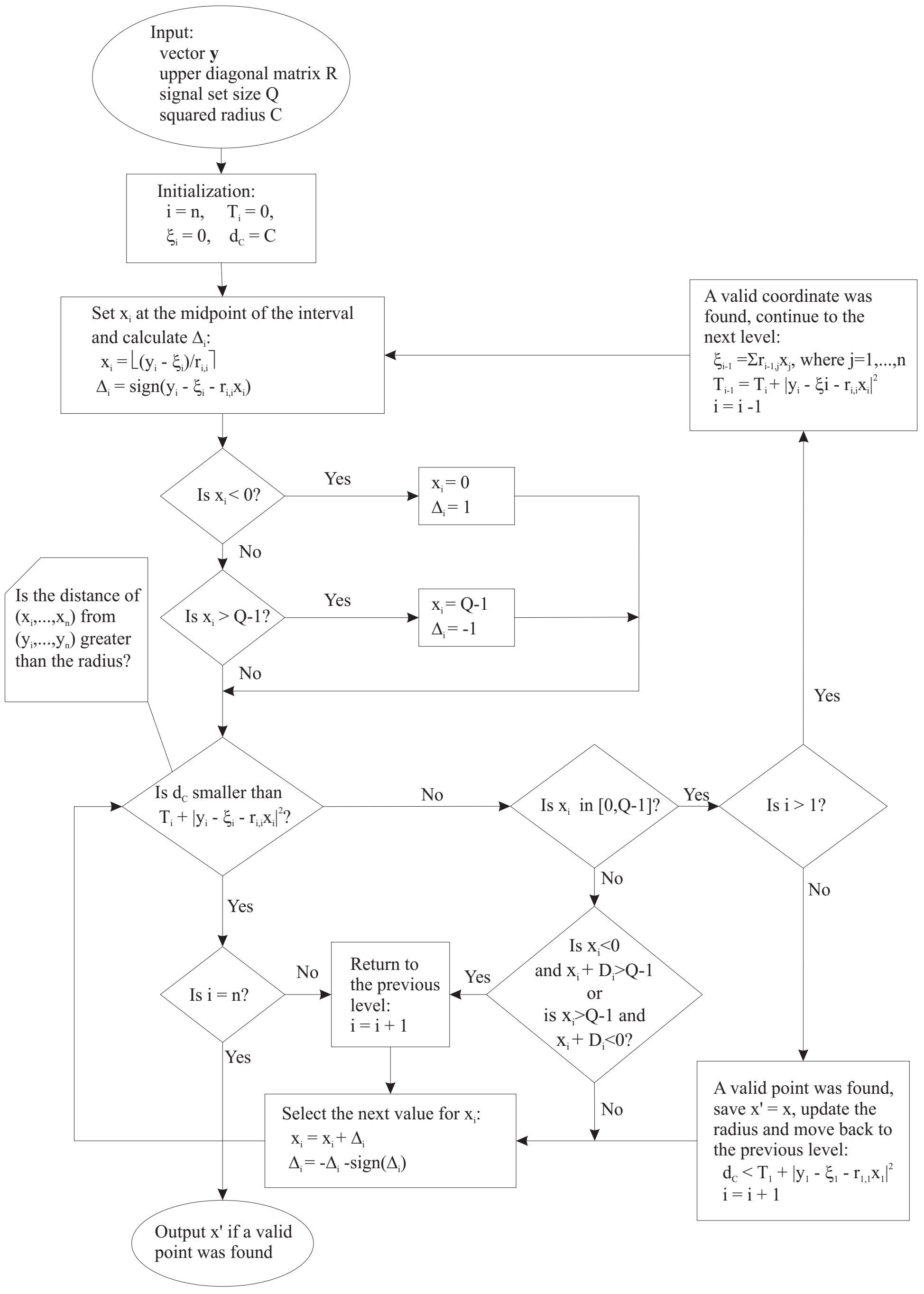
Else ($i = 1$) go to Step 5.

Step 4 If $i = n$, terminate, else set $i := i + 1$ and go to Step 6.

Step 5 (A valid point is found) Let $d_c := T_1 + |y_1' - \xi_1 - r_{1,1}x_1|^2$, save $\hat{\mathbf{x}} := \mathbf{x}$. Then, let $i := i + 1$ and go to Step 6.

Step 6 (Schnorr–Euchner enumeration of level i) Let $x_i := x_i + \Delta_i$, $\Delta_i := -\Delta_i - \text{sign}(\Delta_i)$, and go to Step 3.

□



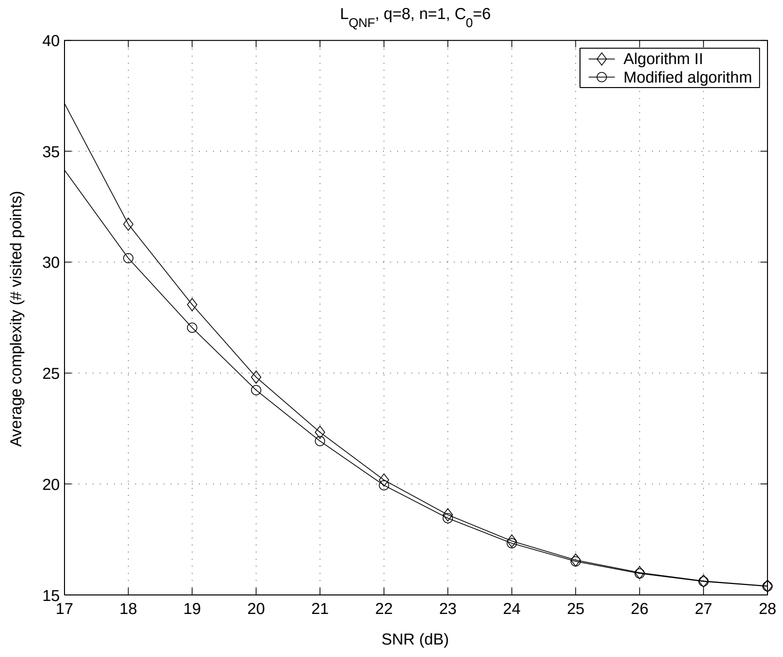
Input: vector y, upper diagonal matrix R, signal set size Q, squared radius C
Initialization: i = n, T_i = 0, ξ_i = 0, d_C = C
Set x_i at the midpoint of the interval and calculate Δ_i: x_i = ⌊(y_i - ξ_i)/r_i,i⌉ Δ_i = sign(y_i - ξ_i - r_i,i x_i)
Is x_i < 0?
Yes
x_i = 0 Δ_i = 1
No
Is x_i > Q-1?
Yes
x_i = Q-1 Δ_i = -1
No
Is the distance of (x_i,...,x_n) from (y_i,...,y_n) greater than the radius?
Is d_C smaller than T_i + |y_i - ξ_i - r_i,i x_i|²?
No
Is x_i in [0,Q-1]?
Yes
Is i > 1?
Yes
A valid coordinate was found, continue to the next level: ξ_i-1 = Σr_i-1,j x_j, where j=1,...,n T_i-1 = T_i + |y_i - ξi - r_i,i x_i|² i = i -1
No
A valid point was found, save x' = x, update the radius and move back to the previous level: d_C < T_1 + |y_1 - ξ_1 - r_1,1 x_1|² i = i + 1
No
Is x_i<0 and x_i + D_i>Q-1 or is x_i>Q-1 and x_i + D_i<0?
Yes
No
Return to the previous level: i = i + 1
Yes
Is i = n?
No
Yes
Select the next value for x_i: x_i = x_i + Δ_i Δ_i = -Δ_i -sign(Δ_i)
Output x' if a valid point was found

Figure 15: Flow chart of the modified sphere decoding algorithm

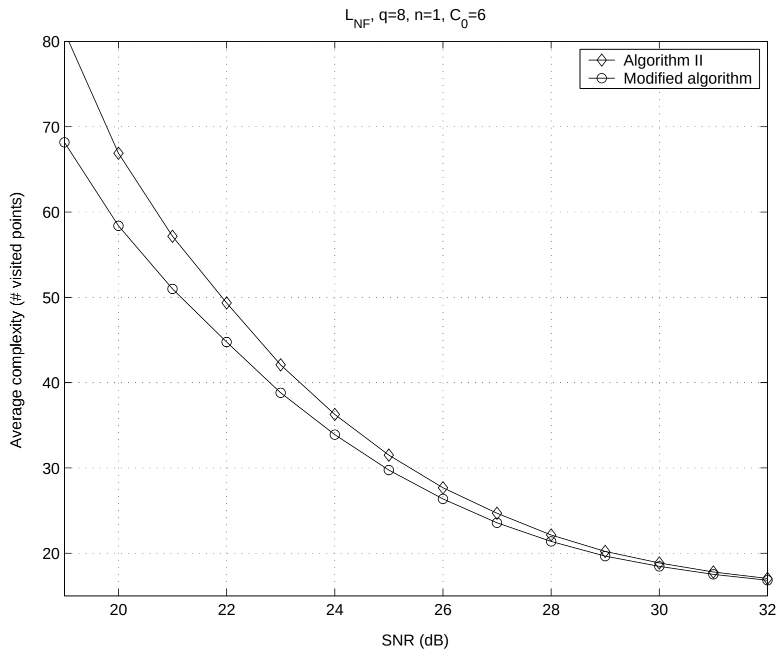
Figures 16(a) and 16(b) show a comparison of Algorithm II and the modified algorithm using the codes L_{QNF} and L_{NF} respectively. The complexity is measured in average number of points visited by the algorithm. It can be seen from the charts that the modifications indeed offer a slight reduction in complexity for both codes.

The reduction is more significant in smaller SNRs. This is a natural result, because as the SNR gets smaller, the influence of the noise grows and both the probability of error and the complexity increase. Then also the size of the search tree grows and it is more common that there are points in the search tree that do not belong to the signal set.

Figures 17(a) and 17(b) show the same comparisons measured in average CPU time used by the algorithms. It can be seen that the reduction in complexity is also visible in terms of CPU time. This means that even though the additional checks in the modified algorithm increase its complexity somewhat, the reduction they offer in the number of visited points more than makes up for the overhead.

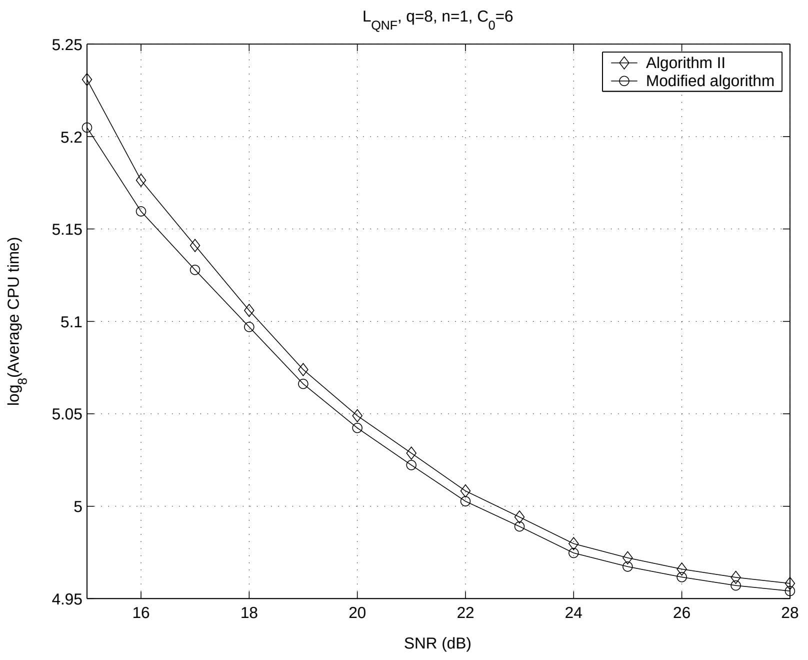


(a) The code L_{QNF} is used for transmission

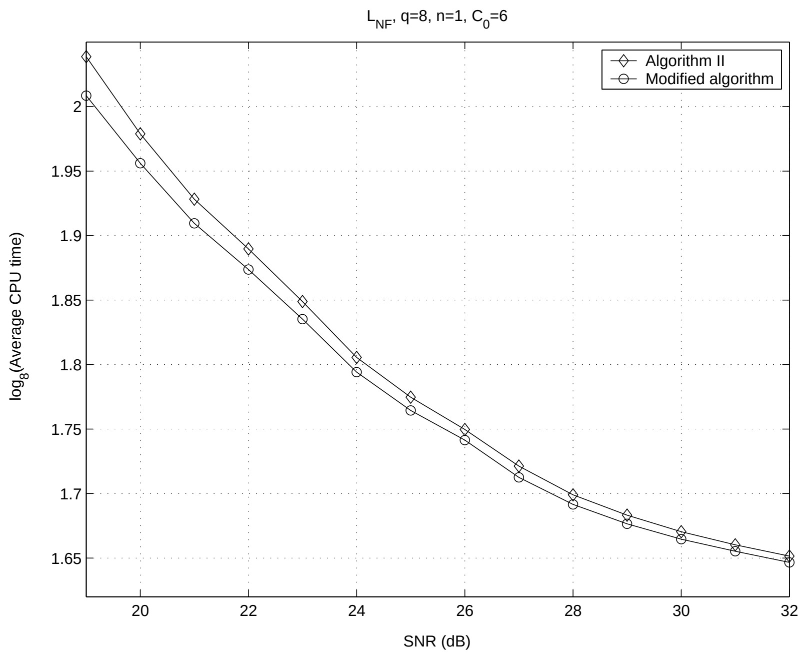


(b) The code L_{NF} is used for transmission

Figure 16: The average complexity of Algorithm II compared with the modified algorithm in a system with one receive antenna and $q = 8$. Complexity is measured in average number of points in the search tree.



(a) The code L_{QNF} is used for transmission



(b) The code L_{NF} is used for transmission

Figure 17: The average complexity of Algorithm II compared with the modified algorithm in a system with one receive antenna and $q = 8$. Complexity is measured in average CPU time used.

7.2 Distributed Search

Recently, new sphere decoding algorithms with lower decoding complexity have been developed, that take advantage of the zeros in the R-matrix; see e.g. the articles [4] and [3]. In the following we introduce a new approach to sphere decoding with a similar idea to utilize the zeros in the R-matrix. It is enabled by the structure of the lattice code L_{QNF}.

At the receiver we compute the complex matrices $\mathbf{HX}_1, \mathbf{HX}_2, \ldots, \mathbf{HX}_8$ and turn them into a real matrix $\mathbf{B}$ as described in Section 5.1. When the code L_{QNF} is used for encoding and there is only one receive antenna, we have

$$\mathbf{H} = \mathbf{h}_1 = (h_1, h_2, h_3, h_4)$$

and the matrix $\mathbf{B}$ is of the form:

$$\begin{pmatrix}
\Re(h_1) & -\Im(h_1) & \Re(h_2) & -\Im(h_2) & \Re(h_3) & -\Im(h_3) & \Re(h_4) & -\Im(h_4) \\
\Im(h_1) & \Re(h_1) & \Im(h_2) & \Re(h_2) & \Im(h_3) & \Re(h_3) & \Im(h_4) & \Re(h_4) \\
\Re(h_2) & -\Im(h_2) & -\Im(h_1) & -\Re(h_1) & \Re(h_4) & -\Im(h_4) & -\Im(h_3) & -\Re(h_3) \\
\Im(h_2) & \Re(h_2) & \Re(h_1) & -\Im(h_1) & \Im(h_4) & \Re(h_4) & \Re(h_3) & -\Im(h_3) \\
\Re(h_3) & \Im(h_3) & \Im(h_4) & -\Re(h_4) & -\Re(h_1) & -\Im(h_1) & -\Im(h_2) & \Re(h_2) \\
\Im(h_3) & -\Re(h_3) & -\Re(h_4) & -\Im(h_4) & -\Im(h_1) & \Re(h_1) & \Re(h_2) & \Im(h_2) \\
\Re(h_4) & \Im(h_4) & \Re(h_3) & \Im(h_3) & -\Re(h_2) & -\Im(h_2) & -\Re(h_1) & -\Im(h_1) \\
\Im(h_4) & -\Re(h_4) & \Im(h_3) & -\Re(h_3) & -\Im(h_2) & \Re(h_2) & -\Im(h_1) & \Re(h_1)
\end{pmatrix}$$

Let us denote the above matrix relating to row vector $\mathbf{h}_1$ as $\mathbf{B}_{h_1}$. If there are n receive antennas, then

$$\mathbf{H} = \begin{pmatrix} \mathbf{h}_1 \\ \vdots \\ \mathbf{h}_n \end{pmatrix}$$

and $\mathbf{B}$ is a $8n \times 8$ matrix of the form

$$\mathbf{B} = \begin{pmatrix} \mathbf{B}_{h_1} \\ \vdots \\ \mathbf{B}_{h_n} \end{pmatrix}.$$

The QR-decomposition is then computed for $\mathbf{B}$ and the R-matrix is given as input to the sphere decoder. Let us now take a look at the properties of the R-matrix.

Theorem 7.1. *For the code L_{QNF}, the R-matrix is of the form:*

$$\mathbf{R} = \begin{pmatrix} r_{1,1} & r_{1,2} & r_{1,3} & r_{1,4} & 0 & 0 & 0 & 0 \\ 0 & r_{2,2} & r_{2,3} & r_{2,4} & 0 & 0 & 0 & 0 \\ 0 & 0 & r_{3,3} & r_{3,4} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & r_{4,4} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & r_{5,5} & r_{5,6} & r_{5,7} & r_{5,8} \\ 0 & 0 & 0 & 0 & 0 & r_{6,6} & r_{6,7} & r_{6,8} \\ 0 & 0 & 0 & 0 & 0 & 0 & r_{7,7} & r_{7,8} \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & r_{8,8} \end{pmatrix}.$$

Proof. We prove this by showing that the Gram–Schmidt procedure introduced in Chapter 2.3 gives the R-matrix the described form, when applied to matrix $\mathbf{B}$. Let $\mathbf{R}_{GS}$ denote the R-matrix obtained by the Gram–Schmidt procedure from $\mathbf{B}$.

$$\mathbf{R}_{GS} = \begin{bmatrix} \|\mathbf{u}_1\| & \langle \mathbf{b}_2, \mathbf{e}_1 \rangle & \langle \mathbf{b}_3, \mathbf{e}_1 \rangle & \dots & \langle \mathbf{b}_n, \mathbf{e}_1 \rangle \\ 0 & \|\mathbf{u}_2\| & \langle \mathbf{b}_3, \mathbf{e}_2 \rangle & \dots & \langle \mathbf{b}_n, \mathbf{e}_2 \rangle \\ 0 & 0 & \|\mathbf{u}_3\| & & \vdots \\ \vdots & \vdots & & \ddots & \langle \mathbf{b}_n, \mathbf{e}_{n-1} \rangle \\ 0 & 0 & \dots & 0 & \|\mathbf{u}_n\| \end{bmatrix}$$

Now we prove that the upper right corner of $\mathbf{R}_{GS}$ is all zeros. Note that the matrix $\mathbf{B}$ has the properties that the first four column vectors are all orthogonal with the last four column vectors, i.e. $\langle \mathbf{b_i}, \mathbf{b_j} \rangle = 0$ for all $i \in [1,4]$ and $j \in [5,8]$.

Since

$$\langle \mathbf{b}_j, \mathbf{e}_i \rangle = \langle \mathbf{b}_j, \frac{\mathbf{u}_i}{\|\mathbf{u}_i\|} \rangle = \frac{1}{\|\mathbf{u}_i\|} \langle \mathbf{b}_j, \mathbf{u}_i \rangle,$$

it suffices to show that $\langle \mathbf{b}_j, \mathbf{u}_i \rangle = 0$ for all $i \in [1,4]$ and $j \in [5,8]$:

$$\begin{aligned} \langle \mathbf{b}_j, \mathbf{u}_1 \rangle &= \langle \mathbf{b}_j, \mathbf{b}_1 \rangle = 0, \\ \langle \mathbf{b}_j, \mathbf{u}_2 \rangle &= \langle \mathbf{b}_j, \mathbf{b}_2 - \phi_{2,1} \mathbf{u}_1 \rangle = \underbrace{\langle \mathbf{b}_j, \mathbf{b}_2 \rangle}_{=0} - \phi_{2,1} \underbrace{\langle \mathbf{b}_j, \mathbf{u}_1 \rangle}_{=0} = 0, \\ \langle \mathbf{b}_j, \mathbf{u}_3 \rangle &= \langle \mathbf{b}_j, \mathbf{b}_3 - \phi_{3,2} \mathbf{u}_2 - \phi_{3,1} \mathbf{u}_1 \rangle \\ &= \underbrace{\langle \mathbf{b}_j, \mathbf{b}_3 \rangle}_{=0} - \phi_{3,2} \underbrace{\langle \mathbf{b}_j, \mathbf{u}_2 \rangle}_{=0} - \phi_{3,1} \underbrace{\langle \mathbf{b}_j, \mathbf{u}_1 \rangle}_{=0} = 0, \\ \langle \mathbf{b}_j, \mathbf{u}_4 \rangle &= \langle \mathbf{b}_j, \mathbf{b}_4 - \phi_{4,3} \mathbf{u}_3 - \phi_{4,2} \mathbf{u}_2 - \phi_{4,1} \mathbf{u}_1 \rangle \\ &= \underbrace{\langle \mathbf{b}_j, \mathbf{b}_4 \rangle}_{=0} - \phi_{4,3} \underbrace{\langle \mathbf{b}_j, \mathbf{u}_3 \rangle}_{=0} - \phi_{4,2} \underbrace{\langle \mathbf{b}_j, \mathbf{u}_2 \rangle}_{=0} - \phi_{4,1} \underbrace{\langle \mathbf{b}_j, \mathbf{u}_1 \rangle}_{=0} = 0. \end{aligned}$$

□

Since the upper right corner is all zeros, the R-matrix actually has form

$$\mathbf{R} = \begin{bmatrix} \mathbf{R_1} & \mathbf{0} \\ \mathbf{0} & \mathbf{R_2} \end{bmatrix}$$

where $\mathbf{R_1}$ and $\mathbf{R_2}$ are upper diagonal matrices.

The zeros in the upper right corner mean that in the sphere decoder search the first four values of x_i found by the sphere decoder, where $i = 8, \ldots, 5$, do not have any effect on what the next four values of x_i, where $i = 4, \ldots, 1$, will be. We can use this fact to divide the search into two parts. We denote $\mathbf{y_1'} = (y_1', y_2', y_3', y_4')^T$, $\mathbf{y_2'} = (y_5', y_6', y_7', y_8')^T$, $\mathbf{x_1} = (x_1, x_2, x_3, x_4)^T$ and $\mathbf{x_2} = (x_5, x_6, x_7, x_8)^T$.

$$\begin{aligned}
|\mathbf{y}' - \mathbf{R}\mathbf{x}|^2 &\leq C_0' \\
\left| \begin{bmatrix} \mathbf{y_1'} \\ \mathbf{y_2'} \end{bmatrix} - \begin{bmatrix} \mathbf{R_1} & \mathbf{0} \\ \mathbf{0} & \mathbf{R_2} \end{bmatrix} \begin{bmatrix} \mathbf{x_1} \\ \mathbf{x_2} \end{bmatrix} \right|^2 &\leq C_0' \\
\left| \begin{bmatrix} \mathbf{y_1'} \\ \mathbf{y_2'} \end{bmatrix} - \begin{bmatrix} \mathbf{R_1 x_1} \\ \mathbf{R_2 x_2} \end{bmatrix} \right|^2 &\leq C_0' \\
|\mathbf{y_1'} - \mathbf{R_1 x_1}|^2 + |\mathbf{y_2'} - \mathbf{R_2 x_2}|^2 &\leq C_0'
\end{aligned}$$

We can now divide the processing into two searches for four symbols instead of one search for 8 symbols.

Distributed algorithm (Input C_0', $\mathbf{y}'$, $\mathbf{R}$, Output $\hat{\mathbf{x}}$):

Step 1 Obtain $\mathbf{R_1}$ and $\mathbf{R_2}$ from $\mathbf{R}$.

Step 2 Get $\mathbf{x_1}$ by performing Algorithm II with input C_0', $\mathbf{y_1'}$ and $\mathbf{R_1}$.

Step 3 Set radius for the second sphere decoder: $d_C := C_0' - |\mathbf{y_1'} - \mathbf{R_1 x_1}|^2$

Step 4 Get $\mathbf{x_2}$ by performing Algorithm II with input d_C, $\mathbf{y_2'}$ and $\mathbf{R_2}$.

Step 5 Return $\hat{\mathbf{x}} := [\mathbf{x_1}, \mathbf{x_2}]$.

$\square$

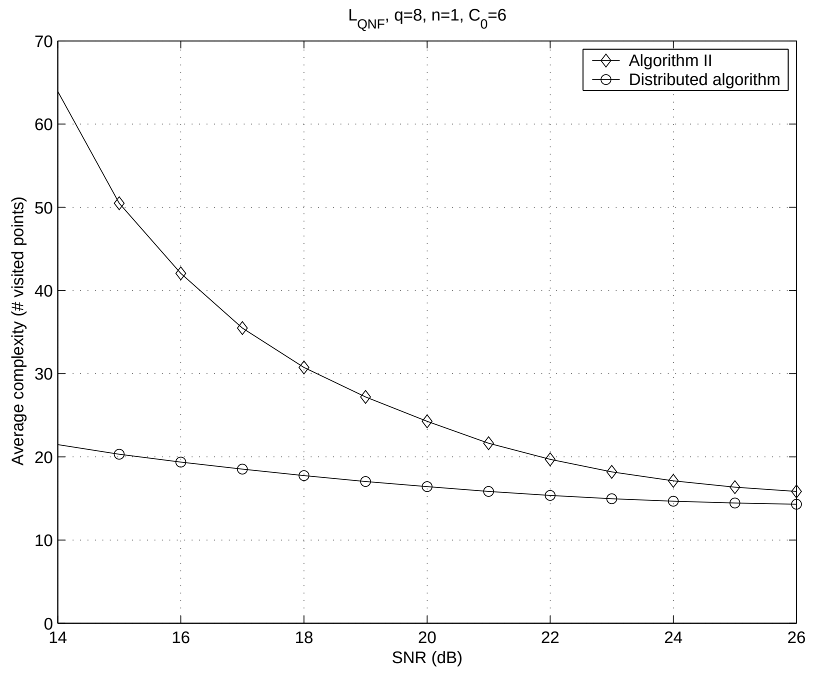
In the first step $\mathbf{R}$ is divided into $\mathbf{R_1}$ and $\mathbf{R_2}$ as defined above. In the second step we use the original sphere decoder to obtain the first four elements of $\mathbf{x}$. In step 3 we calculate a smaller radius for the second sphere decoder by subtracting the squared distance between $\mathbf{x_1}$ and $\mathbf{y_1'}$ from C_0'. In step 4 we use the sphere decoder again to find the last four elements of $\mathbf{x}$.

We could also leave out step 3 altogether and use the same squared radius for both searches. In this case the two searches could be done in parallel.

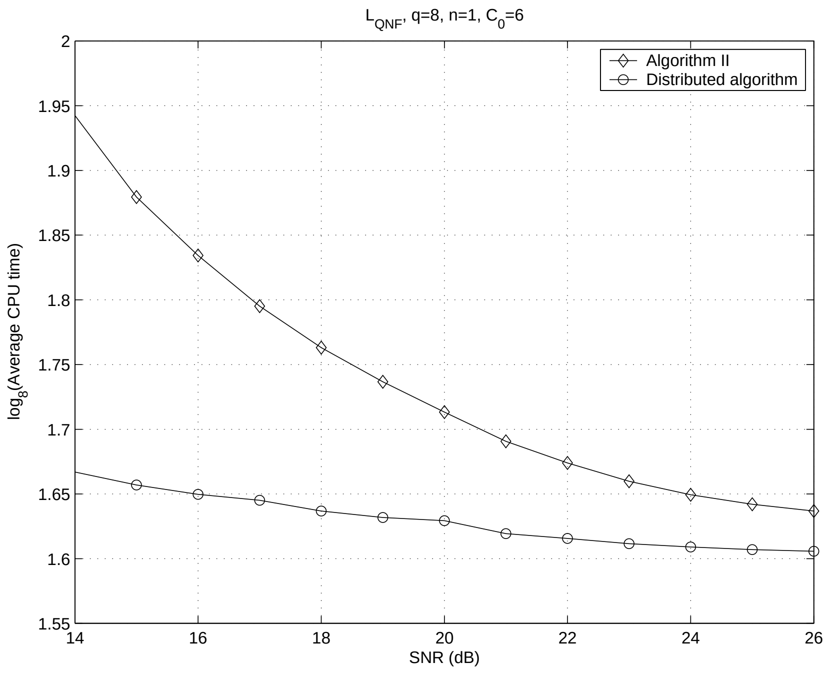
The average number of points in the search tree for the Distributed algorithm is calculated as the sum of the points visited by the sphere decoders in steps 2 and 3. As can be seen from the Figures 18(a) and 18(b), the reduction in complexity is remarkable, especially with lower SNRs. This is due to the fact that the search is divided into two distinct parts. The original sphere decoder has an 8-level search tree. This means that every time it finds suitable values for the first four x_i's, it continues the search to the lower levels of the search tree. With the Distributed algorithm the first search finds the optimal solution for the first four x_i's using a 4-level search tree and only after that will the next search look for the last four values of x_i. Also with an 8-level search tree the minimum amount of points in the search tree is 15, as with two 4-level searches the minimum is $2 * 7 = 14$.

Remark 7.1. The form of the R-matrix depends on the order of the basis matrices.

Remark 7.2. The code L_{NF} cannot be used together with Distributed algorithm, because its R-matrix does not have zeros in the right upper corner.



(a) Complexity measured in average number of points in the search tree



(b) Complexity measured in average CPU time

Figure 18: The average complexity of Algorithm II compared with that of the Distributed algorithm in a system with one receive antenna and $q = 8$.

References

[1] E. Agrell, T. Eriksson, A. Vardy and K. Zeger: *Closest point search in lattices.* IEEE Transactions on Information Theory, Vol. 48, pp.2201–2214, August 2002.

[2] S. M. Alamouti: *A Simple Transmit Diversity Scheme for Wireless Communications.* IEEE Journal on Selected Areas in Communications, Vol. 16, pp. 1451–1458, October 1998.

[3] L. Azzam and E. Ayanoglu: *Reduced Complexity Sphere Decoding for Square QAM via a New Lattice Representation.* arXiv:0705.2435v1 [cs.IT], May 2007.

[4] E. Biglieri, Y. Hong and E. Viterbo: *On Fast-Decodable Space–Time Block Codes.* arXiv:0708.2804v1 [cs.IT], August 2007.

[5] J.W.S. Cassels: *An Introduction to the Geometry of Numbers.* Springer Verlag, 1971.

[6] M.O. Damen, H. El Gamal and G. Caire: *On Maximum-Likelihood Detection and the Search for the Closest Lattice Point.* IEEE Transactions on Information Theory, Vol. 49, pp.2389–2402, October 2003.

[7] G. Foschini, G. Golden, R. Valenzuela and P. Wolniansky: *Simplified Processing for High Spectral Efficiency Wireless Communication Employing Multi-Element Arrays* IEEE Journal on Selected Areas in Communications, Vol. 17, pp. 1841–1852, November 1999.

[8] Gene H. Golub and Charles F. Van Loan: *Matrix Computations*, 3rd ed. Johns Hopkins, 1996.

[9] B. Hassibi and H. Vikalo: *Maximum-likelihood decoding and integer least-squares: The expected complexity.* Multiantenna Channels: Capacity, Coding and Signal Processing, (editors J. Foschini and S. Verdu), AMS 2003.

[10] C. Hollanti, J. Lahtonen and H.-f. Lu: *Maximal Orders in the Design of Dense Space–Time Lattice Codes.* Accepted for publication in IEEE Transactions on Information Theory (2008).

[11] M. Pohst: *On the Computation of Lattice Vectors of Minimal Length, Successive Minima and Reduced Basis with Applications.* ACM SIGSAM, vol. 15, pp. 37–44, 1981.

[12] J. Proakis: *Digital Communications*, 3rd ed. McGraw–Hill, 1995.

[13] V. Tarokh, H. Jafarkhani and A. R. Calderbank: *Space–Time Block Codes from Orthogonal Designs*. IEEE Transactions on Communications, Vol. 45, pp.1456–1467, July 2002.

[14] V. Tarokh, N. Seshadri and A. R. Calderbank: *Space–Time Codes for High Data Rate Wireless Communication: Performance Criterion and Code Construction*. IEEE Transactions on Information Theory, Vol. 44, pp.744–765, July 1998.