# Integer Least Squares: Sphere Decoding and the LLL Algorithm [*]

Sanzheng Qiao[†]
Department of Computing and Software, McMaster University,
1280 Main St. West
Hamilton, Ontario, L8S 4L7, Canada.

## ABSTRACT

This paper considers the problem of integer least squares, where the least squares solution is an integer vector, whereas the coefficient matrix is real. In particular, we discuss the sphere decoding method in communications. One of the key issues in sphere decoding is the determination of the radius of search sphere. We propose a deterministic method for finding a radius of search sphere. Also, we investigate the impact of the LLL algorithm on the computational complexity of the sphere decoding method.

## Categories and Subject Descriptors

G.1.6 [**NUMERICAL ANALYSIS**]: Optimization—*Least squares methods*

## General Terms

Performance

## Keywords

integer least squares, sphere decoding, LLL algorithm

## 1. INTRODUCTION

Given a real $n$-by-$m$, $n \geq m$, matrix $A$ and a real $n$-vector $\mathbf{y}$, the linear least squares problem is to find a minimizer, a real $m$-vector in general, of the problem:

$$\min_{\mathbf{x}} \|A\mathbf{x} - \mathbf{y}\|_2^2. \tag{1}$$

One of standard methods for solving the problem when $A$ is of full column rank is as follows [2]. The matrix $A$ is first reduced to upper triangular by orthogonal transformations using the QR decomposition

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

where $Q$ is orthogonal of order $n$ and $R$ is upper triangular of order $m$. Partitioning $Q = [Q_1\ Q_2]$, where $Q_1$ is $n$-by-$m$ and $Q_2$ is $n$-by-$(n-m)$, we have

$$\begin{aligned} & \|A\mathbf{x} - \mathbf{y}\|_2^2 \\ = & \left\| \begin{bmatrix} R \\ 0 \end{bmatrix} \mathbf{x} - Q^{\mathrm{T}}\mathbf{y} \right\|_2^2 \\ = & \|R\mathbf{x} - Q_1^{\mathrm{T}}\mathbf{y}\|_2^2 + \|Q_2^{\mathrm{T}}\mathbf{y}\|_2^2. \end{aligned}$$

Then the least squares problem (1) is reduced to the triangular least squares problem:

$$\min_{\mathbf{x}} \|R\mathbf{x} - \hat{\mathbf{y}}\|_2^2,$$

where $\hat{\mathbf{y}}$ denotes $Q_1^{\mathrm{T}}\mathbf{y}$. Under the assumption that $A$ has full column rank, $R$ is nonsingular and the solution, a real vector in general, of the triangular system

$$R\mathbf{x} = \hat{\mathbf{y}}$$

is the least squares solution for (1). In this paper, we consider the problem (1) with the constraint that the solution is an integer $m$-vector, that is, its entries are integers, thus called integer least squares problem:

$$\min_{\mathbf{x} \in Z^m} \|A\mathbf{x} - \mathbf{y}\|_2^2. \tag{2}$$

Following the above discussion, this problem can be reduced to the triangular integer least squares problem:

$$\min_{\mathbf{x} \in Z^m} \|R\mathbf{x} - \hat{\mathbf{y}}\|_2^2 \tag{3}$$

by applying the QR decomposition. Equivalently, it is to find a lattice point closest to $\hat{\mathbf{y}}$, in 2-norm sense, where the lattice points are under the basis formed by the columns of $R$.

The integer least squares problem arises from many applications: communications [4], cryptography [1], GPS [3], to name a few, where the solution $\mathbf{x}$ to be found is a code vector consisting of integer entries.

A seemingly obvious way of solving the triangular integer least squares problem (3) would first solve the triangular system $R\mathbf{x} = \hat{\mathbf{y}}$ for a real least squares solution $\mathbf{x}$, then round the entries of the solution vector to their nearest integers, that is, the integer vector closest to the real least squares solution. The following example shows that this simple approach fails to produce the integer least squares solution.

EXAMPLE 1. *Let*

$$A = \begin{bmatrix} 1 & 4 \\ 2 & 5 \\ 3 & 6 \end{bmatrix} \quad \textit{and} \quad \mathbf{y} = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix},$$

[*] The author is partially supported by NSERC of Canada.
[†] E-mail: qiao@mcmaster.ca

*C3S2E-08 2008* May 12-13, Montreal [QC, CANADA]
Copyright ©2008 ACM 978-1-60558-101-9/08/05 ...$5.00.

*then after the QR decomposition, we get*

$$R = \begin{bmatrix} 3.7417 & 8.5524 \\ 0 & 1.9640 \end{bmatrix},$$

*and the real least squares solution*

$$\mathbf{x} = \begin{bmatrix} -0.3333 \\ 0.3333 \end{bmatrix}.$$

*Rounding its entries to their nearest integers, we get*

$$\lceil \mathbf{x} \rfloor = \begin{bmatrix} 0 \\ 0 \end{bmatrix},$$

*which gives the residual norm* $\|A \lceil \mathbf{x} \rfloor - \mathbf{y}\|_2 = 1.7321$. *However, the integer least squares solution is*

$$\mathbf{z} = \begin{bmatrix} -2 \\ 1 \end{bmatrix},$$

*which produces a smaller residual norm* $\|A\mathbf{z} - \mathbf{y}\|_2 = 1.4142$.

As pointed out, the problem of integer least squares is equivalent to that of finding the closest, in the 2-norm sense, lattice point to a given point, which is known to be NP-hard. Hassibi and Vikalo [4], considering the application of sphere decoding in communications, gave a statistical method and showed that its expected complexity is polynomial, often roughly cubic in practice. In this paper, we also consider the application of sphere decoding in communications and propose a practical deterministic method. This is presented in Section 2. Then, in Section 3, we show how the LLL algorithm can be used to reduce the computational complexity. Finally, we demonstrate our experiments in Section 4.

## 2. SPHERE DECODING

The complexity of searching for the integer least squares solution in the entire lattice space is nonpolynomial. So, to reduce the complexity, we limit the search region to a sphere containing the integer least squares solution.

Consider the the triangular integer least squares problem (3) constrained in a sphere $\mathcal{S}$ centered at $\hat{\mathbf{y}}$:

$$\min_{\mathbf{x} \in \mathcal{S}} \|R\mathbf{x} - \hat{\mathbf{y}}\|_2^2, \quad \mathcal{S} = \{\mathbf{x}, \;\; \mathbf{x} \in Z^m \text{ and } \|R\mathbf{x} - \hat{\mathbf{y}}\|_2 \le \rho\}, \tag{4}$$

given a radius of a sphere $\mathcal{S}$. Partition the upper triangular

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

where $R_1$ is the order $m-1$ leading principal submatrix of $R$, $\mathbf{r}_{1:m-1,m}$ is the subvector of the last column of $R$ consisting of its first $m-1$ entries, and $r_{m,m}$ the last entry of the last column of $R$. Accordingly, partition

$$\mathbf{x} = \begin{bmatrix} \mathbf{x}_1 \\ x_m \end{bmatrix} \quad \text{and} \quad \hat{\mathbf{y}} = \begin{bmatrix} \mathbf{y}_1 \\ y_m \end{bmatrix},$$

where $\mathbf{x}_1$ and $\mathbf{y}_1$ are the $(m-1)$-subvectors of $\mathbf{x}$ and $\hat{\mathbf{y}}$ respectively and $x_m$ and $y_m$ the last entries of $\mathbf{x}$ and $\hat{\mathbf{y}}$ respectively. Then we get

$$\begin{aligned} & \|R\mathbf{x} - \hat{\mathbf{y}}\|_2^2 \\ = \;\; & \|R_1\mathbf{x}_1 - (\mathbf{y}_1 - x_m\mathbf{r}_{1:m-1,m})\|_2^2 + (r_{m,m}x_m - y_m)^2. \end{aligned}$$

Denote $\hat{\mathbf{y}}_1 = \mathbf{y}_1 - x_m\mathbf{r}_{1:m-1,m}$ and $\hat{\rho}^2 = \rho^2 - (r_{m,m}x_m - y_m)^2$. In order for $\|R\mathbf{x} - \hat{\mathbf{y}}\|_2^2 \le \rho^2$, it is necessary that

$$|r_{m,m}x_m - y_m| \le \rho \tag{5}$$

and

$$\|R_1\mathbf{x}_1 - \hat{\mathbf{y}}_1\|_2^2 \le \hat{\rho}^2. \tag{6}$$

The inequality (6) is the problem of finding the lattice points inside the sphere of radius $\hat{\rho}$ in dimension $m-1$. The solutions for (5) are the integers between $(-\rho + y_m)/r_{m,m}$ and $(\rho + y_m)/r_{m,m}$, that is, assuming $r_{m,m} > 0$, the integers

$$\lceil(-\rho + y_m)/r_{m,m}\rceil, \ldots, \lfloor(\rho + y_m)/r_{m,m}\rfloor, \tag{7}$$

where $\lceil x \rceil$ denotes the smallest integer that is greater than or equal to $x$ and $\lfloor x \rfloor$ the largest integer that is less than or equal to $x$. Thus, for each integer $x_m$ in (7), we compute a new search radius $\hat{\rho}$: $\hat{\rho}^2 = \rho^2 - (r_{m,m}x_m - y_m)^2$, and $\hat{\mathbf{y}}_1 = \mathbf{y}_1 - x_m\mathbf{r}_{1:m-1,m}$, then solve the $(m-1)$-dimensional problem (6).

We summarize the above procedure of solving the triangular integer least squares problem in the following recursive algorithm.

ALGORITHM 1. *Using the notations above, given an m-by-m upper triangular R with positive diagonal, an m-vector* $\hat{\mathbf{y}}$*, and the radius* $\rho$ *of a search sphere, this algorithm finds the lattice points inside the sphere* $\mathcal{S} = \{\mathbf{x}, \;\; \|R\mathbf{x} - \hat{\mathbf{y}}\|_2 \le \rho\}$.

```
$\mu = \lfloor(\rho + \hat{y}_m)/r_{m,m}\rfloor$;
$\lambda = \lceil(-\rho + \hat{y}_m)/r_{m,m}\rceil$;
if $m = 1$
    return the integers in $[\lambda, \mu]$;
else
    for each integer $x_m \in [\lambda, \mu]$
        $\hat{\mathbf{y}}_1 = \hat{\mathbf{y}}_{1:m-1} - x_m\mathbf{r}_{1:m-1,m}$;
        $\rho_1^2 = \rho^2 - (\hat{y}_m - x_m r_{m,m})^2$;
        apply this algorithm to $R_{1:m-1,1:m-1}$, $\hat{\mathbf{y}}_1$,
        and $\rho_1$;
        append $x_m$ to each of the solutions of
        dimension $m-1$;
    end
end
```

As shown above, this algorithm is a depth-first search method. The solution for (4), if it exists, can then be found by searching all the lattice points inside the sphere for the one that minimizes $\|R\mathbf{x} - \hat{\mathbf{y}}\|_2^2$. Apparently, the above algorithm shows that the number of lattice points inside the sphere can grow rapidly as the search radius increases. Thus the key issue is the choice of the radius. Obviously, as $\rho \to \infty$ in the extreme case, we search the entire lattice space, which is computationally prohibitive. On the other hand, if $\rho$ is too small, the sphere may contain no lattice points. How to determine an appropriate radius? Instead of the general case and worst case complexity, which lead to an NP-hard problem, we consider an important application: Sphere decoding in communications. In this application, $A$ in (2) is a matrix representing a communication channel and $n = m + k - 1$, where $k$ is the order of the channel, $\mathbf{x}$ the original code vector, and $\mathbf{y}$ the received signal vector. Hassibi and Vikalo [4] consider the statistical characteristics of the noise included in $\mathbf{y}$, propose a statistical method for choosing the radius $\rho$, and show that the expected complexity is polynomial, often roughly cubic. Specifically, in this application, $\mathbf{y} = A\mathbf{x} + \mathbf{v}$, where $\mathbf{v}$ is white noise, whose entries are random variables normally distributed with zero mean and variance $\sigma^2$. The following algorithm describes their method for choosing $\rho$.

Algorithm 2. *Given a variance $\sigma^2$ of the random noise and a probability $p$, this algorithm computes a radius $\rho$ of the search sphere $\mathcal{S}$ in dimension $n$ such that the solution for (2) can be found in the sphere with probability $p$.*

1. Find $\alpha$ satisfying
$$p = \int_0^{\alpha n/2} \frac{\lambda^{n/2-1}}{\Gamma(n/2)} e^{-\lambda} \mathrm{d}\lambda;$$
2. $\rho^2 = \alpha n \sigma^2$.

As pointed out in [4], the above method only considers the statistical characteristics of the noise, not the channel matrix. It solely depends on the probability $p$, the noise variance $\sigma^2$, and the dimension $n$.

In this paper, we take the properties of the channel matrix into account as well and propose a deterministic method for finding the radius. In this application, because of the power constraint of the channel, the norm of $A\mathbf{x}$ is capped. In other words, the norm of $A$ is not too large, which means that when $A$ is applied to a vector, it does not magnify the vector length very much. Based on this property, we propose the following procedure of determining the radius of the search sphere.

Algorithm 3. *Given $R$ and $\hat{\mathbf{y}}$ in (4), this algorithm finds a radius $\rho$ so the the search sphere in (4) contains at least one lattice point.*

1. Solve $R\mathbf{x} = \hat{\mathbf{y}}$ for $\mathbf{x}$, the least squares solution, a real vector;
2. Round the entries of $\mathbf{x}$ to their nearest integers, $\hat{\mathbf{x}} = \lceil \mathbf{x} \rfloor$;
3. Set $\rho = \|R\hat{\mathbf{x}} - \hat{\mathbf{y}}\|_2$.

Obviously, the sphere $\|R\mathbf{x} - \hat{\mathbf{y}}\|_2 \le \rho$ contains at least one lattice point, namely $\hat{\mathbf{x}}$, the integer vector closest to the real least squares solution $\mathbf{x}$. Now, we examine the size of $\rho$. Let $\mathbf{d} = \hat{\mathbf{x}} - \mathbf{x}$, then
$$R\hat{\mathbf{x}} - \hat{\mathbf{y}} = R(\mathbf{x} + \mathbf{d}) - \hat{\mathbf{y}} = R\mathbf{d}.$$
Since $\mathbf{d} = \lceil \mathbf{x} \rfloor - \mathbf{x}$, $\|\mathbf{d}\|_2 \le \sqrt{m}/2$. Thus
$$\rho = \|R\mathbf{d}\|_2 \le \frac{\sqrt{m}}{2}\|R\|_2 = \frac{\sqrt{m}}{2}\|A\|_2.$$
As pointed out previously, in this application, due to the channel power constraint, $\|A\|_2$ is not large. Consequently, the radius of the search sphere is not large. Moreover, since $\hat{\mathbf{x}}$, the integer vector closest to the real least squares solution, is on the surface of the sphere, we expect that the radius given by the above method is tight when the variance $\sigma^2$ of the noise is moderate. 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.

## 3. THE LLL ALGORITHM

In this section, we will show how the LLL algorithm [5] can be used to reduce the computational complexity in two ways. First, it can be used to reduce the radius of the search sphere by reducing the norm of $R$. Second, as shown in Algorithm 1, the sphere decoding is a depth-first search for the lattice points inside a sphere, the LLL algorithm can be used to reduce the total number of search paths.

The famous LLL algorithm is originated from Lenstra, Lenstra, and Lovász [5]. In applications like cryptography, where the data matrix $A$ is an integer matrix, the LLL algorithm computes the decomposition
$$A = BM,$$
where $B$ is also an integer matrix and $M$ is unimodular, a nonsingular integer matrix whose determinant equals ±1. Thus the LLL algorithm transforms a basis formed by the columns of $A$ into a basis formed by the columns of $B$. The lengths of the columns of $B$ are shorter than those of $A$. In this case, the LLL algorithm can be performed in integer arithmetic. In our case, however, $A$ in (2) is a real matrix. Thus the LLL algorithm has to be performed in floating-point arithmetic. Since the integer least squares problem (2) is reduced to the triangular integer least squares problem (3), the LLL algorithm is applied to the upper triangular $R$. It decomposes $R$ into
$$R = \widehat{Q}\widehat{R}M^{-1}, \tag{8}$$
where $\widehat{Q}$ is orthogonal, $\widehat{R}$ upper triangular, and $M$ unimodular, so that the columns of $\widehat{R}$ form a reduced basis for the lattice space defined by

Definition 1. *The columns of an upper triangular $R = [r_{i,j}]$ form a reduced basis if*

1. $|r_{i,j}| \le |r_{i,i}|/2, \quad j > i$;
2. $r_{i,i}^2 \ge |\omega r_{i-1,i-1}^2 - r_{i-1,i}^2|$, where $0.25 < \omega < 1$.

We call the decomposition (8) the QRM decomposition. Since $M$ is unimodular, $M^{-1}$ is an integer matrix.

For the LLL algorithm in floating-point, Luk and Tracy [6] present a Givens reflection based LLL algorithm. In [7], Luk and Qiao show that the Givens reflection based LLL algorithm is numerically more robust than the original LLL algorithm. However, the Givens reflection based LLL algorithm must be performed in floating-point arithmetic, whereas the original LLL algorithm can be performed in integer arithmetic when the original $A$ is an integer matrix. Here is an outline of the Givens reflection based LLL algorithm, see [6, 7] for details.

Algorithm 4. *Given an upper triangular $R = [r_{i,j}]$ of order $n$ and a parameter $\omega$, $0.25 \le \omega \le 1.0$, this algorithm computes an orthogonal $Q$ and an integer unimodular $M$ and overwrites $R$, so that the new upper triangular $R$ equals $Q^{\mathrm{T}}RM$ and its columns form a reduced basis.*

```
$k = 2$;
while $k \le n$
    if $|r_{k-1,k-1}| < 2|r_{k-1,k}|$
        postmultiply $R$ with an integer elementary
        unimodular $M_{k-1,k}$ to reduce $|r_{k-1,k}|$;
    end
    if $r_{k,k}^2 < |\omega r_{k-1,k-1}^2 - r_{k-1,k}^2|$
        swap columns $k-1$ and $k$ of $R$;
        premultiply $R$ with a Givens reflection $J_k$
        to restore the upper triangular structure;
        $k = \max(k-1, 2)$;
        else
            for $i = k-2$ down to 1
                if $|r_{i,i}| < 2|r_{i,k}|$
                    postmultiply $R$ with an integer
                    elementary unimodular $M_{i,k}$
                    to reduce $|r_{i,k}|$;
                end
            end
            $k = k+1$;
        end
    end
```

In the above algorithm, an order $n$ elementary unimodular matrix has the form:

$$M_{i,j} = I_n - \gamma \mathbf{e}_i \mathbf{e}_j^{\mathrm{T}}, \quad j > i,$$

where $I_n$ is the identity matrix of order $n$, $\gamma$ an integer, and $\mathbf{e}_i$ the $i$th unit vector, that is, the $i$th column of $I_n$. Similar to Gaussian elimination, by setting

$$\gamma = \lceil r_{i,j}/r_{i,i} \rfloor$$

and postmultiplying $R$ in (3) with $M_{i,j}$, we can reduce $|r_{i,j}|$ to satisfy the first condition in Definition 1.

An order $n$ Givens reflection has the form:

$$J_k = \begin{bmatrix} I_{k-2} & & & \\ & c & s & \\ & s & -c & \\ & & & I_{n-k} \end{bmatrix}, \quad c^2 + s^2 = 1.$$

Let $\Pi_k$ be the permutation matrix:

$$\Pi_k = \begin{bmatrix} I_{k-2} & & & \\ & 0 & 1 & \\ & 1 & 0 & \\ & & & I_{n-k} \end{bmatrix},$$

then $R\Pi_k$ is the matrix obtained by swapping the columns $k-1$ and $k$ of $R$. By setting

$$c = \frac{r_{k-1,k}}{\sqrt{r_{k-1,k}^2 + r_{k,k}^2}} \quad \text{and} \quad s = \frac{r_{k,k}}{\sqrt{r_{k-1,k}^2 + r_{k,k}^2}},$$

$J_k R \Pi_k$ is upper triangular and the new $r_{k-1,k-1}$, $r_{k-1,k}$, and $r_{k,k}$ satisfy the second condition in Definition 1.

The first condition in Definition 1 implies that the off diagonal entries of the reduced $\widehat{R}$ are small relative to the ones on the diagonal. Also, from the algorithm, the diagonal of the reduced $\widehat{R}$ is not much larger than that of $R$. Thus, we expect that $\|\widehat{R}\|_1$ (or $\|\widehat{R}\|_\infty$) is smaller than $\|R\|_1$ (or $\|R\|_\infty$). It is then likely that $\|\widehat{R}\|_2$ is smaller than $\|R\|_2$. Thus the radius is reduced. The following example illustrates this effect of the first condition.

EXAMPLE 2. *Let $A$ and $\mathbf{y}$ be the same as in Example 1. After the QR decomposition of $A$, we have*

$$R = \begin{bmatrix} 3.7417 & 8.5524 \\ 0 & 1.9640 \end{bmatrix} \quad \text{and} \quad \hat{\mathbf{y}} = \begin{bmatrix} 1.6036 \\ 0.6547 \end{bmatrix}.$$

*The solution of $R\mathbf{x} = \hat{\mathbf{y}}$ is*

$$\mathbf{x} = \begin{bmatrix} -0.3333 \\ 0.3333 \end{bmatrix} \quad \text{and} \quad \lceil \mathbf{x} \rfloor = \begin{bmatrix} 0 \\ 0 \end{bmatrix},$$

*which, using Algorithm 3, gives the radius $\rho = \|\hat{\mathbf{y}}\|_2 = 1.7321$.*

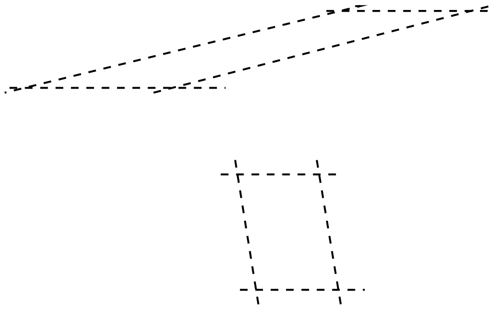

**Figure 1: The bottom lattice grid is obtained by applying the LLL algorithm to the top lattice grid in Example 2.**

*Applying the LLL algorithm to $R$ with $\omega = 0.75$, we get*

$$\widehat{R} = \begin{bmatrix} 2.2361 & -0.4472 \\ 0 & 3.2864 \end{bmatrix},$$

$$\widehat{Q} = \begin{bmatrix} 0.4781 & 0.8783 \\ 0.8783 & -0.4781 \end{bmatrix},$$

$$M = \begin{bmatrix} -2 & 3 \\ 1 & -1 \end{bmatrix}$$

*and the updated*

$$\widehat{Q}^{\mathrm{T}} \hat{\mathbf{y}} = \begin{bmatrix} 1.3416 \\ 1.0955 \end{bmatrix}.$$

*Apparently, both $\|\widehat{R}\|_1 < \|R\|_1$ and $\|\widehat{R}\|_\infty < \|R\|_\infty$ . The solution of $\widehat{R}\mathbf{x} = \widehat{Q}^{\mathrm{T}}\hat{\mathbf{y}}$ is*

$$\hat{\mathbf{x}} = \begin{bmatrix} 0.6667 \\ 0.3333 \end{bmatrix} \quad \text{and} \quad \lceil \hat{\mathbf{x}} \rfloor = \begin{bmatrix} 1 \\ 0 \end{bmatrix},$$

*which, from Algorithm 3, leads to a smaller radius*

$$\begin{aligned} \hat{\rho} &= \|\widehat{R}\,\mathrm{round}(\hat{\mathbf{x}}) - \widehat{Q}^{\mathrm{T}}\hat{\mathbf{y}}\|_2 \\ &= \left\| \begin{bmatrix} 0.8944 \\ -1.0955 \end{bmatrix} \right\|_2 \\ &= 1.4142. \end{aligned}$$

Figure 1 depicts the effect of the LLL algorithm. The top lattice grid is generated by the basis formed by the columns of $R$ in Example 2. The bottom lattice grid is generated by the basis formed by the columns of $\widehat{R}$ in Example 2. The figure shows that the LLL algorithm effectively makes a lattice grid more orthogonal by reducing the off diagonal.

Now we consider the effect of the second condition in Definition 1. The sphere decoding Algorithm 1 performs a depth-first search for the lattice points inside a sphere. The second condition roughly imposes an ascending order on the diagonal elements of $R$. The parameter $\omega$, usually set to 0.75, controls the degree of the strictness of the ascending order. Thus we expect that the last entry $\hat{r}_{m,m}$ in the reduced $\widehat{R}$ be relatively large. Consequently, number of the integers in (7) is reduced. Thus the number of starting paths is reduced. Reducing the number of search paths in the early steps of a depth first search can significantly reduce the total number of search paths, thus the computational complexity.

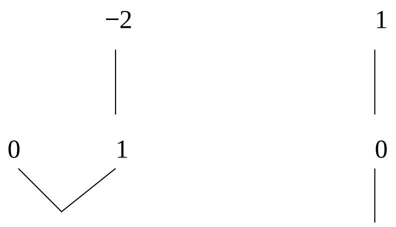

**Figure 2: The search trees for $R$, on the left, and for the reduced $\widehat{R}$, on the right, in Example 3.**

Example 3. *Continue Example 2 and apply Algorithm 1. For $R$, the search bounds for $x_2$ are*

$$\left\lceil \frac{-1.7321+0.6547}{1.9640} \right\rceil = 0 \quad and \quad \left\lfloor \frac{1.7321+0.6547}{1.9640} \right\rfloor = 1.$$

*There are two integers $0, 1$ to be searched for $x_2$. For the reduced $\widehat{R}$, the search bounds are*

$$\left\lceil \frac{-1.4142+1.0954}{3.2863} \right\rceil = 0 \quad and \quad \left\lfloor \frac{1.4142}{3.2863} \right\rfloor = 0.$$

*There is only one integer $0$ to be searched for $\hat{x}_2$. Figure 2 shows the two search trees.*

Figure 2 shows the two search trees. The one on the left corresponds to the search tree when Algorithm 1 is applied to $R$. The one on the right corresponds to $\widehat{R}$, which shows the solution $[1\ 0]^{\mathrm{T}}$. Applying the unimodular $M$ computed by the LLL algorithm in Example 2, we get

$$M \begin{bmatrix} 1 \\ 0 \end{bmatrix} = \begin{bmatrix} -2 & 3 \\ 1 & -1 \end{bmatrix} \begin{bmatrix} 1 \\ 0 \end{bmatrix} = \begin{bmatrix} -2 \\ 1 \end{bmatrix},$$

which is the integer least squares solution for the original $R$ and $\hat{\mathbf{y}}$.

## 4. EXPERIMENTS

In this section, we present our preliminary experimental results. We consider a setting in communications, where the $n$-by-$m$ matrix $A$ in (2) is a channel matrix of the following Toeplitz structure:

$$A = \begin{bmatrix} a_1 & & 0 \\ a_2 & \ddots & \\ \vdots & \ddots & a_1 \\ a_k & \ddots & a_2 \\ & \ddots & \vdots \\ 0 & & a_k \end{bmatrix},$$

where $a_i$, $i = 1, ..., k$, are the parameters of the channel and $k$ is the order of the channel. Thus $n = m + k - 1$. In our experiments we set $k = 3$. The entries of the original signal vector $\mathbf{x}$ are randomly chosen from $\{\pm 1, \pm 3\}$. The length $m$ of the signal is set to 4. The received signal vector $\mathbf{y} = A\mathbf{x} + \mathbf{v}$, where $\mathbf{v}$ is a white noise vector, whose entries are random variables normally distributed with zero mean. The channel matrix $A$ is normalized according to power constraint and the noise $\mathbf{v}$ is scaled based on the given signal to noise ratio (snr) 20dB. We summarize the parameters:

| | radius $\hat{\rho}$ | running time (sec.) | failure rate (%) |
|---|---|---|---|
| without LLL | 0.3834 | 1.155 | 1.155 |
| with LLL | 0.3834 | 1.117 | 1.155 |

**Table 1: Comparison of the combination of Algorithms 2 and 1 without and with the LLL algorithm.**

- Original signal length $m$: 4
- Entries of signal vector: $\pm 1, \pm 3$
- Channel order $k$: 3
- Signal to noise ratio (snr): 20dB

We programmed the algorithms in MATLAB. In our experiments, 10 random channel matrices were constructed by generating random entries $a_i$ uniformly distributed over $[-1, 1]$. For each channel matrix, 200 random signal vectors $\mathbf{x}$ and 200 white noise vectors $\mathbf{v}$ were generated. The received signal vector $\mathbf{y} = A\mathbf{x} + \mathbf{v}$.

**Case 1**. Combination of the statistical Algorithm 2 for choosing a radius of the search sphere and the triangular sphere decoding Algorithm 1:

1. Calculate radius $\rho$ using Algorithm 2 from the given snr = 20 and the probability $p = 0.99$;
2. QR decomposition $A = Q_1 R$ and update $\hat{\mathbf{y}} = Q_1^{\mathrm{T}}\mathbf{y}$;
3. Optional: Apply the LLL algorithm to reduce $R$, where $\omega = 0.75$;
4. Adjust the radius $\hat{\rho}^2 = \rho^2 - (\|\mathbf{y}\|_2^2 - \|\hat{\mathbf{y}}\|_2^2)$; (Note that $\|\mathbf{y}\|_2^2 - \|\hat{\mathbf{y}}\|_2^2$ is the least squares residual norm squared.)
5. Apply the triangular sphere decoding Algorithm 1 to $R$, $\hat{\mathbf{y}}$, and $\hat{\rho}$.

The value of $\omega$ was set to 0.75. When $\omega$ is close to 1, the LLL algorithm imposes a nearly strict ascending order on the diagonal elements. However, the LLL algorithm requires more computation as $\omega$ approaches 1, the computing cost of the LLL algorithm increases rapidly. On the other hand, when $\omega$ is close to 0.25, the LLL algorithm imposes a lose ascending order on the diagonal elements. Consequently, its impact on reducing the total number of search paths diminishes. As a compromise, $\omega$ is commonly set to 0.75.

The values in Table 1 are the averages of the 10 channels and 200 signals for each channel. Table 1 shows that the LLL algorithm improved the performance by about 3.3%, although the radius was not reduced in this application, because $A$ has the Toeplitz structure and its entries are uniformly distributed over $[-1, 1]$. In other words, the norm of $A$ is small.

**Case 2**. Combination of our deterministic Algorithm 3 for finding the radius of the search sphere and the triangular sphere decoding Algorithm 1:

1. QR decomposition $A = Q_1 R$ and update $\hat{\mathbf{y}} = Q_1^{\mathrm{T}}\mathbf{y}$;
2. Optional: Apply the LLL algorithm to reduce $R$, where $\omega = 0.75$;
3. Calculate radius $\hat{\rho}$ using Algorithm 3;

| | radius $\hat{\rho}$ | running time (sec.) | failure rate (%) |
|---|---|---|---|
| without LLL | 0.1891 | 0.605 | 0 |
| with LLL | 0.1891 | 0.584 | 0 |

**Table 2: Comparison of the combination of Algorithms 3 and 1 without and with the LLL algorithm.**

4. Apply the triangular sphere decoding Algorithm 1 to $R$, $\hat{\mathbf{y}}$, and $\hat{\rho}$.

Again, the values in Table 2 are the averages of the 10 channels and 200 signals for each channel. Table 2 shows that the LLL algorithm improved the performance by about 3.5%, although the radius was not reduced, in this application.

Since in this application, the LLL algorithm has no effect on reducing the search radius due to the characteristics of the channel matrix, the improvement caused by the LLL algorithm is due to the reduction of the number of the search paths. Note that in our experiments, the length $m$ of the original signal vector was set to 4, which was the depth of the search trees. We expect that a larger $m$ will show more significant improvement.

Comparing Table 2 with Table 1 shows that our Algorithm 3 chose a much smaller radius than the statistical Algorithm 2. Consequently, the combination of the LLL algorithm and our method for selecting search radius improved the running time by almost 50%, while achieving zero failure rate.

## 5. CONCLUSION

In this paper, we considered the integer least squares problem, the sphere decoding method in communications in particular. We addressed two key issues in sphere decoding performance: Finding a radius of search sphere and reducing the total number of search paths. We proposed a deterministic method for finding a radius. We showed the impact of the LLL algorithm on reducing the number of search paths. Our preliminary experiments demonstrated that our method found accurate search radii while achieving zero failure rate. The combination of our method for finding search radius and the LLL algorithm can significantly improve the performance of sphere decoding in communications. Our future work includes extensive experiments and investigation of numerical properties.

## 6. REFERENCES

[1] O. Goldreich, S. Goldwasser, and S. Halevi. Public-key cryptosystems from lattice reduction problems. In *Advances in Cryptology - CRYPTO97*, pages 112–31. 17th Ann. Int. Crypto. Conf., 1997.

[2] G.H. Golub and C.F. Van Loan. *Matrix Computations 3rd Ed.* The Johns Hopkins University Press, Baltimore, MD, 1996.

[3] A. Hassibi and S. Boyd. Integer parameter estimation in linear models with applications to GPS. *IEEE Transactions on Signal Processing*, 46:2938–52, 1998.

[4] B. Hassibi and H. Vikalo. On the sphere-decoding algorithm I: Expected complexity. *IEEE Transactions on Signal Processing*, 53:2806–2818, 2005.

[5] A. Lenstra, H. Lenstra, and L. Lovasz. Factorizing polynomials with rational coefficients. *Mathematicsche Annalen*, 261:515–534, 1982.

[6] F. T. Luk and D. M. Tracy. An improved LLL algorithm. *Linear Algebra and Its Applications*, 428/2-3:441–452, 2008.

[7] Franklin T. Luk and Sanzheng Qiao. Numerical properties of the LLL algorithm. In Franklin T. Luk, editor, *Advance Signal Processing Algorithms, Architectures, and Implementations XVII*, volume 6697 of *Proceedings of SPIE*, pages 6697–3. The International Society for Optical Engineering, 2007.