ADAPTIVE SPHERE DECODING

AND

RADIUS SELECTION WITH ERROR ANALYSIS IN SPHERE DECODING

ADAPTIVE SPHERE DECODING

AND

RADIUS SELECTION WITH ERROR ANALYSIS IN SPHERE DECODING

By

FEI ZHAO, B.Sc.

A Thesis

Submitted to the School of Graduate Studies

in Partial Fulfillment of the Requirement

for the Degree

Master of Science

McMaster University

Master of Science

Sep. 2007 - May 2009

Department of Computing & Software

McMaster University

Hamilton, Ontario

TITLE:	Adaptive Sphere Decoding and Radius Selection with Error Analysis in Sphere Decoding
AUTHOR:	Fei Zhao, B.Sc.(McMaster University)
SUPERVISOR:	Professor Sanzheng Qiao
NUMBER OF PAGES:	xii, 98

To my family and my parents for their encouragement.

Abstract

Many applications such as communications may be modeled as integer least squares problems. The goal is to find the solution to the integer least squares problem, which could be the encoded integer vector in these applications. From the point of view of lattice space, finding the solution to an integer least squares problem is equivalent to finding the closest lattice point to a given point. Sphere decoding is often applied to the searching of the closest lattice point.

An improved sphere decoding method, named adaptive sphere decoding, is discussed in this thesis. This method is examined from various views such as geometric interpretation and tree representation. The algorithm of adaptive sphere decoding is also presented. In addition, an experiment is conducted to show the improvement of performance provided by adaptive sphere decoding over the original sphere decoding.

One of the key issues in sphere decoding is the determination of the initial radius of a search hypersphere. For communication applications, the hypersphere radius could be computed from the statistical characteristics of signal noise or deterministically by Babai estimate. However, due to the computational error introduced during floating-point arithmetic, the initial radius computed by the

deterministic method may make sphere decoding fail. So, based on the standard computational error analysis of matrix-matrix multiplication and vector-vector addition, we investigate an error analysis for the numerical computation of the initial radius by the deterministic method and propose a revised deterministic method in computing the initial radius by taking the computational error into account in order to make sphere decoding as successful as possible. An experiment of comparing the two methods is conducted and the failure of sphere decoding is eliminated perfectly with the initial radius computed by the revised deterministic method.

Acknowledgments

I would like to express my sincere thanks to my supervisor, Prof. Sanzheng Qiao, for his professional and outstanding supervising during my research work towards the Master degree. During the period of graduate study, Prof. Qiao's invaluable support and continuous encouragement greatly increased my confidence and made the whole research process thoroughly enjoyable. I really appreciate McMaster Univeristy for offering me the opportunities of both undergraduate and graduate study. Special thanks will be also given to all the faculties and staffs in the Department of Computing & Software, whoever has ever taught me or has not. In particular, my truthful appreciation will be given to the examination committee for their time on reviewing my thesis and evaluating my oral defense.

This work is dedicated to my family who have provided me with the ideal environment to research. Their persistent emotional support always gives me the strength to overcome all kinds of diffculties. I would like to extend my special thanks to those faithful friends as well, for the kind help they have ever offered to me and every wonderful minute we have shared together.

Contents

List of Figures

List of Tables

Notations and Abbreviations

Notations

- $\mathbb{Z}$: Set of integer
- $\mathbb{R}$: Set of real
- $\mathbb{C}$: Set of complex
- $s, s \in \mathbb{Z}^m$: the vector of size m whose entries are integer
- $t, t \in \mathbb{R}^m$: the vector of size m whose entries are real
- $\mathcal{D}_L$: Set of L-PAM constellation
- $\mathcal{N}(0, \sigma^2)$: Gaussian normal distribution with mean 0 and variance σ^2
- $\leftarrow$: assignment
- $\triangleq$: definition
- A^T : tranpose of matrix A
- B^{-1} : inverse of matrix B
- $c_{i,j}$: the (i, j)th entry of matrix C
- $d_{i,1..m}$: the ith row of a matrix D of size $n \times m$

- $\lfloor x \rfloor$, $x \in \mathbb{R}$: the greatest integer which is $\leq x$
- $\lceil y \rceil$, $y \in \mathbb{R}$: the least integer which is $\geq y$
- $\lceil z \rfloor$, $z \in \mathbb{R}$: the integer closest to z
- $\tilde{u}$: the computed result of corresponding exact result u

Abbreviations

- AWGN : Additive White Gaussian Noise
- FIR : Finite Impulse Response
- i.i.d. : independent and identically distributed
- ISI : Intersymbol Interference
- IUI : Interuser Interference
- MIMO : Multiple Input Multiple Output
- ML : Maximum-Likelihood
- PAM : Pulse Amplitude Modulation
- QAM : Quadrature Amplitude Modulation
- SD : Sphere Decoding
- ASD: Adaptive Sphere Decoding
- SNR : Signal-to-Noise Ratio
- ZF : Zero-Forcing

Chapter 1

Introduction

1.1 Motivation

The rapid development of digital technologies and the widespread use of these technologies make our life easier and allow the convenient exchange of information. This advancement of digital technologies has a huge impact on our commercial and personal activities as we are relying on the digital technologies nowadays. To handle and deal with these digital information, researchers devote large amount of time and efforts to study the techniques of digital information processing. For example, integer least squares problem is one of the techniques to model applications and process information. This thesis is focused on solving the integer least squares problem, specifically, is focused on one of the solving methods, the sphere decoding.

The integer least squares problem arises from many applications such as communications [1], cryptography [2] and GPS [3]. The goal is to find the

solution to the integer least squares problem, which could be the encoded integer vector in these applications. In order to solve the integer least squares problem, usually, a lattice interpretation is used. A lattice is a set of intersected points of a regular (but not necessarily orthogonal) m-dimensional grid. From the point of view of lattice, finding the solution to an integer least squares problem is equivalent to finding the closest lattice point to a given point.

In aforementioned applications, the lattice is usually represents the communication channel [1] or the encryption function [2], while the given point is the received or given vector. In this context, solving the integer least squares problem corresponds to the decoding or decryption process. Note, the lattice is usually fixed for a certain application. For example, due to the distinguished characteristics of a communication channel, the lattice is fixed for this channel. The lattice is also fixed for a cryptography due to the fixed encryption algorithm. Therefore, if an integer least squares problem is represented by a lattice, the lattice is known in advance, and only the given point should be considered as the input to the problem.

A brief overview of methods in solving integer least squares problem: In general, solving the integer least squares problem or equivalently finding the closest lattice point is an NP-hard problem [4]. The evaluation of the solving process involves the performance of the solving algorithm, namely, the complexity of the solving algorithm. The evaluation also involves the accuracy of the solution, namely, how close is the solved solution to the optimal solution.

- *Approximation and heuristic method*

[1] In communication or GPS.

[2] In cryptography.

In [1], Hassibi and Vikalo give an overview of solving integer least squares problem in communication applications with the focus "all practical systems employ some approximations, heuristics or combinations of them". The solving methods of approximation and heuristic targeting communication can be categorized into *Babai estimate* [5], *Nulling and cancelling* and *Nulling and cancelling with optimal ordering.*

- *Exact method*

However, the above methods only get an approximation of the solution but not the optimal solution. In [6], the question of obtaining the optimal solution is studied for V-BLAST [7] system, where it is shown that the exact solution significantly outperforms even the best heuristics. In order to obtain the optimal solution, it is obvious that one needs to search the entire lattice space, although theoretically feasible for finite lattices but not for infinite lattices. Moreover, searching entirely even a finite lattice requires an exponential complexity, which is infeasible in practice. Anyway, there exist some exact search methods that are more sophisticated and more effective than the full search. These methods are *Kannan's algorithm* [8] which searches only a restricted parallelogram, *KZ algorithm* [9] which is based on the Korkin-Zolotarev reduced basis [10] and *sphere decoding algorithm* [11] which searches a hypersphere of an initial radius.

1.2 Objectives

Sphere decoding is an efficient search method and widely used for the NP-hard integer least squares problem. It has raised a lot of researchers' attention and has been rediscovered in several contexts since its first occurrence by Fincke and Pohst [11]. Sphere decoding is the focus of this thesis and it is the search technique used for finding the closest lattice point in this thesis. The basic idea of sphere decoding is quite simple: searching the lattice points which are lying inside a hypersphere of a certain radius and centered at a given vector rather than searching the entire lattice space, therefore, the search effort is reduced and the required computational complexity is mitigated as well.

The main objective of this thesis is to enhance the search technique of finding the closest lattice point, namely, enhance the efficiency and performance of sphere decoding. To achieve this objective, this thesis concentrates on improving the search technique, sphere decoding itself, as well as reducing the deterministic factor of sphere decoding, the initial radius of a search hypersphere given to sphere decoding as much as possible. In addition, it also addresses the issue of computational error. The main objective includes the following three sub-objectives.

- *Improving sphere decoding itself*

  Although sphere decoding is an efficient search method in finding the closest lattice point, in order to find the closest lattice point inside a hypersphere, the decoding process has to search all lattice points in this hypersphere and compare their distances to the center to determine the closest

lattice point. This approach constitutes an exhaustive search. So improving sphere decoding technique is one of the objectives of this thesis. The goal of improvement is to make sphere decoding avoid going through all the lattice points in a hypersphere, hence the improvement should outperform the original sphere decoding.

- *Reducing the initial radius*

  One of the deterministic factors of sphere decoding is the selection of the initial radius. Clearly, when an initial radius is too large, then too many lattice points are contained in this hypersphere, which degrades the efficiency of sphere decoding. Roughly speaking, the number of lattice points in this hypersphere is exponential to the radius of this hypersphere, so is the complexity of sphere decoding. Thus, attempting to reduce the initial radius of sphere decoding is another objective of this thesis.

- *Considering the computational error*

  For any numerical processing, floating-point errors are inevitable, this is also true for sphere decoding. When an initial radius is very closed to the distance from the closest lattice point to the center, the computational error has to be taken into account when computing the initial radius and there should be some ways to compensate the computational error to avoid the failure of sphere decoding caused by the computed, too small initial radius. Thus, error analysis of computing the initial radius is other objective of this thesis.

1.3 Outline of the thesis

In this thesis, we focus on the improvement of sphere decoding, the issue of radius selection of sphere decoding and the error analysis of computing the initial radius. The outline of this thesis is as follows.

Chapter 2 describes the communication channel model, specially, the frequency-selective channel model, which is also our simulated application in the experiments.

In Chapter 3, we introduce the integer least squares problem and the lattice interpretation of it, and geometrically interpret the lattice space and QR decomposition. Then we focus on sphere decoding from various views, both geometric interpretation and tree representation of sphere decoding. We also discuss in this chapter the mathematics of sphere decoding and present the algorithm of sphere decoding in both a recursive approach and a non-recursive approach.

In Chapter 4, we discuss in details a new improved sphere decoding, named adaptive sphere decoding from various views, such as geometric interpretation and tree representation. The algorithm of adaptive sphere decoding is also presented in this chapter.

One of the major issues in sphere decoding is the selection of the initial radius, which is investigated in Chapter 5. The statistical selection of the initial radius which arises from communication applications is introduced. Then another method, the deterministic selection of the initial radius is discussed. The algorithms of computing the initial radius in this two methods are presented thereafter. The radius update by adaptive sphere decoding is briefly discussed in this chapter as well.

As floating-point errors are inevitable in numerical computation process, the initial radius computed by the deterministic method may make sphere decoding fail. We show the failure of sphere decoding by an example in Chapter 6. Then we conduct an error analysis for the deterministic radius based on the error formulas of matrix-matrix multiplication and vector-vector addition, and propose a revised deterministic selection of initial radius. The algorithm of revised deterministic selection of initial radius is also presented and the revised deterministic method is used to compute the initial radius to show the success of sphere decoding.

In Chapter 7, a couple of experiments are conducted, they are targeting different purposes. First experiment evaluates the improvement of performance of adaptive sphere decoding over the original sphere decoding. The second experiment compares the statistical radius and deterministic radius for the simulated communication system. The third experiment shows the failure rate of sphere decoding with the deterministic radius due to the computational error, then it shows the elimination of failure of sphere decoding with the revised deterministic radius. Last, it is claimed that the revised deterministic radius does not impose a significant overhead on sphere decoding compared to the non-revised deterministic radius, which is verified in the fourth experiment.

Finally, in Chapter 8, some concluding remarks are given and possible directions for future work are advised.

Chapter 2

Communication Channel Model

2.1 Channel model

In wireless communication applications, there are increasing demands for high rate digital transmission systems. Multiple antennae wireless communication systems have the capabilities of providing data transmission at a high rate. But, in real communication systems, problems also come up with multiple antennae systems, the increased amount of noise, intersymbol interference (ISI) [1] and interuser interference (IUI) [2] perturbs the transmitted signals when transmitting over a noisy dispersive channel. The received signals are combinations of transmitted signals with noise, intersymbol interference and interuser interference.

If we consider a multiple antennae communication system with m transmitting antennae and n receiving antennae, $y \in \mathbb{C}^n$ is the received signal vector,

[1] In telecommunication, intersymbol interference is a form of distortion of a signal in which one symbol interferes with subsequent symbols.

[2] In some communication systems, interuser interference is known as multiple access interference.

$s \in \mathbb{C}^m$ is the transmitted signal vector, then a single user Gaussian channel with multiple transmitting and receiving antennae can be modeled as:

$$y = Hs + v, \tag{2.1}$$

where $H \in \mathbb{C}^{n \times m}$ is a complex channel matrix and $v \in \mathbb{C}^n$ is zero mean complex Gaussian noise with independent, equal variance real and imaginary parts [12]. The communication channel model is illustrated in Figure 2.1.

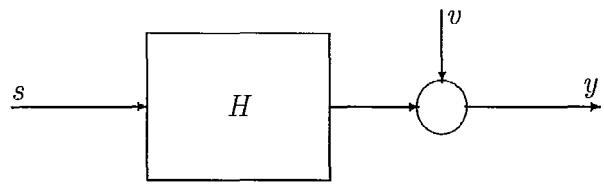

Figure 2.1: The discrete-time channel model

Figure 2.1 shows a communication channel described by a linear discrete-time model with additive white Gaussian noise (AWGN) based on the following assumptions [13].

- The channel H is linear.
- The channel matrix is of full column rank.
- The noise sequence v is additive white Gaussian $\mathbb{CN}(0, \sigma^2)$ with mean zero and variance σ^2.

Let us concentrate on the discrete-time channel model and consider the communication channel as dispersive multiple-input multiple-output (MIMO) channel with finite impulse response (FIR).

2.2 Frequency-selective channel

In [14], Vikalo and Hassibi describe the MIMO channel as block-fading frequency-selective model. The channel impulse response is constant for some discrete interval T, after which it changes to another impulse response that remains constant for another interval T, and so on. The additive noise is spatially and temporally independent and identically distributed (i.i.d) circularly-symmetric complex-Gaussian. The MIMO channel model is shown in Figure 2.2.

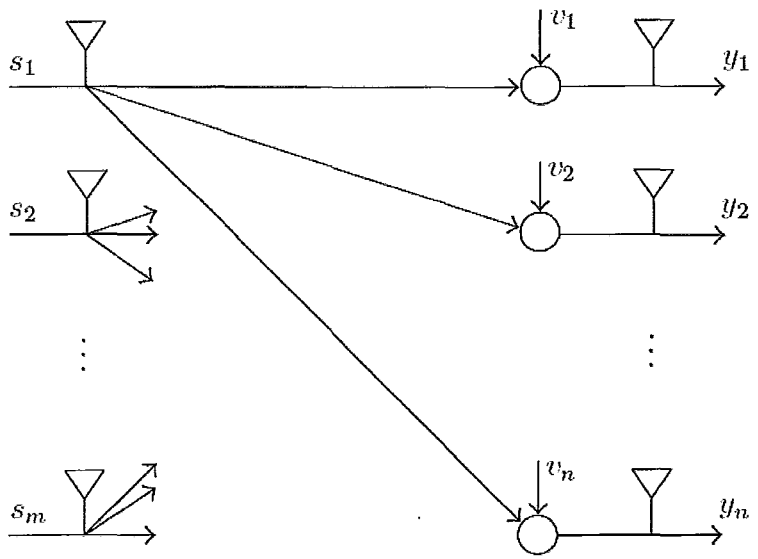

Figure 2.2: FIR MIMO channel model with m transmitting and n receiving antennae

Assuming that the source data sequence is transmitted in length of T symbols, let H be the channel impulse response of length $l+1$, where l is called the channel memory length or equivalently the number of interfering intersymbols, so we may think that the transmitted source data sequence is equally separated

by a guard interval of $l-1$ symbols. The frequency-selective model can be written as:

$$y_i = \sum_{j=1}^{i} h_j s_{i-j+1} + v_i, \tag{2.2}$$

where $h_i, i = 1, ..., l$ are the coefficients of the channel impulse response of length $l+1$, and assumed to be Gaussian $\mathcal{N}(0,1)$, s_i is the ith symbol in the transmitting source data sequence, and v_i is the ith component of the channel independent and identically distributed (i.i.d.) Gaussian noise $\mathcal{N}(0,\sigma^2)$.

We can see that Equation (2.1) and Equation (2.2) are equivalent, the difference is that Equation (2.2) is the component wise form of Equation (2.1) and the matrix in (2.2) has a special structure (shown later).

Remark 2.1. *This model is assumed to be real, in [15], there is detailed description and deduction for FIR MIMO model. Although the FIR MIMO model in [15] was described originally as complex, it is also mentioned that it is more useful to rewrite this model as real. In fact, if we need the model in complex case, it is equivalent to doubling the dimension size of the real model described here. In another words, the assumed real model here (even dimension) can be fitted for complex model of half dimension (for simplicity, it is usually to write the model in real, i.e., the dimension in real case is double of the dimension in complex case for the same model).*

From now on, we consider the channel model in Equation (2.1) as real instead of complex, where matrix $H \in \mathbb{R}^{(T+l-1)\times T}$, T is the length of transmitted symbols, l is the length of channel memory. In Equation (2.1), H has the form

of

$$H = \begin{bmatrix} h_1 & & & \\ h_2 & h_1 & & \\ \vdots & \ddots & \ddots & \\ h_l & \ddots & \ddots & h_1 \\ & h_l & \ddots & h_2 \\ & & \ddots & \vdots \\ & & & h_l \end{bmatrix},$$

s is the vector of the transmitted data sequences, y is the vector of the received data sequence and v is the vector of additive white Gaussian noise, they have the forms of

$$\begin{bmatrix} s_1 \\ s_2 \\ \vdots \\ s_T \end{bmatrix} \in \mathcal{D}_L^T, \qquad \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_T \\ \vdots \\ y_{T+l-1} \end{bmatrix} \in \mathbb{R}^{T+l-1}, \qquad \begin{bmatrix} v_1 \\ v_2 \\ \vdots \\ v_T \\ \vdots \\ v_{T+l-1} \end{bmatrix} \in \mathbb{R}^{T+l-1}$$

respectively, where $\mathcal{D}_L$ is an L-PAM constellation. We can see that the channel matrix has Toeplitz structure.

2.3 The transmitting data sequence

In communication systems, some modulation schemes such as Quadrature Amplitude Modulation (QAM), Pulse Amplitude Modulation (PAM) are very popular. If we treat Equation (2.1) as a complex version of communication channel

model, where the additive white Gaussian noise $v \in \mathbb{C}^n$ is composed of i.i.d. complex-Gaussian $\mathbb{C}\mathcal{N}(0, \sigma^2)$ entries, the channel matrix $H \in \mathbb{C}^{n \times m}$ is composed of i.i.d. complex-Gaussian $\mathbb{C}\mathcal{N}(0, 1)$ entries, then the transmitted vector $s \in \mathbb{C}\mathcal{D}_L^m$ is usually an m-dimensional vector whose entries are complex-valued elements chosen from an L^2-QAM constellation. i.e., both the real and the imaginary parts of entries of s are elements from an L-PAM constellation $\mathcal{D}_L$. L is usually taken as a power of 2.

On the other hand, if we treat Equation (2.1) as real for frequency-selective channel model, where the additive white Gaussian noise $v \in \mathbb{R}^n$ is composed of i.i.d. Gaussian $\mathcal{N}(0, \sigma^2)$ entries, the channel matrix $H \in \mathbb{R}^{n \times m}$ is composed of i.i.d. Gaussian $\mathcal{N}(0, 1)$ entries, then the transmitted vector $s \in \mathcal{D}_L^m$ is usually an m-dimensional vector whose entries are real-valued elements chosen from an L-PAM constellation $\mathcal{D}_L$. L is also usually taken as a power of 2.

For the purpose of simplicity and also for the frequency-selective channel model, we assume that the entries in the vector of transmitted data sequence s are chosen from points of an L-PAM constellation [1]

$$\mathcal{D}_L = \left\{ -\frac{L-1}{2}, -\frac{L-3}{2}, \cdots, \frac{L-3}{2}, \frac{L-1}{2} \right\} \tag{2.3}$$

and L is taken as a power of 2. We can say that the vector s belongs to the lattice-type signal constellation $\mathcal{D}_L^m$

$$\mathcal{D}_L^m = \underbrace{\mathcal{D}_L \times \mathcal{D}_L \times \cdots \times \mathcal{D}_L}_{m-times},$$

where the operation '$\times$' denotes the Cartesian product.

For various values of L, let us examine the entries of s respectively.

1. $\mathcal{D}_2^m$: For a 2-PAM constellation, the entries of s belong to the set $\{\pm\frac{1}{2}\}$.

2. $\mathcal{D}_4^m$: For a 4-PAM constellation, the entries of s belong to the set $\{\pm\frac{1}{2}, \pm\frac{3}{2}\}$.

3. $\mathcal{D}_8^m$: For a 8-PAM constellation, the entries of s belong to the set $\{\pm\frac{1}{2}, \pm\frac{3}{2}, \pm\frac{5}{2}, \pm\frac{7}{2}\}$.

4. $\mathcal{D}_{16}^m$ and higher: The set of entries of 16-PAM constellation can be obtained in a similar pattern, even if for a higher order L-PAM constellation.

2.4 The noise

Whatever communication channel we are considering, the interference and noise are inevitable. When a signal transmits over noisy dispersive channels, the received signal at each receiving antenna is not the original transmitted signal any more, actually, it is the combinations of the transmitted signal perturbed by noise, intersymbol interference and interuser interference. Therefore, the received signal becomes unsatisfactory since the interference and noise cannot be removed by simply raising the signal power.

As aforementioned, when we model the FIR MIMO channel, the intersymbol interference has already been considered into the channel model, now we need to investigate what impact the noise has on the communication channel. In 1972, Forney [16] introduced the whitened matched filter so that an important class of channels can be described by a linear discrete-time model with additive white Gaussian noise.

In engineering, signal-to-noise ratio (SNR) is often defined as power ratio between a meaningful signal and the background noise, and SNR is usually expressed in term of the logarithmic decibel scale. However, in communication, when white noise is assumed present, optimal signal processing systems can sometimes take it into account and their performances typically depend on a modified definition of signal-to-noise ratio. In our FIR MIMO channel model, the noise sequence v is additive white Gaussian $\mathcal{N}(0, \sigma^2)$ with mean zero and variance σ^2, but instead of the noise variance σ^2, we are more interested in the SNR. Specifically, define SNR for our communication model as in [1]:

$$\delta \triangleq \frac{m(L^2-1)}{12\sigma^2}. \tag{2.4}$$

By the SNR definition (2.4), we see that the SNR is closely related to the noise variance and dimension m. We can expect that the noise variance determines the SNR, therefore, impose a significant impact on the selection of the initial radius for sphere decoding by the statistical method (shown in the experiments).

Chapter 3

Sphere Decoding

3.1 Preliminary

Sphere decoding (SD) has got a lot of attention by researchers recently, it is able to solve the closest lattice point problem, i.e., finding the closest lattice point to a given point. It is a maximum-likelihood (ML) detection algorithm with a relatively low expected complexity. It is useful for many applications. For example, it has been adopted for different linear channels in digital communications. The complexity of sphere decoding is usually polynomial in most cases and under certain assumptions, although in some cases, it could be shown to be exponential.

The idea of sphere decoding was introduced originally by Finke and Pohst in [11] in 1985, which enumerates all lattice points within a hypersphere centered at the origin. The hypersphere radius is a decisive factor for the complexity of the algorithm and determines the number of points visited throughout the entire

searching process. The radius problem has been addressed and treated widely in the literature.

Sphere decoding is often composed of two stages:

1. Preprocessing stage

2. Searching stage

The first stage usually transforms the problem into a simpler form which makes the decoding process easier and more efficient. Some of the preprocessing methods such as QR decomposition and lattice reduction can be utilized for this purpose. The goal of the searching stage is to find the closest lattice point constrained by a hypersphere. In communication applications, a sphere decoder can be applied to the receiver side to obtain an optimal maximum-likelihood detection.

3.2 Integer least squares problem

3.2.1 Lattice interpretation

In general, the integer least squares problem is to find a minimizer, an m-dimensional integer vector $s \in \mathbb{Z}^m$ to the problem:

$$\min_{s \in \mathbb{Z}^m} \|Hs - y\|_2^2, \tag{3.1}$$

where $y \in \mathbb{R}^n$, $H \in \mathbb{R}^{n \times m}$. Here, s denotes an m-dimensional vector composed of integer entries. From the point of view of lattice, s also denotes an m-dimensional

integer lattice. H is called the lattice generating matrix.

We can interpret the integer least squares problem simply from a geometric lattice. Since the entries of vector s are composed of integers, s spans a "rectangular" m-dimensional lattice $\mathbb{Z}^m$ (see Figure 3.1).

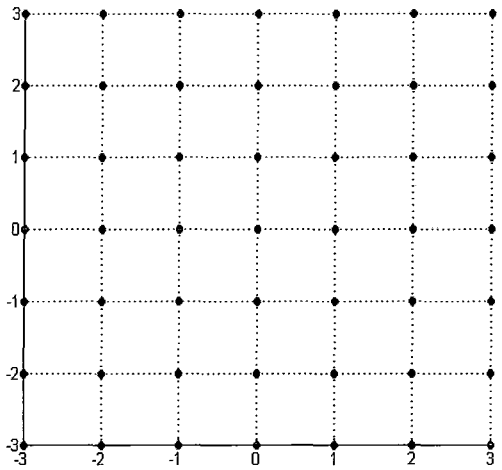

Figure 3.1: Geometric interpretation of a "rectangular" lattice

The lattice generating matrix H transforms the m-dimensional vector s to an n-dimensional vector Hs which spans a "skewed" lattice (it is "skewed" lattice relative to the m-dimensional space, see Figure 3.2). So, given a "skewed" lattice Hs and given a vector $y \in \mathbb{R}^n$, the integer least squares problem can be considered as to find the closest lattice point (in Euclidean sense) to the given vector y (this given vector y is in the n-dimensional space, also see Figure 3.2).

In Figure 3.2, the lattice generating matrix H transforms the "rectangular" lattice s (in Figure 3.1) to a "skewed" lattice, the solid points represent the "skewed" lattice Hs, which is a plane in space. The hollow point in Figure 3.2

represents the vector y, which is in a 3-dimensional space. The integer least squares problem is to find the solid point which is closest to the hollow point[1].

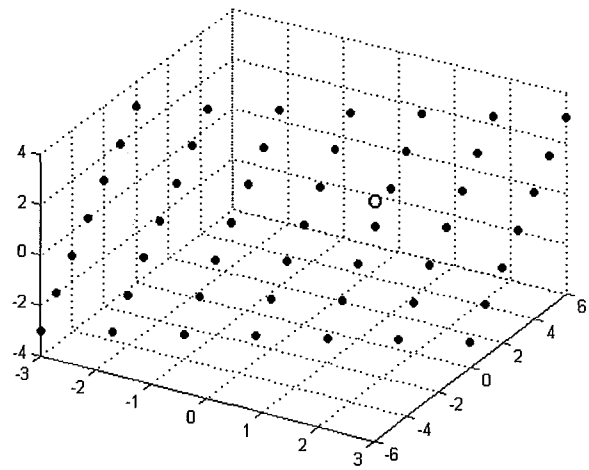

Figure 3.2: Geometric interpretation of "skewed" lattice generated by H and the integer least squares problem

The complexity of the general integer least squares problem has been studied by several people. It has been shown in [4] that it is an NP-hard problem, and a proof can be found in [17].

A typical method of solving the integer least squares problem (3.1) usually has two stages: reduction (preprocessing) and decoding (searching). A comprehensive survey of closest point search methods for lattices without a regular structure is presented in [18], which also provides an implementation of efficient closest point search algorithm, based on the Schnorr-Euchner variation of the Pohst method [19].

[1]Note, the hollow point may not be necessarily in the plane of solid points, although it is also possible for the hollow point lying in the plane of solid points.

3.2.2 QR decomposition

When the matrix H in (3.1) has full column rank, one of standard methods for preprocessing (3.1) is the QR decomposition method [20]. H is reduced to an upper triangular matrix by the QR decomposition

$$H = Q \begin{bmatrix} R \\ 0 \end{bmatrix},$$

where $Q \in \mathbb{R}^{n \times n}$ is orthogonal and $R \in \mathbb{R}^{m \times m}$ is upper triangular. Partition $Q = [Q_1, Q_2]$, where Q_1 is $n \times m$ and Q_2 is $n \times (n-m)$, we get

$$\begin{aligned} & \|Hs - y\|_2^2 \\ = & \left\| Q \begin{bmatrix} R \\ 0 \end{bmatrix} s - y \right\|_2^2 \\ = & \left\| \begin{bmatrix} R \\ 0 \end{bmatrix} s - Q^T y \right\|_2^2 \\ = & \left\| \begin{bmatrix} R \\ 0 \end{bmatrix} s - \begin{bmatrix} Q_1^T \\ Q_2^T \end{bmatrix} y \right\|_2^2 \\ = & \|Rs - Q_1^T y\|_2^2 + \|Q_2^T y\|_2^2 . \end{aligned}$$

We can do nothing about the second term $\|Q_2^T y\|_2^2$, however, we can minimize the first term $\|Rs - Q_1^T y\|_2^2$. Define $\hat{y} \triangleq Q_1^T y$, then the integer least squares problem (3.1) is reduced to a triangular integer least squares problem:

$$\min_{s \in \mathbb{Z}^m} \|Rs - \hat{y}\|_2^2. \tag{3.2}$$

3.2.3 Geometric interpretation of QR decomposition

We can interpret the QR decomposition upon H geometrically. If the QR decomposition is applied to the lattice generating matrix H, H is transformed to R, strictly speaking, H is reduced to R. Since H is an $n \times m$ matrix and R is an $m \times m$ matrix, the transformation is actually a reduction, it reduces the dimension from n to m. In Figure 3.3, the solid points form the "skewed" lattice generated by the lattice generating matrix H upon a "rectangular" lattice spanned by s. After the QR decomposition is applied to H, the "skewed" lattice Hs is reduced to Rs, which is formed by the hollow points in Figure 3.3. Although it is still a "skewed", it is in lower dimension (dimension m instead of dimension n). In the meanwhile, the point y (in dimension n) is also reduced to $\hat{y}$ (in dimension m), which has the same dimension as Rs. Geometrically, y is projected onto Rs, the m-dimensional subspace spanned by the columns of R. Therefore, the QR decomposition reduces the original integer least squares problem.

From the point of view of closest lattice point, after applying the QR decomposition to (3.1), the original integer least squares problem (3.1) is transformed (reduced) to find the closest lattice point (in Euclidean sense) to the point $\hat{y}$, where the new lattice points are under the basis formed by the columns of R.

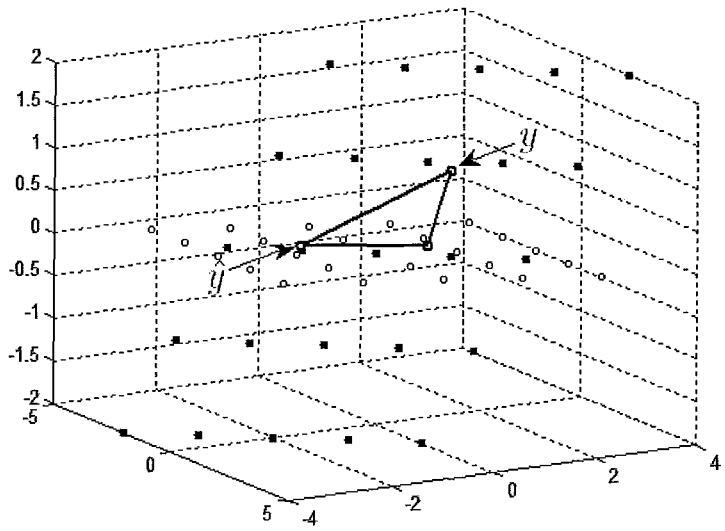

Figure 3.3: Geometric interpretation of QR decomposition

3.3 The sphere decoding

3.3.1 Overview of sphere decoding

Sphere decoding is an efficient search method for obtaining maximum-likelihood detection performance for the NP-hard integer least squares problem. The principle of sphere decoding is to search the closest lattice point to a given point inside a hypersphere of radius d centered at that given point, therefore it can reduce the computational complexity of the maximum-likelihood detection (see Figure 3.4).

The complexity of sphere decoding depends on the following two factors:

- The radius d of the hypersphere

  How do we choose the radius d? If d is too large, then we obtain too

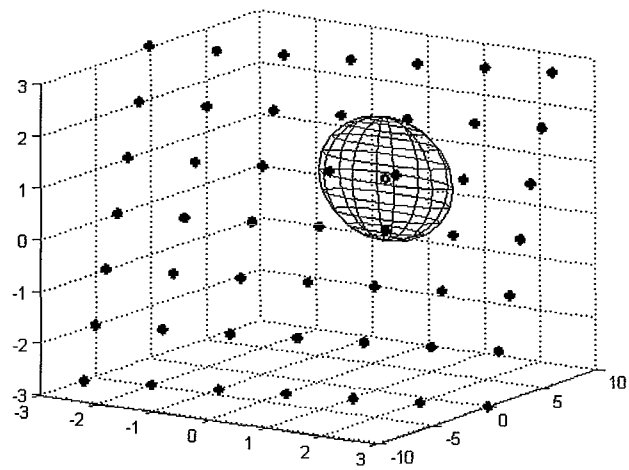

Figure 3.4: Geometric interpretation of a hypersphere on a lattice space

many points inside the hypersphere, and the search complexity may be exponential to d. If d is too small, then we obtain no points inside the hypersphere. In general, no guideline in choosing d, we do not know what size is appropriate for d, it depends on the applications.

- Determination of the lattice points inside the hypersphere

  How to determine whether a lattice point is inside the hypersphere or not? This requires to test the distance (in Euclidean sense) between each lattice point and the given central point to determine whether it is smaller than the hypersphere radius d or not. So, this is an exhaustive search.

3.3.2 Tree representation of sphere decoding

It is not easy to determine the lattice points inside a general m-dimensional hypersphere, so let us first start with the lower dimensional case. Obviously, the

hypersphere in 1-dimensional case is an interval, and the desired lattice points are the integers that lie within this interval. For $m = 2$, the desired lattice points in 2-dimensional case are the points inside a circle. For $m = 3$, the desired lattice points in 3-dimensional case are the points inside a sphere. And so on, for the higher dimensional spaces, the desired lattice points lie inside a hypersphere.

The sphere decoding method works in a reverse order (from m-dimension down to 1-dimension) and in component wise for lattice points. We know that an m-dimensional lattice point can also be considered as an m-dimensional vector. Suppose for an unknown lattice point (also an unknown vector), we have determined its mth to kth entries, the kth entries is now in an integer interval computed from one of the $(k+1)$th entries by searching a hypersphere of radius d_k in a k-dimensional space. For the lower $(k-1)$-dimensional space, we pick any one kth entry in the integer interval determined in the k-dimension, use it to compute the radius d_{k-1} for $(k-1)$-dimensional space, therefore, we are able to determine all the $(k-1)$th entries of this unknown lattice point by searching a hypersphere of radius d_{k-1} in the $(k-1)$-dimensional space, which follows the kth entry we picked in the k-dimension. Let us use the following example to illustrate the decoding process.

Example 1:

Suppose that we solve an integer least squares problem in a 2-dimensional space, so we decode the lattice points inside a circle. Before searching, the QR decomposition need to be applied and then the integer least squares problem (3.2) in 2-dimension is

$$\left\| \begin{bmatrix} r_{1,1} & r_{1,2} \\ 0 & r_{2,2} \end{bmatrix} \begin{bmatrix} s_1 \\ s_2 \end{bmatrix} - \begin{bmatrix} \hat{y}_1 \\ \hat{y}_2 \end{bmatrix} \right\|_2^2.$$

Assuming that an initial radius $\hat{d}$ is given, in order to search in a circle, expand it in component wise[2]

$$\hat{d}^2 \geq \|r_{1,1}s_1 + r_{1,2}s_2 - \hat{y}_1\|_2^2 + \|r_{2,2}s_2 - \hat{y}_2\|_2^2,$$

and it is necessary that

$$\hat{d}^2 \geq \|r_{2,2}s_2 - \hat{y}_2\|_2^2.$$

This determines all the possible values for the second entry s_2 of the closest lattice point inside this circle (see Figure 3.5). In Figure 3.5, there are three possible values (in vertical coordinate) for entry s_2. We pick one of the three possible values of entry s_2, compute the possible values (in horizontal coordinate) for entry s_1 determined by this s_2 value with

$$\hat{d}^2 - \|r_{2,2}s_2 - \hat{y}_2\|_2^2 \geq \|r_{1,1}s_1 + r_{1,2}s_2 - \hat{y}_1\|_2^2.$$

Therefore, the lattice points inside this circle can be found. We follow the same pattern to find other lattice points inside this circle. That is, pick another one of the three possible values of entry s_2, compute the possible values for entry s_1 determined by this s_2 value.

[2]The detailed mathematically analysis of sphere decoding algorithm is presented in the next subsection.

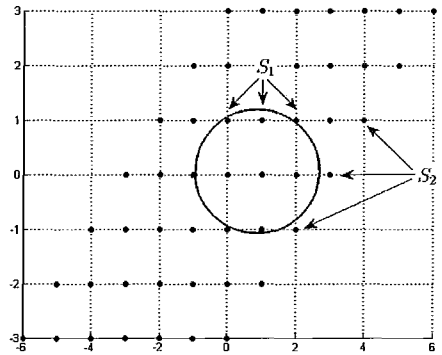

Figure 3.5: Geometric interpretation of a circle in a "skewed" 2-dimensional lattice space

Now let us translate the geometric interpretation (in Figure 3.5) to a tree representation. In Figure 3.5, sphere decoding found three values for entry s_2, the first value of entry s_2 determines one value for entry s_1 (entry s_2 and s_1 determine the bottom lattice point inside the circle in Figure 3.5), the second value of entry s_2 determines three values for entry s_1 (entry s_2 and s_1 determine the three middle lattice points inside the circle in Figure 3.5) and the third value of entry s_2 determines two values for entry s_1 (entry s_2 and s_1 determine the two top lattice points inside the circle in Figure 3.5). The corresponding tree representation of the geometric interpretation (Figure 3.5) is shown in Figure 3.6.

The above method means that we can compute all the $(k-1)$th entries of lattice points in a $(k-1)$-dimensional hypersphere of radius d_{k-1} by successively computing all the kth entries of lattice points in the k-dimensional hypersphere of radius d_k. If we start with $k = m$, this method for computing the lattice points in an m-dimensional hypersphere essentially constructs a tree of $depth = m + 1$

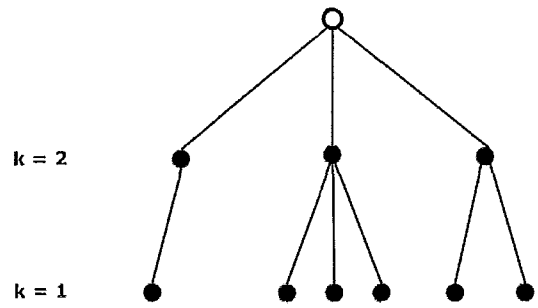

Figure 3.6: Tree representation generated by sphere decoding in an $m = 2$ dimensional lattice space

where the nodes in the kth level of the tree correspond to the kth entries of lattice points (see Figure 3.6). In Figure 3.6, the root node represents none entry of a vector (a lattice point) from the point of view of our decoding, as a matter of fact, the root node's children represent the top mth entries in m-dimensional space. From the view of this tree representation, the complexity of sphere decoding depends on the *size*(*density*) of the tree, i.e., depends on the number of nodes of the generated tree by sphere decoding.

3.3.3 The sphere decoding algorithm

First, let us look into the sphere decoding algorithm mathematically. The lattice points Hs lie inside a hypersphere of radius d centered at y if and only if

$$d^2 \geq \|Hs - y\|_2^2. \tag{3.3}$$

It is useful to apply the QR decomposition to the matrix H

$$H = Q \begin{bmatrix} R \\ 0 \end{bmatrix},$$

where $Q \in \mathbb{R}^{n \times n}$ is orthogonal and $R \in \mathbb{R}^{m \times m}$ is upper triangular. Partition $Q = [Q_1, Q_2]$, where Q_1 is $n \times m$ and Q_2 is $n \times (n-m)$. We know

$$\|Hs - y\|_2^2 = \|Rs - Q_1^T y\|_2^2 + \|Q_2^T y\|_2^2,$$

so, the condition of (3.3) becomes

$$d^2 \geq \|Rs - Q_1^T y\|_2^2 + \|Q_2^T y\|_2^2,$$

and it can be written as

$$d^2 - \|Q_2^T y\|_2^2 \geq \|Rs - Q_1^T y\|_2^2. \tag{3.4}$$

Define $\hat{y} \triangleq Q_1^T y$ and $\hat{d}^2 \triangleq d^2 - \|Q_2^T y\|_2^2$, then the condition of (3.4) is

$$\hat{d}^2 \geq \|Rs - \hat{y}\|_2^2. \tag{3.5}$$

To make sphere decoding more understandable from mathematics, we investigate it mathematically from two approaches: recursive and non-recursive. And following each approach, a corresponding algorithm is presented for that approach.

- **Recursive approach** [21]

Assuming that we have applied QR decomposition to the original integer least squares problem (3.1), and it is reduced to problem (3.2), which is a triangular integer least squares problem now. So, sphere decoding needs to be applied to the problem (3.2) constrained in a hypersphere centered at $\hat{y}$ with radius $\hat{d}$, i.e., inequality (3.5).

Partition the upper triangular matrix R in the inequality (3.5)

$$R = \begin{bmatrix} R_1 & r_{1:m-1,m} \\ 0 & r_{m,m} \end{bmatrix},$$

where R_1 is the order $(m-1)$ leading principal submatrix of R, $r_{1:m-1,m}$ is the subvector consisting of the first $(m-1)$ entries of the last column of R, and $r_{m,m}$ is the last entry of the last column of R. Accordingly, also partition vector s and $\hat{y}$

$$s = \begin{bmatrix} s_{1:m-1} \\ s_m \end{bmatrix}, \qquad \hat{y} = \begin{bmatrix} \hat{y}_{1:m-1} \\ \hat{y}_m \end{bmatrix},$$

where $s_{1:m-1}$ and $\hat{y}_{1:m-1}$ are the first $(m-1)$ entries of s and $\hat{y}$ respectively, s_m and $\hat{y}_m$ are the last entries of s and $\hat{y}$ respectively as well. Then, we can get

$$\|Rs - \hat{y}\|_2^2 = \|R_1 s_{1:m-1} - (\hat{y}_{1:m-1} - r_{1:m-1,m} s_m)\|_2^2 + (r_{m,m} s_m - \hat{y}_m)^2,$$

i.e.,

$$\hat{d}^2 \geq \|R_1 s_{1:m-1} - (\hat{y}_{1:m-1} - r_{1:m-1,m} s_m)\|_2^2 + (r_{m,m} s_m - \hat{y}_m)^2.$$

Denote $y' = \hat{y}_{1:m-1} - r_{1:m-1,m} s_m$ and $d'^2 = \hat{d}^2 - (r_{m,m} s_m - \hat{y}_m)^2$, in order to satisfy inequality (3.5), it is necessary that

$$\hat{d} \geq |r_{m,m} s_m - \hat{y}_m| \tag{3.6}$$

and

$$d'^2 \geq \|R_1 s_{1:m-1} - y'\|_2^2 \tag{3.7}$$

The inequality (3.7) is similar to inequality (3.5), but it is to find the lattice points inside a hypersphere of radius d' for the triangular integer least squares problem (3.2) in dimension $m-1$.

The solution s_m for (3.6) are the integers between $(-\hat{d} + \hat{y}_m)/r_{m,m}$ and $(\hat{d} + \hat{y}_m)/r_{m,m}$, assuming $r_{m,m} > 0$[3], then

$$\left\lceil \frac{-\hat{d} + \hat{y}_m}{r_{m,m}} \right\rceil \leq s_m \leq \left\lfloor \frac{\hat{d} + \hat{y}_m}{r_{m,m}} \right\rfloor \tag{3.8}$$

Thus, for each integer s_m in (3.8), we compute a new search radius d' and a new y' by $d'^2 = \hat{d}^2 - (r_{m,m} s_m - \hat{y}_m)^2$ and $y' = \hat{y}_{1:m-1} - r_{1:m-1,m} s_m$, then solve the $(m-1)$-dimensional problem (3.7) following the similar pattern, i.e., recursively solving until we get to $m = 1$.

[3]For $r_{m,m} < 0$, it is similar, just switch the upper bound and lower bound of the inequality.

Table 3.1: Algorithm 1 - Recursive Sphere Decoding Algorithm

Inputs:	R, where R is upper triangular.
	$\hat{y}$, where $\hat{y}$ is already reduced by QR decomposition.
	r, where r is the hypersphere radius to the reduced problem.
Output:	s.
1.	$d \leftarrow r$;
2.	$UB \leftarrow \lfloor (d + \hat{y}_m)/r_{m,m}; \rfloor$;
3.	$LB \leftarrow \lceil (-d + \hat{y}_m)/r_{m,m}; \rceil$;
4.	**if** $m = 1$
5.	**return** the integers in $[LB, UB]$;
6.	**else**
7.	**for** each integer $s_m \in [LB, UB]$
8.	$d'^2 \leftarrow \hat{d}^2 - (r_{m,m}s_m - \hat{y}_m)^2$;
9.	$y' \leftarrow \hat{y}_{1:m-1} - r_{1:m-1,m}s_m$;
10.	apply this algorithm to $R_{1:m-1,1:m-1}$, y' and d';
11.	append s_m to each of the solutions of dimension $m-1$;
12.	**end**
13.	**end**
14.	**return** s;

With all the above discussion, now it is time to implement the recursive sphere decoding algorithm. A brief description of the recursive sphere decoding algorithm (Algorithm 1) is presented in Table 3.1.

Note, this algorithm finds all the lattice points inside the hypersphere of initial radius d, not only the lattice point closest to y. However, once we have all the lattice points inside the hypersphere, it is straightforward to find the one closest to y (in Euclidean sense), if it is exists. Actually, with tree representation of sphere decoding, this algorithm is a depth-first search method.

- **Non-recursive approach**

Due to the stack usage required by the recursive sphere decoding algorithm and

the limit of computer memory, in reality, we cannot decode a large dimensional space with the recursive sphere decoding algorithm, we need to discuss and implement a non-recursive version of sphere decoding algorithm.

Also assuming that QR decomposition is done for the original integer least squares problem (3.1), and sphere decoding is for Equation (3.5). With the description of the tree representation of sphere decoding, we can now look into a non-recursive sphere decoding algorithm.

We can also rewrite the condition of (3.5) in component wise as

$$\hat{d}^2 \geq \sum_{i=1}^{m} \left(\sum_{j=i}^{m} r_{i,j} s_j - \hat{y}_i \right)^2 , \tag{3.9}$$

where $r_{i,j}, j \geq i$ denotes the (i,j)th entry of upper triangular matrix R. With the upper triangular property of matrix R, the right hand side of inequality (3.9) can be expanded as

$$\begin{aligned} \hat{d}^2 \geq \quad & (\hat{y}_m - r_{m,m} s_m)^2 \\ & + (\hat{y}_{m-1} - r_{m-1,m} s_m - r_{m-1,m-1} s_{m-1})^2 \\ & + \cdots , \end{aligned} \tag{3.10}$$

where the first term in (3.10) depends only on the entry s_m of lattice point s, the second term in (3.10) depends on the entries $\{s_m, s_{m-1}\}$ of lattice point s, and so on.

We can see that a necessary condition for Hs lying inside the hypersphere of radius d is $\hat{d}^2 \geq (\hat{y}_m - r_{m,m} s_m)^2$, and it is equivalent for entry s_m to be within

the interval

$$\left\lceil \frac{-\hat{d}+\hat{y}_m}{r_{m,m}} \right\rceil \leq s_m \leq \left\lfloor \frac{\hat{d}+\hat{y}_m}{r_{m,m}} \right\rfloor . \tag{3.11}$$

Remark 3.1. *(3.8) and (3.11) are the same, i.e., solving s_m is the same in both the recursive approach and the non-recursive approach.*

Furthermore, for each s_m satisfying (3.11), define $\hat{d}^2_{m-1} \triangleq \hat{d}^2 - (\hat{y}_m - r_{m,m}s_m)^2$ and $\hat{y}'_{m-1} \triangleq \hat{y}_{m-1} - r_{m-1,m}s_m$, it is also necessary that $\hat{d}^2_{m-1} \geq (\hat{y}'_{m-1} - r_{m-1,m-1}s_{m-1})^2$, which leads to s_{m-1} being within the interval

$$\left\lceil \frac{-\hat{d}_{m-1}+\hat{y}'_{m-1}}{r_{m-1,m-1}} \right\rceil \leq s_{m-1} \leq \left\lfloor \frac{\hat{d}_{m-1}+\hat{y}'_{m-1}}{r_{m-1,m-1}} \right\rfloor . \tag{3.12}$$

We can follow the similar pattern to obtain the interval for s_{m-2}, s_{m-3} and so on until we obtain the interval for s_1, therefore, we are able to determine all the lattice points which are inside the hypersphere of radius d.

The implementation of the non-recursive sphere decoding algorithm (Algorithm 2) is presented in Table 3.2.

Note, in the implementation of the non-recursive sphere decoding algorithm, there are two issues need to be addressed:

- ☞ We start with $k = m$, decrease k until $k = 1$. But, it is possible for the computed inequalities (3.11), (3.12) and other similar ones that the entry $s_k, m \geq k \geq 1$ locates in an empty interval, that is, the lower bound of the interval is greater than the upper bound of the interval. In the case

Table 3.2: Algorithm 2 - Non-recursive Sphere Decoding Algorithm

Inputs:	R, where R is upper triangular.
	y, where y is already reduced by QR decomposition.
	r, where r is the hypersphere radius to the reduced problem.
Outputs:	s or *null*.
1.	$d_n \leftarrow r$;
2.	$UB_n \leftarrow \lfloor (d_n + y_n)/r_{n,n} \rfloor$, $LB_n \leftarrow \lceil (-d_n + y_n)/r_{n,n} \rceil$;
3.	$minR \leftarrow d_n$;
4.	$interSum_n \leftarrow y_n$;
5.	**if** $LB_n > UB_n$
6.	initial radius too small, **return** *null*;
7.	**else** $s_n \leftarrow LB_n$; **end**
8.	$k \leftarrow n$;
9.	**while** $s_n \leq UB_n$ {top level not exhausted}
10.	**if** $s_k > UB_k$ {this level exhausted already}
11.	$k \leftarrow k + 1$; $s_k \leftarrow s_k + 1$; {go back to upper level}
12.	**else** $\{s_k \leq UB_k\}$
13.	**if** $k > 1$
14.	$k \leftarrow k - 1$;
15.	$d_k^2 \leftarrow d_{k+1}^2 - (interSum_{k+1} - r_{k+1,k+1} * s_{k+1})^2$;
16.	$interSum_k = y_k - r_{k,k+1:n} * s_{k+1:n}$;
17.	$UB_k \leftarrow \lfloor (d_k + interSum_k)/r_{k,k} \rfloor$;
18.	$LB_k \leftarrow \lceil (-d_k + interSum_k)/r_{k,k} \rceil$;
19.	**if** $LB_k > UB_k$ {empty interval}
20.	$k \leftarrow k + 1$; $s_k \leftarrow s_k + 1$; {go back to upper level}
21.	**else** $s_k \leftarrow LB_k$; **end**
22.	**else** $\{k = 1\}$
23.	**while** $s_k \leq UB_k$ {go through all s_k at $k = 1$}
24.	**if** $minR > \|(Rs - y)\|_2$
25.	$found \leftarrow s$;
26.	$minR \leftarrow \|(Rs - y)\|_2$;
27.	**end**
28.	$s_k \leftarrow s_k + 1$;
29.	**end** {inner while}
30.	$k \leftarrow k + 1$; $s_k \leftarrow s_k + 1$; {go back to $k = 2$ level}
31.	**end** {if $k > 1$ else}
32.	**end** {if $s_k > UB_k$ else}
33.	**end** {outer while}
34.	**return** *found* or **return** *null*;

that the entry s_m locates in an empty interval, that means the initial radius d is selected too small, no lattice point can be determined at the very beginning decoding stage. In the case that the entry $s_k, 1 < k < m$ locates in an empty interval, we cannot go down further, instead, we have to go back to the upper level and pick another integer in this level's interval, then compute its lower level's interval (its children in the tree) again. In the case that the entry s_1 locates in an empty interval, we still have to go back to the upper lever. This is why the generated tree structure of sphere decoding is not a balanced tree (all the branches have the same lengths), some branches are probably shorter than the depth of the generated tree ($depth = m+1$). Because we cannot go down further in those branches, along those branches, the integers in some level's interval compute an empty interval in their lower level.

It is also possible that all the branches' lengths of the generated tree structure are less than $m + 1$, it means that from all the branches, the decoding process cannot reach the $k = 1$ level, that is, no lattice point can be determined. So we have to increase the initial radius d of the search hypersphere.

☞ Inside the hypersphere centered at y, there are many lattice points around the given y, however, we only need the one which is closest to y. Therefore, we have to search all the lattice points inside the hypersphere and compare their distance to y (in Euclidean sense) to determine whether a lattice point is the one closest to y or not. In order to determine the closest one, we have to reach the $k = 1$ level so that we are able to compute

the Euclidean distance. For the branches of the generated tree structure whose length equal to the tree depth ($length = depth = m + 1$), the decoding process has to go through all of them, compute and compare the Euclidean distance of the lattice points corresponding to those branches, therefore determine the closest lattice point.

From this two issues, we can see that the entire decoding process requires k to increase its value or decrease its value continually, i.e., go up level or down level continually but within different branches each time. Actually, it goes through the entire tree (except the root node) like a preorder traversal and performs a depth-first search.

3.3.4 Complexity of sphere decoding

In both the recursive and the non-recursive sphere decoding algorithms, they show that the number of lattice points inside the hypersphere can grow rapidly as the search radius increases. So the key issue for sphere decoding algorithm is the selection of the search radius. It is obvious when the initial radius is very large or as the initial radius $\hat{d} \longrightarrow \infty$ in the extreme case, sphere decoding has to search a very large hypersphere or almost the entire lattice space, which is infeasible and computationally prohibitive. On the other hand, when the initial radius is too small, there may be no lattice points inside the hypersphere. How to determine an appropriate initial radius? In Chapter 5, we will focus on the issue of initial radius and examine 2 different methods for the selection of initial radius.

Chapter 4

Adaptive Sphere Decoding

Usually, sphere decoding searches all the lattice points within a hypersphere, then amongst all of these points, finds the one which is closest to the given point. However, going through every point in the hypersphere is not very efficient, we have to remember the currently closest point and compare it with a new point to determine whether the new point is closer than the known one. This is an exhaustive search. And the original sphere decoding must spend much of its search time on the determination and comparison of the new points, although lots of new points are supposed to be not closer than the already known one.

In [14], the authors briefly mentioned an improved, more efficient method on sphere decoding - sphere decoding with radius update, which is able to avoid having to search all lattice points inside a hypersphere and potentially improves the search effort thereafter. However, the authors only introduced the idea without giving details. In this section, we get into this method and describe it in more details from two approaches. Before we start, we name this improved

sphere decoding method as "adaptive sphere decoding" (ASD), because this method can adapt to the radius whenever it has found a smaller radius and restart the search. We also refer sphere decoding sometimes as original sphere decoding in comparison with adaptive sphere decoding.

4.1 Geometry of adaptive sphere decoding

Geometrically, to solve (3.2), $\min_{s\in\mathbb{Z}^m} \|Rs - \hat{y}\|_2^2$, the original sphere decoding constructs a hypersphere originate from the vector $\hat{y}$ and with an initial radius (given) as its radius (see Figure 4.1), then the search algorithm has to go through all the lattice points inside this hypersphere in order to find the minimized solution for (3.2), if it exists.

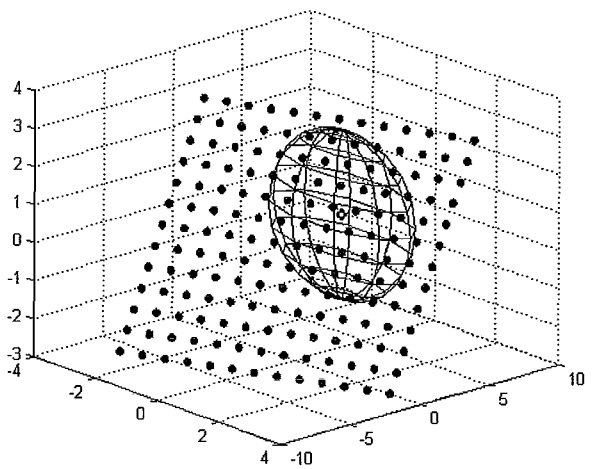

Figure 4.1: Geometric interpretation of a search hypersphere

Once an initial radius is determined and given to sphere decoding, we start the search. In the original sphere decoding, we start the search from $k = m$

level, if we are able to reach $k = 1$ level, then it is likely[1] that we can find a solution for (3.2), although it is not necessarily a closer one and is discarded if it is not. However, it is also possible that we cannot reach the $k = 1$ level at all, namely, no solution can be found, which means that the initial radius is too small.

Assuming that the initial radius we selected is appropriate, and we are able to find the solution to (3.2). We start the search, if we find a closer solution, then we remember it as the currently closest, and search all other unreached points again, no matter whether those points are further or closer than this one. We can imagine that some points are further than the currently closest one, but we still have to go through them. We know, for each lattice point, the search algorithm has to spend time on decoding it and computing its Euclidean distance to the center (the vector $\hat{y}$) as well as doing the comparison. Thus, the original sphere decoding algorithm is not very efficient, since there exist some points which have to be searched are further than the already known one and then are discarded right after reaching it. Generally speaking, when the initial radius is fairly large, many of the lattice points need to be searched and then discarded right away.

Now, let us look at adaptive sphere decoding, this method can improve the search techniques of the original sphere decoding. First of all, it still applies the hypersphere concept and decoding techniques. Before starting, the integer least squares problem (3.1) still needs to be reduced to a triangular integer least squares problem (3.2), then adaptive sphere decoding is applied to (3.2). It starts the search with an initial radius, however, unlike the original sphere decoding,

[1] A solution is not guaranteed to be found even if we have reached the $k = 1$ level.

once it has found a lattice point which is closer than the radius (suppose that the initial radius is large enough to find at least one lattice point inside the hypersphere), then it restarts the search immediately with the distance from this closer point to the center as a new radius. In this way, it does not have to go through all the lattice points inside the initial hypersphere. In other words, adaptive sphere decoding excludes searching the lattice points which are further than the already known one.

Geometrically, at the beginning of a search, adaptive sphere decoding constructs a hypersphere originate from the vector $\hat{y}$ and with an initial radius (given) as its radius (the outer sphere in Figure 4.2), and starts the search. Once it has found a closer point (suppose that the initial radius is large enough to find at least one lattice point inside the hypersphere), it constructs another smaller hypersphere originate from the same vector $\hat{y}$ and with the distance from this found, closer point to the vector $\hat{y}$ as its radius (the inner dense sphere in Figure 4.2). If it restarts the search with the smaller hypersphere, then the lattice points between the two hyperspheres are excluded from reaching, it means that we do not need to search those lattice points in between. How good is it? In general case, it is very likely that certain amount of lattice points are in between (the points between the outer and inner dense sphere in Figure 4.2) and the original sphere decoding has to search them and discard them right away[2], since apparently those points are not closer than the one with which the smaller hypersphere is constructed. Adaptive sphere decoding is able to eliminate the points in between rather than trying to search those points as the original sphere

[2]Although it is also possible that no lattice points in between in some cases.

decoding does.

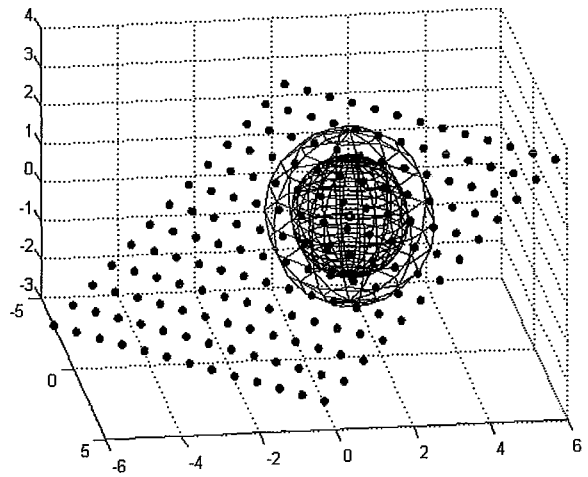

Figure 4.2: Geometric interpretation of adaptive sphere decoding

After restarting the search with the new smaller radius, adaptive sphere decoding follows the same rule, i.e., if a closer point is found, then it constructs another smaller hypersphere with the distance from this point to the center as its radius, and restarts the search. Applying the same pattern repeatedly, until we find the closest lattice point to the vector $\hat{y}$. Intuitively, adaptive sphere decoding can reduce the search radius rapidly and reach the closest lattice point dramatically, it may work very well even if the initial radius is fairly large, and many lattice points are inside the initial hypersphere.

4.2 Tree representation of adaptive sphere decoding

Let us look at adaptive sphere decoding from another view - tree representation. The original sphere decoding starts decoding from m-dimension down to 1-dimension, and works in component wise in determining lattice points. Once it has determined the possible kth ($1 \leq k \leq m$) entries of a lattice point within a hypersphere of radius d_k in a k-dimensional space and created an interval composed of the possible kth entries, then it starts from the lower bound of this interval to the upper bound, picks one kth entry, uses it to compute the radius d_{k-1} for $(k-1)$-dimensional space. Therefore, we are able to determine the possible $(k-1)$th entries of this lattice point within a hypersphere of radius d_{k-1} in the $(k-1)$-dimensional space, which follows from the kth entry picked in the k-dimension.

If we start from $k = m$ down to $k = 1$, this process constructs a tree of $depth = m + 1$ where the nodes in the kth level ($1 \leq k \leq m$) of the tree correspond to the kth entries of the lattice points. Any kth entry picked in k-dimensional space determines an interval in $(k-1)$-dimensional space which is composed of the possible $(k-1)$th entries. If represented in a tree, each node in kth level is used to compute its children in $(k-1)$th level. In this tree representation, the root node represents none entry of the lattice points from the point of view of our decoding[3], actually, the root node's children[4] represent

[3]So, in our figures of tree representation, the root node is empty, means that it is not an entry of the lattice points.

[4]Strictly speaking, the sphere decoding constructs a forest, the meaningless root node here conceptually makes the forest into a tree. Actually, the "root node's children" comprises the

the mth entries of the lattice points in m-dimensional space.

If the initial radius is large, then the mth level's interval determined by this radius is large, and there are many children for the root node, it is also more likely for each node in this tree (except the leaf nodes), there are many children for it. In other words, the constructed tree is denser, just as shown in Figure 4.3.

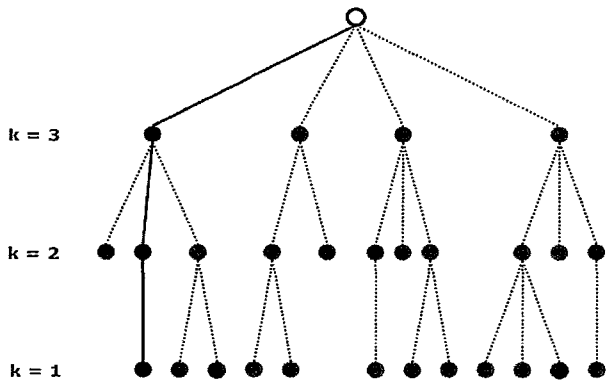

Figure 4.3: Dense tree representation generated by a sphere decoding with a large initial radius

For the original sphere decoding, it searches all the lattice points inside the hypersphere, if represented in a tree, it goes through each node of this tree (except the root node) like a preorder traversal and performs a depth-first search. We can imagine how much time is consumed if this tree is extremely dense and the depth of tree is large (depending on the size of the given matrix).

In this tree representation, if the initial radius is selected appropriately, it means that we are able to reach a leaf of full tree depth. Whenever we have

solution entry s_m. In the following context, we forget the meaningless root node and uses this term "root node's children" to refer the meaningful solution entry s_m.

reached a leaf of full tree depth[5], we have found a lattice point locating inside the hypersphere, and the nodes on the path from the root node's child to the leaf of full tree depth are the found lattice point's corresponding entries. However, it is not guaranteed that it is the closest lattice point (minimized solution for (3.2)). Therefore, we have to remember this lattice point and compare it with other lattice points which correspond to other paths from the root node's children to the leaves of full tree depth in order to find the closest one, and we always remember the currently closest lattice point and compare it with the remaining ones. That is, all the paths from the root node's children arriving at the leaves of full tree depth are all the lattice points locating inside the hypersphere, we have to find all of this kind of paths and compare the distances from their corresponding lattice points to the center (the vector $\hat{y}$), thus we are able to determine the closest lattice point amongst them.

If adaptive sphere decoding starts with the same initial radius as for the original sphere decoding, then the generated tree structure for both of them are the same. But for adaptive sphere decoding, once it reaches a leaf of full tree depth (for example, the dark path in Figure 4.3), the solid nodes on this path are the entries of a found lattice point inside the hypersphere, and we compute the Euclidean distance from this lattice point to the center as a new radius, then adaptive sphere decoding restarts with this new radius. The restarted adaptive sphere decoding constructs a new tree which is sparser (see Figure 4.4) than the constructed tree before (the Figure 4.3). Similarly, adaptive sphere decoding searches this new tree, once a path from root node's children to the leaves of full

[5]We only care about reaching the leaves which are at the $m+1$ depth of the tree, the other leaves whose length are less than $m+1$ are not the entries of lattice points.

tree depth (for example, the dark path in Figure 4.4) is found, we are able to compute another new radius and restart adaptive sphere decoding again, then another more sparser tree is constructed, and so on. We continue this process all the way down until the end (the termination of adaptive sphere decoding will be discussed in next section).

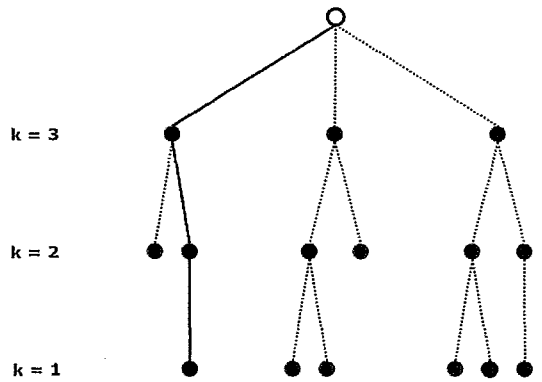

Figure 4.4: A sparse tree representation constructed by an adaptive sphere decoding because of adapting to a smaller radius

Unlike the original sphere decoding which constructs a huge dense tree only and searches this tree entirely, adaptive sphere decoding adapts to new radius and constructs several trees dynamically, and searches only part of those kind of trees. This reduces the overall search efforts, thus improves its total efficiency and performance[6].

[6]Adaptive sphere decoding does not guarantee an improvement, the performance may be the same for both decoding algorithms if the initial radius is exactly the same as the distance between the closest lattice point and the central point.

4.3 Termination of adaptive sphere decoding

For the original sphere decoding, it constructs a huge dense tree only, and goes through all the nodes (except the root node) to determine the closest lattice point. So whenever going through the entire tree is completed, we know that the sphere decoding can be terminated.

Adaptive sphere decoding constructs and searches many trees rather than only one huge dense tree, how does it terminate? From the algorithm of adaptive sphere decoding, simply speaking, if no new smaller radius can be computed, then it has found the closest lattice point (assuming it can). From the point of view of tree representation, adaptive sphere decoding constructs a sequence of trees, but the density of the trees in this sequence is usually reduced, that is, the branches of the trees become fewer and fewer (compare Figure 4.3 and Figure 4.4). If at a stage in this sequence, a very sparse tree is constructed and it happens that we have found the closest lattice point in this tree (see Figure 4.5). But adaptive sphere decoding does not know yet that it is the closest lattice point, it continues to restart again with the distance from this lattice point to the center as a new radius[7]. However, no lattice point can be found inside this hypersphere any more (rather than the one on its surface), that is, no new smaller radius can be computed and it does not need to restart again. So adaptive sphere decoding now knows that the lattice point found in previous tree is the closest one and returns it.

In conclusion, from the point of view of tree representation, there is no a

[7]From the point of view of tree representation, adaptive sphere decoding then construct a final tree, it is possible but not necessary that the final tree is identical as the previous tree.

single form of tree to indicate the termination of adaptive sphere decoding. In addition, the tree from which the closest lattice point is found is not the final constructed tree. When no new smaller radius can be computed, adaptive sphere decoding does not need to restart, it can be terminated. The lattice point found in the previous one of the final tree is the closest lattice point we desired.

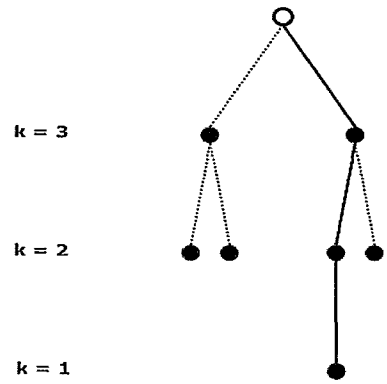

Figure 4.5: The constructed tree from which a closest lattice point is found by an adaptive sphere decoding, but it is not the final tree in a tree sequence constructed by adaptive sphere decoding

Remark 4.1. *In the case that the initial radius given to adaptive sphere decoding happens to be the distance from the closest lattice point to the center, then the adaptive sphere decoding algorithm (Algorithm 3) is not able to find the closest lattice point (the closest lattice point should be on the surface of this hypersphere), because neither is lattice point inside the hypersphere (rather than on its surface) nor can smaller radius be computed, and adaptive sphere decoding does not need to restart either. Since adaptive sphere decoding is designed to work with a relatively large initial radius, we do not take into account of this situation. If we really need to consider this situation, then it is not hard to modify the adaptive*

sphere decoding algorithm (Algorithm 3) a little bit to accommodate the case that the initial radius is the same as the distance from the closest lattice point to the center.

4.4 Adaptive sphere decoding algorithm

The entire adaptive sphere decoding algorithm is presented in Table 4.1. We may notice that it has only very little modification compared to the original sphere decoding algorithm (Algorithm 2), however, in most cases, its improvement of efficiency and performance is significant over the original one. The performance evaluation of the two algorithms will be shown in Chapter 7.

Table 4.1: Algorithm 3 - Adaptive Sphere Decoding Algorithm

Inputs:	R, where R is upper triangular.
	y, where y is already reduced by QR decomposition.
	r, where r is the sphere radius to the reduced problem.
Outputs:	s or *null*.
1.	$d_n \leftarrow r$;
2.	$UB_n \leftarrow \lfloor (d_n + y_n)/r_{n,n} \rfloor$, $LB_n \leftarrow \lceil (-d_n + y_n)/r_{n,n} \rceil$;
3.	$minR \leftarrow d_n$;
4.	$interSum_n \leftarrow y_n$;
5.	**if** $LB_n > UB_n$
6.	radius too small, **return** s or **return** *null*;
7.	**else** $s_n \leftarrow LB_n$; **end**
8.	$k \leftarrow n$;
9.	**while** $s_n \leq UB_n$ {top level not exhausted}
10.	**if** $s_k > UB_k$ {this level exhausted already}
11.	$k \leftarrow k + 1$; $s_k \leftarrow s_k + 1$; {go back to upper level}
12.	**else** $\{s_k \leq UB_k\}$
13.	**if** $k > 1$
14.	$k \leftarrow k - 1$;
15.	$d_k^2 \leftarrow d_{k+1}^2 - (interSum_{k+1} - r_{k+1,k+1} * s_{k+1})^2$;
16.	$interSum_k = y_k - r_{k,k+1:n} * s_{k+1:n}$;
17.	$UB_k \leftarrow \lfloor (d_k + interSum_k)/r_{k,k} \rfloor$;
18.	$LB_k \leftarrow \lceil (-d_k + interSum_k)/r_{k,k} \rceil$;
19.	**if** $LB_k > UB_k$ {empty interval}
20.	$k \leftarrow k + 1$; $s_k \leftarrow s_k + 1$; {go back to upper level}
21.	**else** $s_k \leftarrow LB_k$; **end**
22.	**else** $\{k = 1\}$
23.	**while** $s_k \leq UB_k$ {go through all s_k at $k = 1$}
24.	**if** $minR > \|(Rs - y)\|_2$
25.	$found \leftarrow s$;
26.	$r \leftarrow \|Rs - y\|_2$;
27.	**goto 1;** {start over with the new r}
28.	**end**
29.	$s_k \leftarrow s_k + 1$;
30.	**end** {inner while}
31.	$k \leftarrow k + 1$; $s_k \leftarrow s_k + 1$; {go back to $k = 2$ level}
32.	**end** {if $k > 1$ else}
33.	**end** {if $s_k > UB_k$ else}
34.	**end** {outer while}
35.	**return** *found* or **return** *null*;

Chapter 5

The Selection of Radius

The basic idea for both sphere decoding methods is the same and quite simple: compute an initial radius, construct a hypersphere centered at the given vector with this initial radius, then search all the lattice points inside this hypersphere to determine the closest one. However, how to select the initial radius is not clear and this is a major factor of computational complexity for both sphere decoding methods. As mentioned in previous chapters, if the initial radius is selected too large, then too many lattice points are contained in the hypersphere; whereas if the initial radius is selected too small, then there are no lattice points lying inside the hypersphere. We will discuss the issue of radius selection in this chapter.

5.1 Statistical selection of initial radius

If sphere decoding is applied to solve the communication system (2.1) which models the frequency-selective channel, then the statistical characteristics of noise added to the signal can be considered and exploited to compute the initial

radius [1]. This method is based on the assumption that the received vector y cannot be arbitrary, rather, it is a lattice point perturbed by additive noise with known statistical properties. Since this method takes into account of the statistical properties of the additive noise included in the received vector when computing the initial radius, it is only suitable for this particular frequency-selective channel model when using sphere decoding to find the minimizer for the equivalent integer least squares problem of this model.

In the application as modeled in (2.1), H is a matrix representing a communication channel and $n = m + l - 1$ where l is the order of the channel and its entries are assumed to be Gaussian $\mathcal{N}(0, 1)$, v is the additive white noise vector, whose entries are independent and identically distributed Gaussian noise with mean 0 and variance σ^2 ($\mathcal{N}(0, \sigma^2)$). Furthermore, y is the received signal vector, s is the transmitted signal vector. So when y is received after transmitting through this frequency-selective channel, the source signal s can be determined by applying sphere decoding to solve the equivalent integer least squares problem (3.1) of this model. Note, from (2.1), we get

$$\|v\|_2^2 = \|Hs - y\|_2^2, \tag{5.1}$$

then

$$\frac{1}{\sigma^2}\|v\|_2^2 = \frac{1}{\sigma^2}\|Hs - y\|_2^2. \tag{5.2}$$

It is a $\mathcal{X}^2$ random variable with n degrees of freedom. Thus the initial radius r

for sphere decoding may be computed as a scaled variance of the noise

$$r^2 = \alpha n \sigma^2, \tag{5.3}$$

where α can be solved from the following probability density function

$$p = \int_0^{\frac{n\alpha}{2}} \frac{\lambda^{\frac{n}{2}-1}}{\Gamma(\frac{n}{2})} e^{-\lambda} \mathrm{d}\lambda, \tag{5.4}$$

where the integrand is the probability density function of the $\mathcal{X}^2$ random variable with n degrees of freedom, p is its probability, usually a high probability and is set to a value close to 1. For example, set $p = 0.99$, then solve α from (5.4) and compute the radius r from (5.3). If with this computed radius, the sphere decoding cannot find the closest lattice point, then the probability in (5.4) should be increased to adjust the radius in (5.3) and search the larger hypersphere, until the closest lattice point is found. In such a way, the initial radius r may be selected appropriately for this specific communication channel model and the closest lattice point may be found with a high probability. The brief algorithm of statistical selection of initial radius is presented in Table 5.1.

Remark 5.1. *As mentioned in [1], this method only takes into account of the statistical properties of the noise, under some assumptions for the channel matrix H, because [22] and [23] show that selecting the radius based on the noise mitigates the computational complexity. So, statistical selecting radius solely depends on the probability p, the noise variance σ^2 and the dimension n. If the channel matrix H is not ideal and should be taken into account as well, then it is an NP-hard problem, and this statistical method cannot be utilized any more.*

Table 5.1: Algorithm 4 - Algorithm of statistical selection of initial radius

Inputs:	variance σ^2 of the random noise.
	initial probability p (a value close to 1).
Output:	the initial radius r of the hypersphere.
Dependency:	algorithm 2 in Table 3.2.
1.	solve α in probability density function (5.4);
2.	compute r from (5.3);
3.	sphere decoding with r as initial radius;
4.	**if** no solution found in sphere decoding
5.	increase the probability p;
6.	**goto 1;**
7.	**else**
8.	**return** r**;**
9.	**end**

5.2 Deterministic selection of initial radius

The above statistical method of computing the initial radius is based on the assumption that the channel matrix H is ideal. If we also need to take the characteristics of the channel matrix H into account, then in [21], Qiao proposed a deterministic method in computing the initial radius. This method is also proposed for communication applications. It assumes that the norm of Hs cannot be too large due to the power constraint of the channel. This means that when the channel matrix H is applied to the source signal s, it does not magnify the length of the source signal vector too much. Based on this assumption, the initial radius can be computed by the following deterministic method.

Suppose that QR decomposition is already applied to (3.1) and it is reduced to (3.2), now we will find the initial radius for sphere decoding by the following deterministic method. First, we need to solve the triangular system $Rs = \hat{y}$ by

whatever means to obtain s, $s \in \mathbb{R}^m$, which is the real least squares solution to the problem $\min \|Hs - y\|_2^2$. For example, s can be solved by

$$s = R^{-1}\hat{y}, \tag{5.5}$$

where R^{-1} is the inverse of matrix R. Since the entries of s are real, round them off to their nearest integers to obtain the lattice point $\hat{s}$, $\hat{s} \in \mathbb{Z}^m$

$$\hat{s} = \lceil s \rfloor. \tag{5.6}$$

Thus the initial radius r can be computed by setting it to the distance between $R\hat{s}$ and the vector $\hat{y}$ (in Euclidean sense)

$$r = \|R\hat{s} - \hat{y}\|_2. \tag{5.7}$$

The vector $\hat{s}$ is called the Babai estimate [5]. In communication applications, this procedure is often referred to as Zero-Forcing (ZF) equalization.

It is clear that the hypersphere of radius r contains at least one lattice point, namely $\hat{s}$. It is the lattice point closest to the real least squares solution. If sphere decoding starts with this initial radius r, it gets at least one lattice point which lies on the surface of the hypersphere and this lattice point is also very likely to be the closest one to the vector $\hat{y}$.

If the real least squares solution s happens to be an integer vector, that is, $\hat{s} = s$, then the radius r is zero, means that no sphere decoding is required, s is already the integer least squares solution. In communication applications, this

Table 5.2: Algorithm 5 - Algorithm of deterministic selection of initial radius

Inputs:	R, where R is upper triangular.
	$\hat{y}$, where $\hat{y}$ is the y reduced by QR decomposition.
Output:	the initial radius r of the hypersphere.
1.	solve s, $s \in \mathbb{R}^m$ in triangular system $Rs = \hat{y}$;
2.	round: $\hat{s} = \lceil s \rfloor$, $\hat{s} \in \mathbb{Z}^m$;
3.	set $r = \|R\hat{s} - \hat{y}\|_2$;

situation implies that both channel and signal are perfect, no channel distortion on transmitted signal and no additive noise to transmitted signal either, which is only possible in theory, not in practice.

The brief algorithm of deterministic selection of initial radius r is presented in Table 5.2.

The size of r is also examined in [21]. Let $f = \hat{s} - s$, then from (5.7), we have

$$\begin{aligned} r &= \|R\hat{s} - \hat{y}\|_2 \\ &= \|R(s+f) - \hat{y}\|_2 \\ &= \|Rf\|_2. \end{aligned}$$

We know $f = \hat{s} - s = \lceil s \rfloor - s$, $s \in \mathbb{R}^m$, then $\|f\|_2 \leq \sqrt{m}/2$, thus from above, we get

$$\begin{aligned} r &= \|Rf\|_2 \\ &\leq \frac{\sqrt{m}}{2}\|R\|_2 \\ &= \frac{\sqrt{m}}{2}\|H\|_2. \end{aligned}$$

The above derivation shows that the size of radius depends on the norm of the channel matrix H as well as the size m of the channel matrix H. In communication applications, under the assumption of channel power constraint, the norm of channel matrix, $\|H\|_2$, cannot be too large. In addition, the size of the channel matrix H depends on the number of transmitting antennae and the number of receiving antennae, which cannot be too large either. Thus the radius is expected to be moderate.

Remark 5.2. *This method of computing the initial radius is for communication applications only. As mentioned in [21], "We must emphasize that this method for finding a search radius is for applications like communications where the signal to noise ratio is relatively high, that is, the noise variance is relatively small." When the SNR is high, the deterministic method gives a small radius. In general case (low SNR), this deterministic method still works, only the radius can be large.*

5.3 Adaptively updating radius

For adaptively updating radius, there are three issues which need to be addressed.

- An initial radius must be given to adaptive sphere decoding

  Although adaptive sphere decoding is able to adapt itself to the new smaller radius when searching a hypersphere, the initial radius must be given to it first. Unlike the statistical selection of initial radius or the deterministic selection of initial radius, the initial radius is computed by them, adaptive sphere decoding does not compute the initial radius and cannot start sphere

decoding until an appropriate initial radius is given to it. But what is an appropriate initial radius for adaptive sphere decoding, this still remains an unknown issue. From the characteristics of adaptive sphere decoding, this initial radius could be relatively large, since adaptive sphere decoding is able to reduce radius very rapidly. But, how large is relatively large for the initial radius? Theoretically, adaptive sphere decoding is not able to search the entire lattice space, i.e., the initial radius cannot be arbitrarily large.

- Restarting sphere decoding may be costly

  Whenever adaptive sphere decoding restart, there is an associated cost to the restarting process. We have to consider whether it is worth restarting sphere decoding or not. If the initial radius is already relatively small, such as the radius computed by the deterministic method, there are not too many lattice points inside the hypersphere. Although adaptive sphere decoding starts with a relatively small initial radius, the overall restarting cost may be more than the cost of going through all the lattice points inside the hypersphere. So, applying adaptive sphere decoding is not an very good idea in this kind of situation.

  As pointed out in [14], in communication applications, when the SNR in a channel is low (the noise variance is relatively high), the number of lattice points inside an initial hypersphere are relatively large, and adaptive sphere decoding is quite useful. But, when the SNR in a channel is high (the noise variance is relatively low), restarting sphere decoding may be costly, then

it is not beneficial to apply adaptive sphere decoding.

- Not only restricting on communication applications

  Computing the initial radius from both the statistical method and the deterministic method is based on certain assumptions for communication applications. But for adaptive sphere decoding, there is no such assumptions and no particular application restriction, it does not restrict itself only on communication applications. The initial radius does not need to be relatively small. Adaptive sphere decoding may be useful for certain applications in which the initial radius cannot be computed very small. In particular, it is more beneficial to apply adaptive sphere decoding if the initial hypersphere is relatively large and the density of lattice points inside the hypersphere is large as well.

Chapter 6

Error Analysis of Radius Computation

6.1 Computational error

In the deterministic method, the initial radius r is computed by setting it equal to the distance between $R\hat{s}$ and the vector $\hat{y}$ (in Euclidean sense) where $\hat{s}$ is the integer vector closest to the real least squares solution vector s, and the hypersphere to be decoded is actually determined by $\|Rs - \hat{y}\| \leq r$. We know the initial radius computed by this method can be small, we do not know how many lattice points are contained in this hypersphere. Anyway, it is obvious that at least one lattice point, namely $\hat{s}$, is on the surface of the hypersphere, this hypersphere cannot be empty. We may expect that with the initial radius computed by this method, the sphere decoding definitely finds a closest lattice point to the vector y.

The above conclusion seems to be correct, but in practice, in the presence of inexact arithmetic due to rounding error in floating-point computation, the computed radius may not be the exact radius, rounding error is introduced into this computed radius. If the initial radius computed by this method is relatively small and the closest lattice point is on the surface of the hypersphere of exact radius, then it requires that the computed radius is at least not smaller than the exact radius. However, this is not always true. The introduced error also makes it possible that the computed radius is smaller than the exact radius, thus it is possible that sphere decoding finds no lattice point. We will examine this in the following example.

Example 1:

This example is done in MATLAB and using double precision. In (3.1), let H and y be

$$H = \begin{bmatrix} 2 & -1 & -1 \\ -1 & 0 & -2 \\ -1 & -1 & -1 \end{bmatrix}, \qquad y = \begin{bmatrix} -1 \\ 1 \\ 0 \end{bmatrix}$$

respectively. After QR decomposition is applied to (3.1), it is reduced to (3.2), then we get R and $\hat{y}$

$$R \approx \begin{bmatrix} -2.4495 & 0.4082 & -0.4082 \\ 0 & 1.3540 & 1.6002 \\ 0 & 0 & 1.8091 \end{bmatrix}, \qquad \hat{y} \approx \begin{bmatrix} 1.2247 \\ 0.3693 \\ -0.6030 \end{bmatrix}$$

respectively. If we solve the real solution s, $s \in \mathbb{R}^m$ of the triangular system $Rs = \hat{y}$, we get

$$s \approx \begin{bmatrix} -0.3333 \\ 0.6667 \\ -0.3333 \end{bmatrix},$$

and round the entries of s to their nearest integers, then we get an integer vector $\hat{s}$, $\hat{s} \in \mathbb{Z}^m$

$$\hat{s} = \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix}.$$

Therefore the initial radius r can be computed from (5.7)

$$\tilde{r} \approx 1.4142,$$

where $\tilde{r}$ denotes that this radius is a computed result, not the exact radius.

Now, we search the closest lattice point to the vector y inside the hypersphere of this initial radius and centered at y with the sphere decoding algorithm (the Algorithm 2 in Table 3.2), then surprisingly, no closest lattice point can be found, this initial radius is too small to find a lattice point. What happened? It is against our expectation that at least one lattice point which is on the surface of the hypersphere should be found.

As we mentioned earlier, during the floating-point computation process, rounding error is introduced. In this example, the exact radius should be exactly $\sqrt{2}$, however, the radius computed by (5.7) is approximately equal to $\sqrt{2}$, specifically, the difference $\tilde{r} - \sqrt{2} \approx -2.2204 \times 10^{-016}$, which is negative. So, this example shows that the introduced rounding error makes the computed radius

slightly smaller than the exact radius.

If the error makes the computed radius greater than the exact radius (e.g. on the right hand side of the exact radius in Figure 6.1), that is fine, it makes the hypersphere a little bit larger, we can find the closest lattice point anyway. However, if the computed radius happens to be smaller than the exact radius (e.g. on the left hand side of the exact radius in Figure 6.1), then it is possible that there are no lattice points inside the hypersphere.

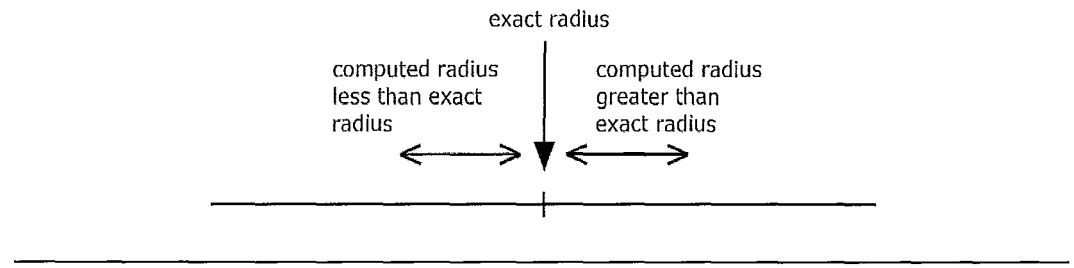

Figure 6.1: Computed radius versus exact radius

Since for any floating-point computation, the rounding errors are inevitable and must be introduced. How to make sure the radius computed by the deterministic method to be able to find at least one closest lattice point? The basic idea is to find a bound of the magnitude of the rounding error for this computed radius and compensate the computed radius with this error bound, then we ensure that the radius computed by the deterministic method, after compensation, does not make sphere decoding fail again.

6.2 Formulas of error analysis

In floating-point arithmetic, the purpose of rounding error analysis is to find a bound of rounding error with some appropriate measures of the effects of rounding errors for an algorithm. It is better to put error bounds in a concise, easily interpreted form if we can. However, whether an error bound exists is dependent on the given problem data set. Ideally, the error bound is small and achievable for all sets of problem data, but if not, the error analysis should investigate the characteristics of an algorithm that raise any potential instability and therefore provide a solution on how the instability of an algorithm can be avoided and revised.

Usually, there are two categories when doing an error analysis, forward error analysis and backward error analysis.

- Forward error analysis

  If an algorithm takes x as an input and produces y as an output, this algorithm can be considered as a function $y = f(x)$. We denote $\tilde{y}$ as the computed result from this algorithm, then the absolute error $|y - \tilde{y}|$ or the relative error $|y - \tilde{y}|/|y|$ is called *forward error*, the process of finding this forward error is called *forward error analysis*.

- Backward error analysis

  Alternatively, the computed result $\tilde{y}$ may be the exact result of a slightly perturbed input $x + \Delta x$, that is, $\tilde{y} = f(x + \Delta x)$. Generally speaking, there may be many such Δx, but we are only interested in a minimal one of those Δx and a bound for $|\Delta x|$. This bound, possibly divided by $|x|$, is

called *backward error*, the process of finding this backward error is called *backward error analysis.*

6.2.1 Error analysis of matrix multiplication

The error analysis of matrix multiplication can be found in many textbooks, the following is just a restatement from [24].

The error of matrix-vector or matrix-matrix multiplication depends on the error of inner products since the multiplication is actually the inner products of their corresponding entries. Let $A \in \mathbb{R}^{n\times m}$, $x \in \mathbb{R}^m$ and matrix-vector multiplication be $y = Ax$. The vector y can be formed as n inner products, $y_i = a_{i,1..m}x$, $1 \leq i \leq n$, where y_i is the ith entry of vector y and $a_{i,1..m}$ is the ith row of A. Denote the computed result for y as $\tilde{y}$ and y_i as $\tilde{y}_i$, then from the error analysis of inner product [24], we have

$$\tilde{y}_i = (a_{i,1..m} + \Delta a_{i,1..m})x, \qquad |\Delta a_{i,j}| \leq \gamma_m |a_{i,j}|,\ 1 \leq i \leq n,\ 1 \leq j \leq m,$$

where

$$\gamma_m = \frac{m\mu}{1 - m\mu}$$

and μ is the unit of roundoff[1].

Thus, the backward error of matrix-vector multiplication is

$$\tilde{y} = (A + \Delta A)x, \qquad |\Delta A| \leq \gamma_m |A|,$$

[1]Here we assume that the computer system complies with the IEEE floating-point standards.

where $|A|$ is the absolute value of its each entry, i.e., $|A| = [|a_{i,j}|]$, $1 \leq i \leq n$ and $1 \leq j \leq m$. The above implies the bound of the component wise forward error of matrix-vector multiplication is

$$|y - \tilde{y}| \leq \gamma_m |A||x|.$$

And the bound of 2-norm forward error of matrix-vector multiplication is

$$\|y - \tilde{y}\|_2 \leq \sqrt{min(n,m)}\gamma_m \|A\|_2 \|x\|_2. \tag{6.1}$$

6.2.2 Error analysis of vector addition

The forward error of vector-vector addition is pretty straightforward. Let $x, y, z \in \mathbb{R}^m$ and the vector-vector addition be $z = x + y$. Denote the computed result for z is $\tilde{z}$. Thus, the bound of forward error of vector-vector addition is

$$|z - \tilde{z}| \leq \mu |\tilde{z}|. \tag{6.2}$$

6.2.3 Error analysis of deterministic radius

Now, we perform a forward error analysis to derive an error bound of the computed radius for the deterministic radius selection algorithm. In this deterministic method, we round the real least squares solution s to its nearest integer vector $\hat{s}$ and then the radius is computed by $\|R\hat{s} - \hat{y}\|_2$. We know the rounding error has to be introduced in computing $\|R\hat{s} - \hat{y}\|_2$. Actually, the error is the combinations of the rounding errors introduced by the computations of matrix-

vector multiplication, vector-vector addition and the 2-norm. In order to find an error bound of the deterministic radius, it remains to apply the formulas of matrix-vector multiplication and vector-vector addition as well as the 2-norm to the computation of the deterministic radius. Denote the computed radius by the deterministic method as $\tilde{r}$, and we are trying to find the error bound $|r - \tilde{r}|$.

Let $u = R\hat{s}$ and $\tilde{u}$ be the computed $R\hat{s}$. By (6.1), the bound of forward error of matrix-vector multiplication, shows

$$\|u - \tilde{u}\|_2 \leq \gamma_m \sqrt{m} \|R\|_2 \|\hat{s}\|_2, \tag{6.3}$$

where

$$\gamma_m = \frac{m\mu}{1 - m\mu}$$

and μ is the unit of roundoff. By (6.3), the error in the computed radius we are looking for is

$$\begin{aligned} |r - \tilde{r}| &= |\,\|R\hat{s} - \hat{y}\|_2 - \tilde{r}| \\ &= |\,\|u - \tilde{u} + \tilde{u} - \hat{y}\|_2 - \tilde{r}| \\ &\leq \|u - \tilde{u}\|_2 + |\,\|\tilde{u} - \hat{y}\|_2 - \tilde{r}| \\ &\leq \gamma_m \sqrt{m}\, \|R\|_2 \|\hat{s}\|_2 + |\,\|\tilde{u} - \hat{y}\|_2 - \tilde{r}|. \end{aligned} \tag{6.4}$$

The term $|\,\|\tilde{u} - \hat{y}\|_2 - \tilde{r}|$ is the computational error in computing the vector difference $\tilde{u} - \hat{y}$ and then 2-norm of the computed $\tilde{u} - \hat{y}$. Denote $\tilde{z}$ as the computed $\tilde{u} - \hat{y}$, then $\tilde{r}$ is the computed $\|\tilde{z}\|_2$. The 2-norm $\|\tilde{z}\|_2$ is the square

root of the inner product, thus from (6.1), we get

$$| \|\tilde{z}\|_2 - \tilde{r}| \leq \gamma_m \tilde{r}. \tag{6.5}$$

For the computation of the vector subtraction, by (6.2), we have the component wise forward error

$$|\tilde{z} - (\tilde{u} - \hat{y})| \leq \mu |\tilde{z}|$$

and the 2-norm forward error

$$\|\tilde{z} - (\tilde{u} - \hat{y})\|_2 \leq \mu \|\tilde{z}\|_2. \tag{6.6}$$

Combining (6.5) and (6.6), the forward error analysis of vector-vector addition and inner product gives

$$\begin{aligned} | \|\tilde{u} - \hat{y}\|_2 - \tilde{r}| &\leq | \|(\tilde{u} - \hat{y}) - \tilde{z}\|_2 + \|\tilde{z}\|_2 - \tilde{r}| \\ &\leq \|(\tilde{u} - \hat{y}) - \tilde{z}\|_2 + | \|\tilde{z}\|_2 - \tilde{r}| \\ &\leq \mu \|\tilde{z}\|_2 + \gamma_m \tilde{r}. \end{aligned}$$

Thus, from (6.4), we have the error bound of the computed radius for the deterministic method, which is

$$|r - \tilde{r}| \leq \gamma_m \sqrt{m} \, \|R\|_2 \|\hat{s}\|_2 + \mu \|\tilde{z}\|_2 + \gamma_m \tilde{r}. \tag{6.7}$$

The above formula gives an upper bound of rounding error in the computation of deterministic radius $\tilde{r}$, which can be used to find an upper bound of the exact

radius r.

6.3 Revised algorithm of deterministic method

For the deterministic method, we have already got the upper bound of rounding error of computed radius, by applying the error bound (6.7), we are able to take the computational error into account when deterministically computing the initial radius and get an upper bound of the exact radius r.

The μ in γ_m is the unit of roundoff. In single precision $\mu \approx 10^{-7}$ and in double precision $\mu \approx 10^{-16}$, therefore we can assume in most cases that

$$m\mu < \frac{1}{2}.$$

Thus we have

$$\gamma_m < 2m\mu,$$

since

$$1 - m\mu > \frac{1}{2}.$$

Also note that

$$\|R\|_2 = \|H\|_2.$$

Then neglecting the term $\mu\|\tilde{z}\|_2$ in (6.7), it follows

$$r \leq \tilde{r} + 2m(\sqrt{m}\,\|H\|_2\|\hat{s}\|_2 + \tilde{r})\mu. \tag{6.8}$$

The above inequality gives an upper bound of the exact radius r, in terms of

Table 6.1: Algorithm 6 - Algorithm of revised deterministic selection of initial radius

Inputs:	R, where R is upper triangular.
	$\hat{y}$, where $\hat{y}$ is the y reduced by QR decomposition.
Output:	the initial radius r of the hypersphere.
1.	solve s, $s \in \mathbb{R}^m$ in triangular system $Rs = \hat{y}$;
2.	round: $\hat{s} = \lceil s \rfloor$, $\hat{s} \in \mathbb{Z}^m$;
3.	compute $\tilde{r} = \|R\hat{s} - \hat{y}\|_2$;
4.	set $r = \tilde{r} + 2m(\sqrt{m}\,\|H\|_2\|\hat{s}\|_2 + \tilde{r})\mu$;

the computed radius $\tilde{r}$, the 2-norm of the channel matrix H and the 2-norm of the integer vector $\hat{s}$. By using the bound (6.8), we present a revised algorithm of deterministic selection of initial radius (Algorithm 6) in Table 6.1. This revised algorithm incorporates the rounding error into the computation of the initial radius for a search hypersphere.

Example 2:

Following the Example 1, we have already computed $\tilde{r} \approx 1.4142$, which made sphere decoding fail to find the closest lattice point. Now let us apply the Algorithm 6 to get the upper bound of the exact radius r. In MATLAB and using double precision $\mu \approx 10^{-16}$,

$$\|H\|_2 \approx 2.6762, \qquad \|\hat{s}\|_2 = 1.$$

By (6.8), the upper bound of the exact radius

$$r \approx 1.4142 + 3.6298 \times 10^{-015},$$

and the difference $r - \sqrt{2} \approx 3.3307 \times 10^{-015}$, it is positive now, which means that the upper bound of the exact radius r is slightly greater than the exact radius, and sphere decoding should not fail any more. So we get the closest lattice point by sphere decoding with this upper bound of the exact radius as the initial radius of the search hypersphere, which is $[0\ 1\ 0]^T$.

Chapter 7

Numerical Simulation and Experiment

In this chapter, we conduct a couple of experiments, they target different purposes. The first experiment tests the improvement of performance provided by the adaptive sphere decoding over the original sphere decoding. In the second experiment, we see what the statistically computed radius and deterministically computed radius are, this experiment focuses on the simulated communication channel. The third experiment examines the failure rate of sphere decoding with the deterministically computed radius, then what is the failure rate after using the revised deterministic algorithm (Algorithm 6) to compute radius. From the last experiment, we see the performance evaluation of sphere decoding between the radius computed by the deterministic method and the radius computed by the revised deterministic method, this experiment shows that the radius computed by the revised deterministic method does not cause a significant overhead

for sphere decoding.

7.1 Experiment setup

For all experiments and simulations, the environment is MATLAB and uses double precision. A specific hardware configuration and software version is not a matter for these experiments. Although we need to record programs' running time sometimes and take the running time as measurements, the measurements of running time of programs are based on the same platform, the same platform has the same affects on both measured parties. The sphere decoding program is implemented in MATLAB code according to Algorithm 2, and the adaptive sphere decoding program is implemented in MATLAB code as well according to Algorithm 3. If an experiment does not involve a simulation of a particular application, then both matrix and vector given to a program are generated randomly by the MATLAB build-in function "*rand*", i.e., the entries of the input matrix H and the entries of the input vector y are pseudo random values drawn from the standard uniform distribution on the open interval $(0, 1)$.

For both sphere decoding and adaptive sphere decoding, the matrix given to them is already reduced to upper triangular by QR decomposition, and QR decomposition is applied to the given vector as well, i.e., the actual inputs to both decoding programs are the triangular integer least squares problem (3.2). Similarly, the initial hypersphere radius given to both decoding programs are computed from the triangular integer least squares problem (3.2) too. The QR decomposition we use is the MATLAB build-in function "*qr*".

Since we usually do not compare directly two real values in numerical implementation, what needs to be pointed out is when we really need to compare two Euclidean distances in both sphere decoding algorithm and adaptive sphere decoding algorithm, i.e., "$minR > \|(Rs - y)\|_2$" (line 24 in both Algorithm 2 and Algorithm 3), we replaced this line with "$minR - \|(Rs - y)\|_2 > \mu$" when implementing this two algorithms in MATLAB code, where μ is set to $\mu = 10^{-7}$. We know that a real value may not be exactly represented by floating-point, two reals may be considered as equal in floating-point as long as the difference between this two reals is smaller than a certain small value such as 10^{-7}.

In adaptive sphere decoding algorithm, there is a *goto* statement, but there is no *goto* statement provided in MATLAB, so we cannot translate our algorithm into a MATLAB code directly. Alternatively, we implement the *goto* statement in the adaptive sphere decoding algorithm with the *while* loop combined with some flags in MATLAB code.

7.2 Sphere decoding versus adaptive sphere decoding

In this section, we conduct an experiment which evaluates the performance improvement provided by adaptive sphere decoding over the original sphere decoding. We take the number of iterations of a program as a measurement rather than measuring the running time of a program, since measuring running time is not very accurate even if both programs run on the same platform. We know that the number of iterations may be considered as the complexity of an algorithm.

In order to measure the number of iterations, we simply put a iteration counter right after each *while* statement[1] in both original sphere decoding and adaptive sphere decoding. Although there are nested *while* loops in both algorithms, we just count the total number of iterations. Since the two algorithms are very similar, they have only very little differences, and the *while* loop structure is similar in both algorithms too, this makes it feasible for us to pursue the performance evaluation for both algorithms based on measuring the number of iterations.

As pointed out in Chapter 5, adaptive sphere decoding is able to adapt itself to the new smaller radius, but the initial radius to start sphere decoding is still not told yet by adaptive sphere decoding, the same as to the original sphere decoding. For this experiment, a radius is computed by the deterministic method as a base radius, then the initial radius given to both algorithms is the various magnifications of this base radius, so that we may evaluate both algorithms' performance for different given initial radii.

Experiment 1:

In this experiment, we see the overall performance evaluation for both algorithms from the total number of iterations, and compare the improvement provided by adaptive sphere decoding over the original sphere decoding. For a comparison, the size of matrix and the initial radius given to both algorithms are the same so that the comparison is based on the same criteria, and a couple of comparisons are for various matrix sizes and various initial radii. In addition, for a matrix size and an initial radius, we run both algorithms for 100 times, then

[1]Just after entering a *while* loop, we count it as one iteration.

Table 7.1: Number of iterations by sphere decoding (SD) versus adaptive sphere decoding (ASD) with various initial radii

Size of H	Base computed radius (r)	$2 \times r$ (#)		$3 \times r$ (#)		$4 \times r$ (#)	
		SD	ASD	SD	ASD	SD	ASD
3×3	0.412051	1244	196	3533	314	7683	436
4×4	0.606766	3313	332	13134	539	36861	778
5×5	0.724504	36525	797	229828	1350	885205	1973
4×3	0.51985	79	37	214	59	457	85
5×4	0.610472	572	113	2269	195	6383	285
6×5	0.846448	9869	350	62212	622	239821	950

there are 100 sets of matrix and vector. We take the average of total number of iterations for this 100 runs, thus it is more accurate for the evaluation of the overall performance, and reduces the random factor of a single run. The result of this experiment is in Table 7.1.

From Table 7.1, roughly speaking, the complexity of sphere decoding is almost exponential to both the size of matrix and the initial radius. But, the adaptive sphere decoding is able to reduce the complexity of the search. Furthermore, the larger the size of matrix is, the more performance improvement adaptive sphere decoding provides; the larger the initial radius is, the more performance improvement adaptive sphere decoding provides as well.

Remark 7.1. *Although adaptive sphere decoding is able to search a hypersphere of relatively large initial radius, we cannot test too large matrix size or too large initial radius in this experiment, because this experiment focuses on the performance evaluation of both adaptive sphere decoding and original sphere decoding. The original sphere decoding is not able to search a hypersphere of relatively large initial radius, and the complexity of searching a hypersphere increases as the size*

of matrix increases as well.

7.3 Simulation of communication channel

In this section, we present the simulation of a communication model, actually, the simulation of block-fading frequency-selective model as we discussed in Chapter 2. The communication system can be modeled as (2.1), to find the transmitted source signal s, it is equivalent to finding the minimizer to the integer least squares problem (3.1). If the communication system is modeled as (3.1), then the channel matrix $H \in \mathbb{R}^{(T+l-1)\times T}$ in this communication system has the following form:

$$H = \begin{bmatrix} h_1 & & & \\ h_2 & h_1 & & \\ \vdots & \ddots & \ddots & \\ h_l & \ddots & \ddots & h_1 \\ & h_l & \ddots & h_2 \\ & & \ddots & \vdots \\ & & & h_l \end{bmatrix}$$

where $h_i, i = 1, \cdots, l$ are the parameters of the channel and l is the order of the channel. Obviously, the channel matrix has a Toeplitz structure. In Experiment 2, the channel matrix H is normalized according to the power constraint of the channel, and it is randomly generated for this experiment, strictly speaking, its entries h_i are randomly generated and uniformly distributed over the interval $[-1, 1]$.

In this frequency-selective channel model, we assume that the entries of the

source signal vector s are chosen from points of an L-PAM constellation (2.3), where L is usually taken as a power of 2. Thus, in Experiment 2, the entries of the source signal vector are randomly chosen from the set $\{\pm 1, \pm 3, \pm 5, \pm 7, \cdots\}$, the size of this set depends on L.

In this communication model, the noise vector v is additive white Gaussian $\mathcal{N}(0, \sigma^2)$ with mean zero and variance σ^2, and the noise vector is scaled based on the given signal-to-noise ratio, namely, the noise variance σ^2 is obtained from the definition of SNR (2.4) for our communication model.

Therefore, the actual received vector y given to Experiment 2 is computed by $y = Hs + v$.

7.4 Statistical radius versus deterministic radius

In this section, we use the simulated communication system as given inputs, and investigate various behaviors in computing the initial radius of sphere decoding under certain conditions. We compare the size of radius computed by the statistical method and the deterministic method with the same set of input data under a certain SNR, and the number of iterations of sphere decoding with these radii. We see the size of radius computed by the statistical method and the deterministic method under various SNRs. We also see the size of radius computed by the statistical method under various probabilities, and the failure rate of sphere decoding with these radii.

Experiment 2:

In this experiment, L in L-PAM constellation is set to 4 ($L = 4$), so the entries of the source signal vector are chosen from the set $\{\pm 1, \pm 3\}$. We use the communication system discussed in previous section for channel matrix, source signal vector and white noise vector, and for each randomly generated channel matrix H, 200 random source signal vectors s and 200 random white noise vectors v are generated, then 200 received signal vectors y are computed by $y = Hs + v$. We take the average of total radius, total number of iterations and total failure rate of this 200 input data sets as each measurement. The order of channel is set to 3 or 5 ($l = 3$ or $l = 5$), and the length of source signal is set to 4 or 8 ($T = 4$ or $T = 8$). The sphere decoding algorithm we use in this experiment is the original sphere decoding algorithm (Algorithm 2).

First, we compare the size of statistical radius and deterministic radius under $SNR = 20dB$ and probability $p = 0.90$, and the number of iterations of sphere decoding with these radii as initial radius. The location of iteration counter in sphere decoding is exactly the same as in Experiment 1. Table 7.2 is the result of this comparison, it shows that the deterministic radius is smaller than the half of statistical radius for the same input data set, and the number of iterations of sphere decoding is significantly reduced by the deterministic radius as well.

Second, we compare the radii computed by the statistical method from various SNRs and under the starting probability $p = 0.90$. If the statistical method cannot find a radius with this starting probability, then we increment the probability p by 0.001 until a statistical radius is found. The result of this comparison is presented in Table 7.3. It shows that the radius computed by the statistical method is dependent on the SNR of source signal, as the SNR increases, the

Table 7.2: Statistical radius versus deterministic radius, and their number of iterations of sphere decoding, $SNR = 20dB$, $L = 4$ and starting probability $p = 0.90$

Size of signal T, Order of channel l	Statistical		Deterministic	
	Radius (r)	Num of iter (#)	Radius (r)	Num of iter (#)
$T = 4, l = 3$	1.95276	56	0.782344	9
$T = 4, l = 5$	2.1181	43	0.873108	7
$T = 8, l = 3$	3.35385	17830	1.46811	229
$T = 8, l = 5$	3.584	10865	1.68363	177

Table 7.3: Radius statistically computed from various $SNRs(dB)$, $L = 4$ and starting probability $p = 0.90$

Size of signal T, Order of channel l	SNR=10 (r)	SNR=15 (r)	SNR=20 (r)	SNR=25 (r)
$T = 4, l = 3$	2.76156	2.21867	1.76828	1.40668
$T = 4, l = 5$	3.08233	2.41042	1.79252	1.66449
$T = 8, l = 3$	4.71613	3.83557	2.93175	2.3602
$T = 8, l = 5$	4.76763	3.84142	2.96562	2.41821

statistical radius decreases.

Third, we compare the radii computed by the deterministic method from various SNRs. Since probability is irrelative to the deterministic radius, we will not take it as an input. The result of this comparison is presented in Table 7.4, it shows that the radius computed by the deterministic method is independent of the SNR of source signal. Whatever the SNR is large or small, the deterministic radius is almost the same.

Fourth, we compare the radii computed by the statistical method from various probabilities under $SNR = 20dB$, and the failure rates of sphere decoding with the radii computed by those probabilities. Table 7.5 is the result of this

Table 7.4: Radius deterministically computed from various $SNRs(dB)$, $L = 4$

Size of signal T, Order of channel l	SNR=10 (r)	SNR=15 (r)	SNR=20 (r)	SNR=25 (r)
$T = 4, l = 3$	0.697119	0.787917	0.70803	0.654931
$T = 4, l = 5$	1.0599	0.778816	0.773576	0.62176
$T = 8, l = 3$	1.52893	1.40133	1.40327	1.40129
$T = 8, l = 5$	1.53602	1.58759	1.6094	1.44537

Table 7.5: Radius statistically computed from various probabilities and failure rate of sphere decoding with the radius computed by that probability, $SNR = 20dB$ and $L = 4$

Size of signal T, Order of channel l	$p = 0.80$		$p = 0.85$		$p = 0.90$		$p = 0.95$	
	Radius (r)	Fail rate (%)	Radius (r)	Fail rate (%)	Radius (r)	Fail rate (%)	Radius (r)	Fail rate (%)
$T = 4, l = 3$	1.31651	7	1.39218	1.5	1.47905	1.5	1.62916	0.5
$T = 4, l = 5$	1.33581	8.5	1.41738	2	1.52552	1.5	1.70255	0.5
$T = 8, l = 3$	2.37797	1	2.4846	0	2.64743	0	2.86259	0
$T = 8, l = 5$	2.48788	1	2.5591	1	2.63494	0.5	2.89788	0

comparison. It shows that as probability increases, the radius computed by the statistical method increases as well, and the failure rate of sphere decoding decreases. Although increasing probability, therefore increasing radius is able to improve the success of sphere decoding, note, the efficiency and performance of sphere decoding decreases as the initial radius increases, so there is always a tradeoff between the efficiency of sphere decoding and the initial radius given to it. We must take this tradeoff into account when applying sphere decoding with the initial radius computed by the statistical method.

7.5 Deterministic radius versus revised deterministic radius

For communication applications, the matrix H and vector y could be simulated by the MATLAB build-in function "*rand*", since the entries of matrix H and vector y generated by "*rand*" are pseudo random values drawn from a standard uniform distribution on the open interval $(0, 1)$. The norm of this generated matrix H is relatively small, this corresponds the power constraint of the channel matrix. So the channel matrix and the received vector could be represented by this simulated H and y. And the size of H will be the transmitting antennae and receiving antennae in communication applications.

Experiment 3:

This experiment tests the failure rate of sphere decoding with the initial radius computed by the deterministic method (i.e., the initial radius is computed by Algorithm 5, the pure deterministic method), and a continuing experiment still targets the failure rate of sphere decoding, but compares the failure rates of the initial radii computed by the pure deterministic method and by the revised deterministic method (i.e., the initial radius is computed by Algorithm 6). The sphere decoding algorithm used in this experiment is the original sphere decoding algorithm (i.e., Algorithm 2). We test various sizes of matrix to evaluate their failure rates and the computed initial radii. In order to make the evaluation more convincible, we run the test for each size of matrix for a certain number of times, then take the average failure rate as the measurement. The number of

Table 7.6: Failure rate of sphere decoding with the initial radius computed by the deterministic method

Size of H	Average of computed radius (r)	Total number of runs (#)	Failure rate (%)
3×3	0.429184	1000	34
5×5	0.727958	500	17
7×7	1.0104	200	11
6×4	0.714878	500	37
9×6	1.07786	200	25
12×8	1.45425	100	16

runs depends on the size of matrix[2]. The result of this experiment is shown in Table 7.6.

From Table 7.6, we can see as the size of matrix increases, the hypersphere radius computed by the deterministic method increases as well, but the failure rate of sphere decoding decreases reversely. This implies the success or failure of sphere decoding is sensitive to the radius computed by the deterministic method.

Continuing on the last experiment, now the initial radius used for sphere decoding is computed not only from the pure deterministic method but also from the revised deterministic method, then we search the hypersphere again with the same sphere decoding algorithm (Algorithm 2). The result of this second experiment is shown in Table 7.7.

Table 7.7 compares the failure rates of sphere decoding. The "failure rate before" column is the sphere decoding with the initial radius computed by the pure deterministic method, and the "failure rate after" column is the sphere decoding with the initial radius computed by the revised deterministic method.

[2]Since sphere decoding is an NP-hard problem, consider the time issue, the number of runs decreases as the size of matrix increases.

Table 7.7: Failure rate of sphere decoding after using the initial radius computed by the revised deterministic method

Size of H	Average of computed radius (r)	Total num of runs (#)	Failure rate before (%)	Failure rate after (%)
3×3	0.428378	1000	36	0
5×5	0.735231	500	18	0
7×7	0.990062	200	9	0
6×4	0.711996	500	41	0
9×6	1.08746	200	28	0
12×8	1.46622	100	14	0

We can see after using the revised deterministic method to compute the initial radius, the failure rate of sphere decoding becomes zero. So by using the revised deterministic method to compute the initial radius, sphere decoding achieved a perfect success rate in our experiment, thus the success of sphere decoding is ensured.

7.6 Performance evaluation for revised deterministic radius

Since the initial radius computed by the revised deterministic method is increased a little bit, we may doubt that this increased initial radius can increases the performance of sphere decoding. We know that sphere decoding is an NP-hard problem, and its complexity increases as the initial radius increases. In this section, we evaluate the performance of sphere decoding with the initial radius computed by the revised deterministic method through the following experiment.

Experiment 4:

Table 7.8: Overall performance evaluation of sphere decoding with the radius computed by the non-revised deterministic method (Algorithm 5) and the revised deterministic method (Algorithm 6)

Size of H	Radius computed by Alg 5		Radius computed by Alg 6	
	Num of iter (#)	Running time (*sec*)	Num of iter (#)	Running time (*sec*)
3×3	213	0.00140401	213	0.00140401
5×5	20999	0.153661	20999	0.177685
7×7	156367	1.34348	156368	1.35705
6×4	58	0.000624004	58	0.000156001
9×6	3916	0.0313562	3916	0.0329162
12×8	23178	0.201865	23178	0.199525

In this experiment, we compare the performance of sphere decoding with the initial radius computed by both the pure deterministic method and the revised deterministic method from two measurements, the number of iterations and the running time. The iteration counter is exactly the same as in Experiment 1, that is, the iteration counter is simply placed right after each *while* statement. For the initial radii computed by both the pure deterministic method and the revised deterministic method, we run sphere decoding with these radii for a total number of 100 times, then take the average of total iterations of this 100 runs as a measurement. Similarly, we take the average of total running time of this 100 runs [3] as another measurement. The summary of result of this experiment is shown in Table 7.8.

From Table 7.8, we can see that the performances of sphere decoding with the initial radii computed by both the pure deterministic method (Algorithm 5) and the revised deterministic method (Algorithm 6) are almost the same, sphere

[3]Here, the running time is measured for sphere decoding only, not includes any setup time, QR decomposition time and the initial radius computing time etc.

decoding with the initial radius computed by the revised deterministic method does not introduce any significant overhead, and it achieves a perfect success rate which is already shown in previous experiment.

Chapter 8

Conclusion and Future Work

From our experiments, roughly speaking, the complexity of sphere decoding is exponential to either the initial radius of a hypersphere or the size of the lattice generating matrix.

8.1 Conclusion

This research work is motivated to address solving the integer least squares problem, which can be deployed in many applications such as communications. From the point of view of lattice, finding the solution to an integer least squares problem is equivalent to finding the closest lattice point to a given point. Sphere decoding is an effective technique in finding the closest lattice point, it is a maximum-likelihood detection algorithm with a relatively low expected complexity. This thesis investigates and discusses an improved sphere decoding technique, adaptive sphere decoding, which enhances the performance of the original sphere decoding. In addition, reducing the deterministic factor of sphere decod-

ing, the initial radius of a search hypersphere given to sphere decoding is also addressed. Furthermore, the floating-point errors in numerical processing are inevitable and may cause sphere decoding fail to find the closest lattice point, so an error analysis is performed for the computation of the initial radius of a search hypersphere. The major results and contributions of this thesis are presented below.

In Chapter 4, adaptive sphere decoding is shown and explained from both geometric interpretation and tree representation, it is an improvement to the original sphere decoding. Unlike the original sphere decoding, adaptive sphere decoding can reduce the radius of the search hypersphere to avoid going through all the lattice points inside the initial hypersphere, therefore it is able to reduce the search complexity. Generally speaking, the search complexity can be expected to be reduced in most cases. What needs to be pointed out is that the search complexity of adaptive sphere decoding may be the same as the search complexity of the original sphere decoding.

- In the extreme case, it is possible and unluckily that the next closer lattice point which adaptive sphere decoding found is the furthest one to the center inside this hypersphere. That is, adaptive sphere decoding still has to go through all the lattice points inside the initial hypersphere. Although it reduces the search radius whenever it finds a new closer lattice point, no other lattice point lies between this closer one and the surface of the search hypersphere, i.e., no any lattice points can be excluded from reaching.

- If the initial radius given to adaptive sphere decoding is just the distance between the closest lattice point and the center, then adaptive sphere de-

coding is not able to reduce the search radius and does not have a chance to reduce the search radius either. Because it has found the closest lattice point already in the initial hypersphere. So, the complexity of both sphere decoding algorithms is the same.

- If the initial radius given to adaptive sphere decoding is smaller than the distance between the closest lattice point and the center, then both sphere decoding algorithms are not able to find any lattice points inside the hypersphere. So, the complexity of both sphere decoding algorithms is the same as well.

The experiment shows that adaptive sphere decoding outperforms the original sphere decoding a lot.

In Chapter 5, different radius selection methods are discussed. The statistical method takes advantage of the characteristics of communication channel and utilizes the noise variance to compute the radius of a search hypersphere, therefore the search radius depends on the SNR of communication channel.

The deterministic method may target the communication channel as well. It is based on the Babai estimate in communications, but unlike the Babai estimate, which is taken as the solution to the integer least squares problem, the deterministic method goes further. It utilizes the Babai estimate to compute the radius of a search hypersphere, then try to find the closest lattice point inside this hypersphere. The search radius computed by the deterministic method can be small, therefore, the search complexity of sphere decoding is reduced.

Adaptively updating the search radius is discussed in this chapter, the basic idea is to reduce the search radius whenever adaptive sphere decoding finds a

closer lattice point inside a hypersphere. In this way, some lattice points inside the initial hypersphere may be excluded from reaching, therefore, the search complexity of sphere decoding is reduced.

In any numerical processing, computational errors are inevitable and need to be taken into account. The radius computed by the deterministic method can be small, but sometimes this radius can be too small and make sphere decoding failure. So, in Chapter 6, the rounding error of floating-point is considered in the computation of the initial radius. Based on the error analysis of matrix-vector multiplication and vector-vector addition, an error analysis of computing the initial radius is performed and a correctional formula is proposed for computing the initial radius by the deterministic method. The experiment shows that the revised deterministic method guarantees the success of sphere decoding, and at the same time it does not produce a significant overhead to the performance of sphere decoding.

8.2 Future work

8.2.1 Complexity of adaptive sphere decoding

We know that sphere decoding is an effective technique for obtaining maximum-likelihood detection performance in polynomial complexity for certain applications which can be modeled as the NP-hard integer least squares problem and under certain assumptions. Generally speaking, solving a general NP-hard integer least squares problem with sphere decoding is still an NP-hard problem, this is shown in our experiments. When the input data given to sphere decoding is

general, at the same time both the size of matrix and the initial radius given to sphere decoding is large, then the running time of sphere decoding is exponentially increased. Sometimes the running of sphere decoding cannot terminate in a reasonable time. Sphere decoding applying to a general integer least squares problem cannot be exploited beneficially.

In our experiment, we have shown that adaptive sphere decoding is a more effective searching method than the original sphere decoding when the integer least squares problem is general. The experiment shows that adaptive sphere decoding outperforms the original sphere decoding a lot, it reduces the running time significantly for the same input data. But we do not know what is the exact complexity of adaptive sphere decoding. In future work, we will derive the complexity of adaptive sphere decoding formally and are expecting that adaptive sphere decoding is able to achieve the polynomial complexity for the general integer least squares problem.

8.2.2 Applying LLL algorithm

As mentioned before, some of the preprocessing methods such as QR decomposition and lattice reduction may be applied before sphere decoding to transform the integer least squares problem into a simpler form. We have already discussed the QR decomposition for sphere decoding. Another approach is lattice reduction. In this approach, we attempt to find an invertible $m \times m$ matrix M, such that both M and M^{-1} are integer matrices (unimodular matrices), therefore the matrix HM preserves the lattice structure. Denote $s = Mt$ and $G = HM$, where M is aforementioned $m \times m$ invertible integer matrix (unimodular matrix), then

the integer least squares problem (3.1) becomes

$$\min_{t\in\mathbb{Z}^m} \|Gt - y\|_2^2 \tag{8.1}$$

Thus, sphere decoding may be applied to (8.1), then it is straightforward to solve s by $s = Mt$.

However, the lattice reduction approach is itself NP-hard, the famous LLL algorithm [25] is a strategic approach to lattice reduction. The LLL algorithm is originated from Lenstra, Lenstra and Lovász, it is widely used by researchers as a preprocessor to solve the integer least squares problem, it is often arguably referred to as an integer Gram-Schmidt procedure.

Suppose that QR decomposition is applied to the integer least squares problem (3.1), it is reduced to (3.2), then apply the LLL algorithm to the upper triangular matrix R to decompose R into

$$R = \hat{Q}\hat{R}M^{-1} \tag{8.2}$$

where $\hat{Q}$ is orthogonal, $\hat{R}$ is upper triangular and M is unimodular, so M^{-1} is an integer matrix. The LLL algorithm transforms a basis formed by the columns of R into a basis formed by the columns of $\hat{R}$, the lengths of the columns of $\hat{R}$ are shorter than those of R, so that the columns of $\hat{R}$ form a reduced basis for a lattice space, see [26], [27] and [21] for the details of LLL reduction.

Generally speaking, LLL algorithm is able to reduce the computational complexity of sphere decoding in two ways. First, it can reduce the initial radius of the hypersphere by reducing the norm of R. Second, since sphere decoding

is a depth-first search algorithm for the lattice points inside a hypersphere, the LLL algorithm can reduce the total number of search paths. Because in the tree representation of sphere decoding, LLL algorithm can shrink the integer interval at each level of the tree, therefore the number of nodes is reduced at each level of the tree. Consequently, the search paths as well as the complexity of sphere decoding is reduced.

8.2.3 Other methods improving sphere decoding

We know that integer least squares problem is NP-hard, although sphere decoding is an effective technique for obtaining maximum-likelihood detection performance in polynomial complexity for certain applications and under certain assumptions, generally speaking, it could still be shown to be exponential complexity for general cases. In this thesis, we do not address the other deterministic factor of integer least squares problem, the size of lattice space, which also causes sphere decoding to be exponential in general. How to reduce the complexity of sphere decoding for a general NP-hard integer least squares problem, we propose the following idea for the future work.

In the tree representation of sphere decoding, the size of lattice space is the depth of the tree. Since sphere decoding is depth-first search technique, it has to go through all the nodes of the tree (except the root). Sphere decoding has to find all the paths from the root node's children to the leaves of full tree depth and compares the Euclidean distances of the lattice points corresponding to those paths. If the size of lattice space is large, then the depth of the tree is large as well, and the computation of those Euclidean distances is expensive;

there are also tremendous paths and the comparison of those Euclidean distances is expensive as well.

Amongst these tremendous many paths of full tree depth, only one path[1] is the one corresponding to the closed lattice point, but sphere decoding must compare all the paths of full tree depth to determine the closest lattice point. This means, sphere decoding has to reach a leaf of full tree depth every time in order to compute the Euclidean distance. So we propose an improvement for sphere decoding, no need to reach the leaf of full tree depth, just compute and compare the distance. In the algorithm of sphere decoding, we always keep the currently shortest distance. In the proposed improvement, we compute the distance for each sub-dimensional lattice space whenever we compute a new entry for the vector s, namely, we compute the distance at each level of the tree, and compare this distance with the currently shortest one[2] to determine whether going further from this node or not. If the distance in sub-dimensional lattice space is already greater than the currently shortest distance, we ignore searching this node and thereafter its descendants, and go to the next node (its sibling) or go back to upper level (go up to search its parent's sibling or go up further). If the distance in sub-dimensional lattice space is still smaller than the currently shortest distance, we keep going further. We know this strategy is correct because of the characteristics of Euclidean distance. In general cases, the integer interval is large at each level of the tree, we can expect that this improvement method is feasible.

[1]Possibly many, but they all have the same Euclidean distance.

[2]This currently shortest Euclidean distance is in the full-dimensional lattice space, namely, m-dimensional lattice space.

Note, the drawback of this improvement is obvious too, i.e., the cost of computing distance at each level of the tree. So we expect that this improved sphere decoding may be suitable for a problem with a large lattice space and we have to take the cost of computing distance at each level into account. If the lattice space is large, the performance we obtained overcomes the cost of computing distance at each level. But if the lattice space is relatively small, then cost of computing distance at each level is overwhelming.

Author's Publications

Published papers

1. Fei Zhao and Sanzheng Qiao, "Radius Selection Algorithms for Sphere Decoding". In: *Proceedings of C3S2E-09, ACM International Conference Proceedings Series*, edited by Bipin C. Desai. Concordia University, Montreal, QC, CANADA, 19-21 May, 2009.

Bibliography

[1] B. Hassibi and H. Vikalo, "On the Sphere Decoding Algorithm I. Expected Complexity", *IEEE Transactions on Signal Processing*, vol. 53, no. 8, pp. 2806-2818, Aug. 2005

[2] O. Goldreich, S. Goldwasser and S. Halevi, "Public-Key Cryptosystems from Lattice Reduction Problems", *Proceedings of the 17th Annual International Cryptology Conference on Advances in Cryptology*, pp. 112-131, Aug. 1997

[3] A. Hassibi and S. Boyd, "Integer parameter estimation in linear models with applications to GPS", *IEEE Transactions on Signal Processing*, vol. 46, no. 11, pp. 2938-2952, Nov. 1998

[4] P. van Emde Boas, "Another NP-complete partition problem and the complexity of computing short vectors in lattices", *Tech. Rept. 81-04, Department of Mathematics, University of Amsterdam*, 1981.

[5] M. Grotschel, L. Lovasz and A. Schriver, "Geometric Algorithms and Combinatorial Optimization", *2nd Edition, Springer-Verlag*, New York, 1993.

[6] O. Damen, A. Chkeif and J.-C. Belfiore, "Lattice code decoder for space-time codes", *IEEE Communications Letters*, vol. 4, no. 5, pp. 161-163, May 2000

[7] Gerard J. Foschini, "Layered space-time architecture for wireless communicatio in a fading environment when using multi-element antennas", *Bell Labs Technical Journal*, vol. 1, no. 2, pp. 41-59, Oct. 1996

[8] Ravi Kannan "Improved algorithms on integer programming and relate lattice problems", *Proceedings of the fifteenth annual ACM symposium on Theory of computing*, pp. 193-206, 1983

[9] J. Lagarias, H. Lenstra, and C. Schnorr, "Korkin-Zolotarev bases and successive minima of a lattice and its reciprocal", *Combinatorica*, vol. 10, pp. 333-348, 1990

[10] A. Korkin and G. Zolotarev, "Sur les formes quadratiques", *Mathematische Annalen*, vol. 6, pp. 366-389, 1873

[11] U. Fincke and M. Pohst, "Improved methods for calculating vectors of short length in a lattice, including a complexity analysis", *Mathematics of Computation*, vol. 44, no. 170, pp. 463-471, Apr. 1985

[12] I. E. Telatar, "Capacity of multi-antenna Gaussian channels", *European Transactions on Telecommunications*, vol. 6, no. 10, pp. 585-595, 1999

[13] W. H. Mow, "Maximum likelihood sequence estimation from the lattice-viewpoint", *IEEE Transactions on Information Theory*, vol. 40, no. 9, pp. 1591-1600, Sep. 1994

[14] H. Vikalo and B. Hassibi, "On the Sphere-Decoding Algorithm II. Generalizations, Second-Order Statistics, and Applications to Communications", *IEEE Transactions on Signal Processing*, vol. 53, no. 8, pp. 2819-2834, Aug. 2005

[15] H. Vikalo and B. Hassibi, "Maximum-Likelihood Sequence Detection of Multiple Antenna Systems over Dispersive Channels via SphereDecoding", *EURASIP Journal on Applied Signal Processing*, vol. 2002, no. 5, pp. 525-531, 2005

[16] G. D. Forney, Jr. "Maximum-Likelihood Sequence Estimation of Digital Sequences in the Presence of Intersymbol Interference", *IEEE Transactions on Information Theory*, vol. 18, no. 318, pp. 363-378, May 1972

[17] D. Micciancio, "The hardness of the closest vector problem with preprocessing", *IEEE Transactions on Information Theory*, vol. 47, no. 3, pp. 1212-1215, Mar. 2001

[18] E. Agrell, T. Eriksson, A. Vardy and K. Zeger, "Closest point search in lattices", *IEEE Transactions on Information Theory*, vol. 48, no. 8, pp. 2201-2214, Aug. 2002

[19] C. P. Schnorr and M. Euchner, "Lattice basis reduction: Improved practical algorithms and solving subset sum problems", *Mathematical Programming*, vol. 66, no. 2, pp. 181-191, Sep. 1994

[20] G. H. Golub and C. F. Van Loan, *Matrix Computations, 3rd Edition*, Johns Hopkins University Press, Baltimore, Maryland, 1996

[21] Sanzheng Qiao, "Integer least squares: Sphere decoding and the LLL algorithm", *Proceedings of C3S2E-08, ACM International Conference Proceedings Series*, pp. 23-28, May 2008

[22] E. Viterbo and J. Boutros, "A universal lattice code decoder for fading channels", *IEEE Transactions on Information Theory*, vol. 45, no. 5, pp. 1639-1642, Jul. 1999

[23] E. Viterbo and E. Biglieri, "A universal decoding algorithm for lattice codes", *Quatorzieme colloque GRETSI*, pp. 611-614, Sep. 1993

[24] Nicholas J. Higham, "Accuracy and Stability of Numerical Algorithms, Second Edition", *Society for Industrial and Applied Mathematics*, 2002

[25] A. K. Lenstra, H. W. Lenstra and L. Lovász, "Factoring polynomials with rational coefficients", *Mathematische Annalen*, vol. 261, no. 4, pp. 515-534, Dec. 1982

[26] F. Luk and D. Tracy, "An improved LLL algorithm", *Linear Algebra and Its Applications*, vol. 428, no. 2-3, pp. 441-452, 2008.

[27] F. Luk and S. Qiao, "Numerical properties of the LLL algorithm", *Advance Signal Processing Algorithms, Architectures, and Implementations XVII*, volume 6697 of *Proceedings of SPIE, the International Society for Optical Engineering*, pp 6697-3, 2007.