J. KSIAM Vol.19, No.2, 171–188, 2015 http://dx.doi.org/10.12941/jksiam.2015.19.171

EFFICIENT LATTICE REDUCTION UPDATING AND DOWNDATING METHODS AND ANALYSIS

JAEHYUN PARK[1†] AND YUNJU PARK[2]

[1]DEPARTMENT OF ELECTRONIC ENGINEERING, PUKYONG NATIONAL UNIVERSITY, SOUTH KOREA
E-mail address: jaehyun@pknu.ac.kr

[2]DEPARTMENT OF MATHEMATICS AND COMPUTER SCIENCE, KOREA SCIENCE ACADEMY OF KOREA ADVANCED INSTITUTE OF SCIENCE AND TECHNOLOGY, SOUTH KOREA
E-mail address: yunjupark@kaist.ac.kr

ABSTRACT. In this paper, the efficient column-wise/row-wise lattice reduction (LR) updating and downdating methods are developed and their complexities are analyzed. The well-known *LLL* algorithm, developed by Lenstra, Lenstra, and Lovász, is considered as a LR method. When the column or the row is appended/deleted in the given lattice basis matrix $\mathbf{H}$, the proposed updating and downdating methods modify the preconditioning matrix that is primarily computed for the LR with $\mathbf{H}$ and provide the initial parameters to reduce the updated lattice basis matrix efficiently. Since the modified preconditioning matrix keeps the information of the original reduced lattice bases, the redundant computational complexities can be eliminated when reducing the lattice by using the proposed methods. In addition, the rounding error analysis of the proposed methods is studied. The numerical results demonstrate that the proposed methods drastically reduce the computational load without any performance loss in terms of the condition number of the reduced lattice basis matrix.

1. INTRODUCTION

Lattice reduction (LR) is a method to find the bases of the given lattice space close to the shortest vector, which has been successfully applied to cryptography, factoring the polynomials ([1–3] and references therein), and multiple-input multiple output (MIMO) communication system [4–9].

A well-known LR method due to its simplicity and efficiency is the LLL algorithm developed by Lenstra, Lenstra, and Lovász [1] which has the complexity of $O(N^4 \log B)$ multiplications (or equivalently, $O(N^5(\log B)^2)$ bit operations) for a lattice basis matrix $\mathbf{H}(\in \mathbb{R}^{M\times N})$ where B is the maximum in the squared norm of columns of $\mathbf{H}$.

Received by the editors February 12 2015; Revised May 6 2015; Accepted in revised form May 6 2015; Published online June 11 2015.

2000 *Mathematics Subject Classification.* 15B99.

Key words and phrases. Lattice Reduction, LLL algorithm, Matrix Updating/Downdating.

This work was supported by a Research Grant of Pukyong National University (2014 year).

† Corresponding author.

During recent decades, several efficient algorithms have been developed which has less computational complexities than the LLL algorithm. For example, the LLL using the fraction-free Gaussian elimination and modular operation was proposed by Storjohann [10], which costs $O\left(N^4 (\log B)^2\right)$ bit operations. Koy and Schnorr proposed the segment LLL-reduction method, whose complexity is $O\left(N^4 \log B + N^3 (\log B)^2\right)$ bit operations [11, 12]. However, to the best of the authors' knowledge, the LLL updating and downdating methods when a column or a row is updated/downdated in a given lattice basis matrix have not been addressed so far. This LLL updating/downdating could be important in many applications; For example, in multi-user (MU) MIMO precoding system when new users enter the current system or existing users leave the system, the LLL algorithm should be newly run to find the preconditioning matrix for the updated channel matrix (lattice basis matrix) which could be a major bottleneck in LR aided MU MIMO precoding [9].

In this paper, we propose efficient LLL updating and downdating methods when the column-wise or row-wise modification occurs in the given lattice basis matrix. In the proposed methods, the preconditioning matrix for the original basis matrix is modified to provide the suitable initial parameters to reduce the newly updated lattice basis matrix efficiently. Because the modified preconditioning matrix keeps the information of the original reduced lattice bases, the proposed methods require less computational complexities than the conventional LLL. Moreover, the rounding error analysis for the proposed methods is studied, based on the previously developed results [13–17], showing that the proposed algorithms are numerically stable. In the numerical results, the condition numbers of the reduced lattice basis matrices are evaluated when the proposed methods are applied and their computational complexities are also compared with those without updating/downdating methods.

The rest of this paper is organized as follows. In Section 2 the LLL algorithm is briefly reviewed. In Section 3 the column-wise LLL updating and downdating methods are proposed and their complexities are analyzed. In Section 4 the row-wise LLL updating and downdating methods are developed. In Section 5 the rounding error analysis of the proposed methods is provided. Several simulation results for various conditions are presented in Section 6. Concluding remarks are made in Section 7.

2. Lattice Reduction: LLL algorithm

Lattice $\mathcal{L}(\mathbf{H})$ is defined as $\sum_{n=1}^{N} \mathbf{h}_n s_n$, where $\mathbf{h}_n$, the nth column vector of $\mathbf{H} \in \mathbb{R}^{M\times N}$ where $M \geq N$, is a basis of the lattice and $s_n \in \mathbb{Z}$ [1]. A lattice basis matrix $\mathbf{H}$ is LLL reduced if

$$|\mu_{ij}| \leq 1/2 \text{ for } 1 \leq i < j \leq N \tag{2.1}$$

and

$$3/4\|\mathbf{h}_{i-1}^o\|_2^2 \leq \|\mathbf{h}_i^o + \mu_{i-1i}\mathbf{h}_{i-1}^o\|_2^2 \text{ for } 1 < i \leq N, \tag{2.2}$$

TABLE 1. LLL algorithm

```
1   $\mathbf{T} = \mathbf{I}_N$, $\mathbf{H} = \mathbf{QR}$
2   $i = 2$
3   while $i \leq N$
4    for $l = i-1, ..., 1$
5    [$\mathbf{R}$, $\mathbf{T}$] = Size-reduction($\mathbf{R}$, $\mathbf{T}$, $i$, $l$)
6    end
7    if $3/4r^2_{i-1i-1} > r^2_{ii} + r^2_{i-1i}$
8    [$\mathbf{R}$, $\mathbf{T}$] = two-reduction($\mathbf{R}$, $\mathbf{T}$, $i$ )
9    $i = \max\{i-1,\ 2\}$
10   else
11   $i = i+1$
12   end
13  end
```

where $\mathbf{h}^o_j = \mathbf{h}_j - \sum_{i=1}^{j-1} \mu_{ij}\mathbf{h}^o_i$ with $\mu_{ij} = \frac{\mathbf{h}_i^T \mathbf{h}_j^o}{\mathbf{h}_i^{oT}\mathbf{h}_i^o}$ and $(\cdot)^T$ represents the matrix transposition. Equivalently, a lattice basis matrix $\mathbf{H}$ with QR decomposition $\mathbf{H} = \mathbf{QR}$, where $\mathbf{Q}^T\mathbf{Q} = \mathbf{I}_N$, $\mathbf{Q} \in \mathbb{R}^{M\times N}$, and $\mathbf{R} \in \mathbb{R}^{N\times N}$ is upper triangular, is LLL reduced if

$$|r_{ij}| \leq 1/2|r_{ii}| \text{ for } 1 \leq i < j \leq N \tag{2.3}$$

and

$$3/4r^2_{i-1i-1} \leq r^2_{ii} + r^2_{i-1i} \text{ for } 1 < i \leq N, \tag{2.4}$$

where r_{ij} is the (i,j)th element of $\mathbf{R}$. Here, $\mathbf{I}_N$ denotes an $N \times N$ identity matrix. Note that $\mu_{ij} = \frac{r_{ij}}{r_{ii}}$. If Eqn. (2.1) (equivalently, Eqn. (2.3)) is satisfied, $\mathbf{H}$ is said to be *size-reduced* and if Eqn. (2.2) (equivalently, Eqn. (2.4)) is satisfied, $\mathbf{H}$ is said to be *two-reduced*. The constant $\frac{3}{4}$ in Eqn. (2.2) and Eqn. (2.4) could be arbitrarily replaced by any fixed real number within $(1/4, 1)$ [1].

The LLL algorithm finds a unimodular matrix $\mathbf{T}$ such that $\mathbf{HT}$ is LLL reduced, i.e.,

$$\mathbf{G} = \mathbf{HT}, \tag{2.5}$$

where $\mathbf{T}$ is a unimodular matrix, a square matrix with integer entries such that $|\det(\mathbf{T})| = 1$, and $\mathbf{G}$ is lattice reduced. The LLL algorithm along with two functions, size-reduction and two-reduction [1, 7], is rephrased in Table 1 and 2. In size-reduction step, by a modular operation, Eqn. (2.3) can be satisfied. In two-reduction step, if Eqn. (2.4) is not satisfied, the corresponding adjacent columns are swapped and the triangular form of matrix $\mathbf{R}$ is recovered by the Givens rotation. Since the Givens rotation preserves the norm of each column, Eqn. (2.4) can be satisfied by simply swapping the columns.

To evaluate the computational complexity of LLL algorithm, the following lemma [18] will be useful.

TABLE 2. Functions of LLL algorithm

	$[\mathbf{R}, \mathbf{T}]$= Size-reduction($\mathbf{R}$, $\mathbf{T}$, i, l)
1	$\mu = \lceil r_{l,i}/r_{l,l} \rfloor$
2	if $\mu \neq 0$
3	$\mathbf{R}_{1:l,i} = \mathbf{R}_{1:l,i} - \mu \mathbf{R}_{1:l,l}$
4	$\mathbf{T}_{1:\text{end},i} = \mathbf{T}_{1:\text{end},i} - \mu \mathbf{T}_{1:\text{end},l}$
5	end
	$[\mathbf{R}, \mathbf{T}]$= two-reduction($\mathbf{R}$, $\mathbf{T}$, i)
1	Swap columns $i-1$ and i in $\mathbf{R}$ and $\mathbf{T}$
2	Triangularize $\mathbf{R}$ using Givens rotation matrix $\mathbf{\Theta}$
3	$\mathbf{R}_{i-1:i,i-1:\text{end}} = \mathbf{\Theta}\mathbf{R}_{i-1:i,i-1:\text{end}}$

Lemma 1. *If we define $D = r_{11}^{2N} \cdot r_{22}^{2(N-1)} \cdot ... \cdot r_{NN}^2$, the size-reduction step does not affect D, while the two-reduction step decreases D by a factor of at least 3/4.*

Therefore, the number of $while$ loop iterations in the algorithm is at most $O(\log D_0)$ where $D_0 = \|\mathbf{h}_1\|^{2N} \cdot \|\mathbf{h}_2\|^{2(N-1)} \cdot ... \cdot \|\mathbf{h}_N\|^2$. If we set $B = \max\{\|\mathbf{h}_1\|^2, ..., \|\mathbf{h}_N\|^2\}$, then $O(\log D_0) \simeq O(N^2 \log B)$. Since the size-reduction step in lines $4-6$ of Table 1 requires only $O(N^2)$ multiplications, total $O(N^4 \log B)$ multiplications are required. Each real number used during the process of the algorithm is bounded by $O(N \log B)$ bits [1]. Hence, total $O(N^5 (\log B)^2)$ bit operations are required in the LLL algorithm. Note that because the complexity of LLL algorithm depends directly on the number of while loop iterations, we would focus only on the number of the two-reduction steps as a measure of the computational complexity.

Remark 1. *The LLL algorithm depends only on the upper triangular matrix* $\mathbf{R}$ *in Table 1. That is, the output of LLL algorithm is invariant on the pre-multiplication of the orthogonal matrix. Therefore, the updating and downdating algorithms are developed by modifying* $\mathbf{R}$ *in the following sections.*

3. LLL COLUMN-WISE UPDATING AND DOWNDATING

3.1. **Column-wise Updating.**

Let $\mathbf{H} \in \mathbb{R}^{M \times N}, M \geq N$ have a QR decomposition as

$$\mathbf{H} = \mathbf{QR}, \tag{3.1}$$

where $\mathbf{Q}^T\mathbf{Q} = \mathbf{I}_N$ and $\mathbf{R} \in \mathbb{R}^{N \times N}$ is upper triangular and assume that, given $\mathbf{H}$, we have the preconditioning matrix $\mathbf{T} \in \mathbb{Z}^{N \times N}$ which is a unimodular matrix computed by the LLL algorithm. Here, the QR decomposition of $\mathbf{HT} \in \mathbb{R}^{M \times N}$ is also given after completing the LLL algorithm as:

$$\mathbf{G} = \mathbf{HT} = \tilde{\mathbf{Q}}\tilde{\mathbf{R}}, \quad \tilde{\mathbf{R}} \in \mathbb{R}^{N \times N} \tag{3.2}$$

where $\tilde{\mathbf{R}}$ satisfies Eqns. (2.3) and (2.4). We then want to find the new preconditioning matrix $\mathbf{T}_u$ after a new column $\mathbf{h}_c$ is added:

$$\mathbf{G}_u = \mathbf{H}_u\mathbf{T}_u = [\mathbf{H}\ \mathbf{h}_c]\mathbf{T}_u, \tag{3.3}$$

where $\mathbf{G}_u$ is lattice reduced.

First, we define the upper triangular matrix $\tilde{\mathbf{R}}_u^{in} \in \mathbb{R}^{(N+1)\times(N+1)}$ which is updated from $\tilde{\mathbf{Q}}$ and $\tilde{\mathbf{R}}$ in Eqn. (3.2) as

$$\tilde{\mathbf{R}}_u^{in} = \begin{bmatrix} \tilde{\mathbf{R}} & \mathbf{w} \\ \mathbf{0}_{1\times N} & \|\mathbf{q}\|_2 \end{bmatrix}, \tag{3.4}$$

where $\mathbf{0}_{M\times N}$ denotes a zero $M \times N$ matrix and

$$\mathbf{w} = \tilde{\mathbf{Q}}^T\mathbf{h}_c, \quad \mathbf{q} = \mathbf{h}_c - \tilde{\mathbf{Q}}\mathbf{w}, \tag{3.5}$$

so that $\tilde{\mathbf{R}}_u^{in}$ holds the information about the reduced lattice basis structure for $\mathbf{H}$ and $[\mathbf{G}\ \mathbf{h}_c] = \tilde{\mathbf{Q}}_u^{in}\tilde{\mathbf{R}}_u^{in}$ with $(\tilde{\mathbf{Q}}_u^{in})^T\tilde{\mathbf{Q}}_u^{in} = \mathbf{I}_{N+1}$. The corresponding $\mathbf{T}_u^{in}$ is then given as

$$\mathbf{T}_u^{in} = \begin{bmatrix} \mathbf{T} & \mathbf{0}_{N\times 1} \\ \mathbf{0}_{1\times N} & 1 \end{bmatrix}, \tag{3.6}$$

such that $[\mathbf{G}\ \mathbf{h}_c] = [\mathbf{H}\ \mathbf{h}_c]\mathbf{T}_u^{in}$. Therefore, the preconditioning matrix $\mathbf{T}_u$ can be efficiently obtained from $\tilde{\mathbf{R}}_u^{in}$ and $\mathbf{T}_u^{in}$, rather than by applying the LLL algorithm to $[\mathbf{H}\ \mathbf{h}_c]$ from scratch. To complete the updating algorithm, we only need to replace the line 1 of Table 1 with

$$\mathbf{T} = \mathbf{T}_u^{in}, \quad \mathbf{R} = \tilde{\mathbf{R}}_u^{in}. \tag{3.7}$$

To see the computational complexity, the value D in Lemma 1 indicating the number of the required two-reduction steps is evaluated for the updating algorithm. Let $D_N = r_{11}^{2N} r_{22}^{2(N-1)} \cdots r_{NN}^2$ from $\mathbf{R}$ in Eqn. (3.1). If a new column $\mathbf{h}_u$ is updated from $\mathbf{Q}$ and $\mathbf{R}$, D_{N+1} for $[\mathbf{H}\ \mathbf{h}_c]$ is given as

$$\begin{aligned} D_{N+1} &= r_{11}^{2(N+1)} r_{22}^{2N} \cdots r_{N+1N+1}^2 \\ &= D_N \cdot r_{11}^2 r_{22}^2 \cdots r_{N+1N+1}^2 \\ &= D_N \cdot \det([\mathbf{H}\ \mathbf{h}_c]^T[\mathbf{H}\ \mathbf{h}_c]). \end{aligned} \tag{3.8}$$

According to Lemma 1, the numbers of two-reduction steps carried out when applying the conventional LLL to $[\mathbf{H}\ \mathbf{h}_c]$ and $\mathbf{H}$ are, respectively, $O(\log(D_{N+1}))$ and $O(\log(D_N))$. Therefore, because the first N columns of $\tilde{\mathbf{R}}_u^{in}$ represents the already reduced bases for $\mathbf{H}$, the number of the required two-reduction steps when adopting the updating method becomes $O(\log(\det([\mathbf{H}\ \mathbf{h}_c]^T[\mathbf{H}\ \mathbf{h}_c])))$.

3.2. **Column-wise Downdating.**

In this section, given $\mathbf{H}$ in Eqn. (3.1) with preconditioning matrix $\mathbf{T}$, we find the new preconditioning matrix $\mathbf{T}_d$ for a submatrix $\mathbf{H}_d$ where $\mathbf{H} = [\mathbf{H}_d\ \mathbf{h}_N]$ and $\mathbf{h}_N \in \mathbb{R}^{M\times 1}$. Note that the QR decomposition of $\mathbf{HT}$ is also given as in Eqn. (3.2).

Downdating $\mathbf{h}_N$ from $\mathbf{H}$ can be represented as:

$$\mathbf{H} - \mathbf{h}_N\mathbf{e}_N^T = \mathbf{Q}'\mathbf{R}' = \begin{bmatrix} \mathbf{H}_d & \mathbf{0}_{M\times 1} \end{bmatrix}, \tag{3.9}$$

where $\mathbf{Q}' = \mathbf{Q}$ and $\mathbf{R}' = \begin{bmatrix} \mathbf{R}(1:N, 1:N-1) & \mathbf{0}_{N\times 1} \end{bmatrix}$. Here, $\mathbf{A}(i:j, k:l)$ denotes the submatrix of $\mathbf{A}$ with elements from the ith row to the jth row and from the kth column to the lth column. Similarly, to downdate $\mathbf{h}_N$ from $\mathbf{G}$, the influence of $\mathbf{h}_N$ on $\mathbf{G}$ should be removed, which induces

$$\begin{aligned} \mathbf{G} - \mathbf{h}_N \mathbf{t}^{(N)T} &= \tilde{\mathbf{Q}}\tilde{\mathbf{R}} - \mathbf{h}_N \mathbf{t}^{(N)T} \\ &= \tilde{\mathbf{Q}}'\tilde{\mathbf{R}}', \end{aligned} \tag{3.10}$$

where $\mathbf{t}^{(N)T}$ is the Nth row of $\mathbf{T}$, $\tilde{\mathbf{Q}}'^T\tilde{\mathbf{Q}}' = \mathbf{I}_N$, and $\tilde{\mathbf{R}}'$ is upper triangular. Here, $\tilde{\mathbf{R}}'$ can be computed by utilizing the QR rank-one changing method [19], which leads

$$\tilde{\mathbf{R}}' = \left[\underbrace{\begin{matrix} r'_{11} & \cdots & r'_{1i'-1} \\ 0 & \ddots & \vdots \\ & & r'_{i'-1i'-1} \\ & & 0 \\ \vdots & \ddots & \vdots \\ 0 & & \\ 0 & \cdots & 0 \end{matrix}}_{\tilde{\mathbf{R}}'_a \in \mathbb{R}^{N\times(i'-1)}} \middle| \begin{matrix} r'_{1i'} \\ \vdots \\ r'_{i'-1i'} \\ 0 \\ 0 \\ \vdots \\ 0 \end{matrix} \middle| \underbrace{\begin{matrix} r'_{1i'+1} & \cdots & r'_{1N} \\ \vdots & & \vdots \\ r'_{i'i'+1} & & \\ r'_{i'+1i'+1} & \ddots & \vdots \\ 0 & \ddots & r'_{NN-1} \\ & & r'_{NN} \end{matrix}}_{\tilde{\mathbf{R}}'_b \in \mathbb{R}^{N\times(N-i')}} \right], \quad rank(\tilde{\mathbf{R}}') = N-1 \tag{3.11}$$

for some i', $1 \le i' \le N$. Since $\det(\mathbf{T}) = \pm 1$ and the rank-one deficiency in both $\mathbf{R}'$ and $\tilde{\mathbf{R}}'$ arises from downdating $\mathbf{h}_N$, by deleting the corresponding columns causing the rank-one deficiency in both of $\mathbf{R}'$ and $\tilde{\mathbf{R}}'$ (that is, the Nth column of $\mathbf{R}'$ and the i'th column of $\tilde{\mathbf{R}}'$), it can be found that

$$\det\left(\tilde{\mathbf{R}}''^T\tilde{\mathbf{R}}''\right) = \det\left(\mathbf{R}'(1:N, 1:N-1)^T\mathbf{R}'(1:N, 1:N-1)\right) \tag{3.12}$$

where $\tilde{\mathbf{R}}'' = \begin{bmatrix} \tilde{\mathbf{R}}'_a & \tilde{\mathbf{R}}'_b \end{bmatrix}$. Now let us define the correspondingly partitioned matrices:

$$\mathbf{T} = \left[\mathbf{T}_a \; \mathbf{t}'_i \; \mathbf{T}_b\right], \quad \mathbf{T}_a \in \mathbb{Z}^{N\times(i'-1)}, \quad \mathbf{T}_b \in \mathbb{Z}^{N\times(N-i')}. \tag{3.13}$$

Because

$$\begin{aligned} \tilde{\mathbf{Q}}'\tilde{\mathbf{R}}' &= \tilde{\mathbf{Q}}\tilde{\mathbf{R}} - \mathbf{h}_N\mathbf{t}^{(N)T} \\ &= \tilde{\mathbf{Q}}\tilde{\mathbf{R}}\mathbf{T}^{-1}\mathbf{T} - \mathbf{h}_N\mathbf{t}^{(N)T}\mathbf{T}^{-1}\mathbf{T} \\ &= (\mathbf{H} - \mathbf{h}_N\mathbf{e}_N^T)\mathbf{T} \\ &= \mathbf{Q}'\mathbf{R}'\mathbf{T}, \end{aligned} \tag{3.14}$$

the following equation can then be derived:

$$\tilde{\mathbf{Q}}'\tilde{\mathbf{R}}'' = \mathbf{Q}'\mathbf{R}'\mathbf{T}', \tag{3.15}$$

where $\mathbf{T}' = \begin{bmatrix} \mathbf{T}_a & \mathbf{T}_b \end{bmatrix}$. Since the last column of $\mathbf{R}'$ is $\mathbf{0}_{N\times 1}$,

$$\tilde{\mathbf{Q}}'\tilde{\mathbf{R}}'' = \mathbf{H}_d\mathbf{T}'', \tag{3.16}$$

where $\mathbf{T}'' = \mathbf{T}'(1 : N-1, 1 : N-1)$. Since $\mathbf{T}'' \in \mathbb{Z}^{N-1\times N-1}$ and $\det(\mathbf{T}'') = \pm 1$ from Eqn.(3.12), $\mathbf{T}''$ is also unimodular. Here, to recover QR formula in the left side hand of Eqn. (3.16), a total of $N - i' - 1$ Givens rotations are required as:

$$(\tilde{\mathbf{Q}}'\mathbf{J}_{i'}^T \cdots \mathbf{J}_{N-2}^T)\,(\mathbf{J}_{N-2} \cdots \mathbf{J}_{i'}\tilde{\mathbf{R}}'') \;=\; \mathbf{H}_d\mathbf{T}'', \tag{3.17}$$

where $\mathbf{J}_i$ is the Givens rotation matrix forcing zero on the $(i+1, i)$th entry of $\tilde{\mathbf{R}}''$ which has Hessenberg form. Therefore, let $\tilde{\mathbf{R}}_d^{in} = \mathbf{J}_{N-2} \cdots \mathbf{J}_{i'}\tilde{\mathbf{R}}''$ and $\mathbf{T}_d^{in} = \mathbf{T}''$ then the preconditioning matrix $\mathbf{T}_d$ can be computed by using $\tilde{\mathbf{R}}_d^{in}$ and $\mathbf{T}_d^{in}$ as initial parameters, similarly to Eqn. (3.7).

To see the complexity for the downdating method, the value D, defined in Lemma 1, is evaluated. From Eqn. (3.10),

$$(\tilde{r}_{11}^{in})^2 \;\leq\; (|\tilde{r}_{11}| + \|\mathbf{h}_N\|\|\mathbf{t}^{(N)}\|)^2, \tag{3.18}$$

where $\tilde{r}_{ii}^{in}$ is the ith diagonal element of $\tilde{\mathbf{R}}_d^{in}$. Let

$$D_n \;=\; \prod_{k=1}^{N-1} (\tilde{r}_{kk}^{in})^{2(N-k)},$$

then, to get the upper bound of D_n, we set $(\tilde{r}_{11}^{in})^2 = (|\tilde{r}_{11}| + \|\mathbf{h}_N\|\|\mathbf{t}^{(N)}\|)^2$ from Eqn. (3.18) which implies that $\mathbf{h}_N$ and $\mathbf{t}^{(N)T}$ in Eqn. (3.10) should have the following form:

$$\mathbf{h}_N = \pm\|\mathbf{h}_N\|\tilde{\mathbf{q}}_1, \quad \mathbf{t}^{(N)T} = \pm\|\mathbf{t}^{(N)}\|\mathbf{e}_1^T, \tag{3.19}$$

where their signs depends on the sign of $\tilde{r}_{11}$. Then, D_n has the following inequality:

$$D_n \;\leq\; \left(1 + \frac{\|\mathbf{h}_N\|\|\mathbf{t}^{(N)}\|}{|\tilde{r}_{11}|}\right)^{2(N-1)} \prod_{k=1}^{N-1} \tilde{r}_{kk}^{2(N-k)}. \tag{3.20}$$

Since $\tilde{r}_{kk}$ is from the reduced basis matrix, D_n is lower bounded by $\prod_{k=1}^{N-1} \tilde{r}_{kk}^{2(N-k)}$. Therefore, the number of two-reduction steps required in the downdating method is at most $O(\log D_d)$, where

$$D_d \;=\; \left(1 + \frac{\|\mathbf{h}_N\|\|\mathbf{t}^{(N)}\|}{|\tilde{r}_{11}|}\right)^{2(N-1)}, \tag{3.21}$$

indicating that the complexity depends mainly on the contribution of downdated column $\mathbf{h}_N$ on the reduced lattice basis (represented by the norm of $\mathbf{t}^{(N)}$), that is, if $\|\mathbf{h}_N\|\|\mathbf{t}^{(N)}\|$ is relatively small to the norm of the shortest lattice basis ($|\tilde{r}_{11}|$), then the required computation overhead also becomes small.

4. LLL Row-wise Updating and Downdating

4.1. **Row-wise Updating.** In this section, given $\mathbf{H}$ in Eqn. (3.1) with the preconditioning matrix $\mathbf{T}$, we find the new preconditioning matrix $\mathbf{T}_u$ after a new row $\mathbf{h}_r^T$ is added:

$$\mathbf{G}_u = \mathbf{H}_u\mathbf{T}_u = \begin{bmatrix} \mathbf{h}_r^T \\ \mathbf{H} \end{bmatrix}\mathbf{T}_u, \quad \mathbf{h}_r^T \in \mathbb{R}^{1\times N} \tag{4.1}$$

where $\mathbf{G}_u$ is lattice reduced. Since the QR decomposition of $\mathbf{HT}$ is known as Eqn. (3.2), the new row $\mathbf{h}_r^T$ can be updated based on $\tilde{\mathbf{Q}}$ and $\tilde{\mathbf{R}}$ rather than using $\mathbf{Q}$ and $\mathbf{R}$ as follows:

$$\mathbf{G}' = \begin{bmatrix} \mathbf{h}_r^T\mathbf{T} \\ \mathbf{HT} \end{bmatrix} = \begin{bmatrix} \mathbf{h}_r^T\mathbf{T} \\ \tilde{\mathbf{Q}}\tilde{\mathbf{R}} \end{bmatrix} = \begin{bmatrix} 1 & \mathbf{0}_{1\times N} \\ \mathbf{0}_{N\times 1} & \tilde{\mathbf{Q}} \end{bmatrix}\begin{bmatrix} \mathbf{h}_r^T\mathbf{T} \\ \tilde{\mathbf{R}} \end{bmatrix}, \tag{4.2}$$

where $\begin{bmatrix} \mathbf{h}_r^T\mathbf{T} \\ \tilde{\mathbf{R}} \end{bmatrix}$ has a Hessenberg form, which can be recovered into the triangular form by using $N-1$ Givens rotations as:

$$\mathbf{J}_{N-1}\cdots\mathbf{J}_1\begin{bmatrix} \mathbf{h}_r^T\mathbf{T} \\ \tilde{\mathbf{R}} \end{bmatrix} = \tilde{\mathbf{R}}_u^{in}. \tag{4.3}$$

The preconditioning matrix $\mathbf{T}_u$ can then be computed by using $\tilde{\mathbf{R}}_u^{in}$ and $\mathbf{T}$ as initial parameters.

Here, to analyze the upper bound of the computational complexity, the process of applying the Givens rotation to the Hessenberg matrix of Eqn. (4.3) is investigated in more detail.

Let

$$\begin{bmatrix} \mathbf{h}_r^T\mathbf{T} \\ \tilde{\mathbf{R}} \end{bmatrix} = \begin{bmatrix} h'_1 & h'_2 & h'_3 & \cdots & h'_N \\ \tilde{r}_{11} & \tilde{r}_{12} & \tilde{r}_{13} & \cdots & \tilde{r}_{1N} \\ & \tilde{r}_{22} & \tilde{r}_{23} & \cdots & \tilde{r}_{2N} \\ & & \tilde{r}_{33} & \cdots & \tilde{r}_{3N} \\ & & & \ddots & \vdots \\ & & & & \tilde{r}_{NN} \end{bmatrix}, \tag{4.4}$$

$\mathbf{J}_1\begin{bmatrix} \mathbf{h}_r^T\mathbf{T} \\ \tilde{\mathbf{R}} \end{bmatrix}$ can then be represented as:

$$\begin{bmatrix} k_1^{(1)} & k_2^{(1)}\cos\theta_2^{(1)} & k_3^{(1)}\cos\theta_3^{(1)} & \cdots & k_N^{(1)}\cos\theta_N^{(1)} \\ & k_2^{(1)}\sin\theta_2^{(1)} & k_3^{(1)}\sin\theta_3^{(1)} & \cdots & k_N^{(1)}\sin\theta_N^{(1)} \\ & \tilde{r}_{22} & \tilde{r}_{23} & \cdots & \tilde{r}_{2N} \\ & & \tilde{r}_{33} & \cdots & \tilde{r}_{3N} \\ & & & \ddots & \vdots \\ & & & & \tilde{r}_{NN} \end{bmatrix}, \tag{4.5}$$

where $k_i^{(1)} = \sqrt{h_i'^2 + \tilde{r}_{1i}^2}$ for $i \geq 1$ and $\theta_1^{(1)} = 0, \theta_i^{(1)} = \cos^{-1} \frac{\tilde{r}_{11}h_i' + \tilde{r}_{1i}h_1'}{k_1^{(1)}k_i^{(1)}}$ for $i \geq 2$. Similarly, after $\mathbf{J}_2$ is multiplied to Eqn. (4.5), the following equation can be obtained:

$$\begin{bmatrix} k_1^{(1)} & k_2^{(1)}\cos\theta_2^{(1)} & k_3^{(1)}\cos\theta_3^{(1)} & \cdots & k_N^{(1)}\cos\theta_N^{(1)} \\ & k_2^{(2)} & k_3^{(2)}\cos\theta_3^{(2)} & \cdots & k_N^{(2)}\cos\theta_N^{(2)} \\ & & k_3^{(2)}\sin\theta_3^{(2)} & \cdots & k_N^{(2)}\sin\theta_N^{(2)} \\ & & \tilde{r}_{33} & \cdots & \tilde{r}_{3N} \\ & & & \ddots & \vdots \\ & & & & \tilde{r}_{NN} \end{bmatrix}, \tag{4.6}$$

where $k_i^{(2)} = \sqrt{(k_i^{(1)}\sin\theta_i^{(1)})^2 + \tilde{r}_{2i}^2}$ for $i \geq 2$ and $\theta_2^{(2)} = 0, \theta_i^{(2)} = \cos^{-1} \frac{\tilde{r}_{22}k_i^{(1)}\sin\theta_i^{(1)} + \tilde{r}_{2i}k_2^{(1)}\sin\theta_2^{(1)}}{k_2^{(2)}k_i^{(2)}}$ for $i \geq 3$. Continuing the process, the parameters for the entries of $\tilde{\mathbf{R}}_u^{in}$ can be generalized as:

$$\begin{aligned} k_i^{(j)} &= \sqrt{(k_i^{(j-1)}\sin\theta_i^{(j-1)})^2 + \tilde{r}_{ji}^2} \quad \text{for } 1 \leq j \leq i \leq N, \\ \theta_i^{(j)} &= \cos^{-1} \frac{\tilde{r}_{jj}k_i^{(j-1)}\sin\theta_i^{(j-1)} + \tilde{r}_{ji}k_j^{(j-1)}\sin\theta_j^{(j-1)}}{k_j^{(j)}k_i^{(j)}} \quad \text{for } 1 \leq j < i \leq N, \\ \theta_i^{(i)} &= 0 \quad \text{for } 1 \leq i \leq N, \end{aligned} \tag{4.7}$$

with $(k_i^{(0)})^2 = h_i'^2$ and $\theta_i^{(0)} = 90°$. Therefore, the squares of the diagonals of $\tilde{\mathbf{R}}_u^{in}$ are represented as:

$$\begin{aligned} (\tilde{r}_{ii}^{in})^2 = (k_i^{(i)})^2 &= (k_i^{(i-1)})^2 \sin^2\theta_i^{(i-1)} + \tilde{r}_{ii}^2, \\ &= (k_i^{(i-2)})^2 \sin^2\theta_i^{(i-2)} \sin^2\theta_i^{(i-1)} + \tilde{r}_{i-1i}^2 \sin^2\theta_i^{(i-1)} + \tilde{r}_{ii}^2, \\ &\vdots \\ &= \tilde{r}_{ii}^2 + \sum_{j=1}^{i-1} \tilde{r}_{ji}^2 \prod_{k=1}^{i-j} \sin^2\theta_i^{(k)} + h_i'^2 \prod_{j=1}^{i} \sin^2\theta_i^{(i-j)} \end{aligned} \tag{4.8}$$

To get the upper bound of the number of required two-reduction steps, the value D defined in Lemma 1 is computed for $\tilde{\mathbf{R}}_u^{in}$ as:

$$\begin{aligned} D_n &= (k_1^{(1)})^{2N}(k_2^{(2)})^{2(N-2)}\cdots(k_N^{(N)})^2 \\ &= (\tilde{r}_{11}^2 + h_1'^2)^N(\tilde{r}_{22}^2 + \tilde{r}_{12}^2\sin^2\theta_2^{(1)} + h_2'^2\sin^2\theta_2^{(1)}\sin^2\theta_2^{(0)})^{N-2}\cdots \\ &\quad (\tilde{r}_{NN}^2 + \sum_{j=1}^{N-1}\tilde{r}_{jN}^2\prod_{k=1}^{N-j}\sin^2\theta_N^{(k)} + h_N'^2\prod_{j=1}^{N}\sin^2\theta_N^{(N-j)}) \\ &= D_p\prod_{i=1}^{N}\left(1 + \sum_{j=1}^{i-1}\frac{\tilde{r}_{ji}^2}{\tilde{r}_{ii}^2}\prod_{k=1}^{i-j}\sin^2\theta_i^{(k)} + \frac{h_i'^2}{\tilde{r}_{ii}^2}\prod_{j=1}^{i}\sin^2\theta_i^{(i-j)}\right)^{N-i+1}, \end{aligned} \tag{4.9}$$

where $D_p = \tilde{r}_{11}^{2N}\tilde{r}_{22}^{2(N-1)}\cdots\tilde{r}_{NN}^2$ is the value for $\mathbf{HT}$, the reduced lattice bases before adding a new row. Therefore, the number of two-reduction steps required in the updating method is at most $O(\log D_u)$, where

$$D_u = \prod_{i=1}^{N}\left(1 + \sum_{j=1}^{i-1}\frac{\tilde{r}_{ji}^2}{\tilde{r}_{ii}^2}\prod_{k=1}^{i-j}\sin^2\theta_i^{(k)} + \frac{h_i'^2}{\tilde{r}_{ii}^2}\prod_{j=1}^{i}\sin^2\theta_i^{(i-j)}\right)^{N-i+1}. \tag{4.10}$$

Here, since $\tilde{r}_{ij}$ is from the QR decomposition of the reduced lattice basis matrix in Eqn. (3.2), satisfying two LR conditions of Eqns. (2.3) and (2.4), it can be easily induced that $\tilde{r}_{ji}^2 \le \tilde{r}_{ii}^2$ for $j < i$, which leads to the following equation:

$$D_u \le \prod_{i=1}^{N}\left(i + \frac{h_i'^2}{\tilde{r}_{ii}^2}\prod_{j=1}^{i}\sin^2\theta_i^{(i-j)}\right)^{N-i+1}. \tag{4.11}$$

Therefore, the complexity of the updating method depends on the matrix size (the first term, i, in Eqn. (4.11)) and the newly updated column (the second term, $\frac{h_i'^2}{\tilde{r}_{ii}^2}\prod_{j=1}^{i}\sin^2\theta_i^{(i-j)}$, in Eqn. (4.11)).

4.2. **Row-wise Downdating.**

In the previous section, the preconditioning matrix $\mathbf{T}$ for the given lattice basis matrix become a good initial parameter to find the new preconditioning matrix when a new row is updated. Similarly, it could also be a good initial parameter when the existing row is removed. In this section, given $\mathbf{H}$ in Eqn. (3.1) with preconditioning matrix $\mathbf{T}$, we find the new preconditioning matrix $\mathbf{T}_d$ after the first row $\mathbf{h}^{(1)T}$ is removed:

$$\mathbf{G}_d = \mathbf{H}_d\mathbf{T}_d, \quad \mathbf{H} = \begin{bmatrix} \mathbf{h}^{(1)T} \\ \mathbf{H}_d \end{bmatrix}, \quad \mathbf{H}_d \in \mathbb{R}^{M-1\times N} \tag{4.12}$$

where $\mathbf{G}_d$ is lattice reduced.

Since the LLL algorithm is independent on the orthogonal part of QR decomposition, we first start from the following equation as:

$$\mathbf{H}_d^T\mathbf{H}_d = \mathbf{H}^T\mathbf{H} - \mathbf{h}^{(1)}\mathbf{h}^{(1)T} = \begin{bmatrix}\mathbf{H}^T & \mathbf{h}^{(1)}\end{bmatrix}\begin{bmatrix}\mathbf{I}_N & \mathbf{0}\\ \mathbf{0} & -1\end{bmatrix}\begin{bmatrix}\mathbf{H}\\ \mathbf{h}^{(1)T}\end{bmatrix}. \tag{4.13}$$

Since $\mathbf{H}^T\mathbf{H} = \mathbf{T}^{-T}\tilde{\mathbf{R}}^T\tilde{\mathbf{R}}\mathbf{T}^{-1}$, Eqn. (4.13) can be written as:

$$\begin{aligned}\mathbf{H}_d^T\mathbf{H}_d &= \mathbf{T}^{-T}\tilde{\mathbf{R}}^T\tilde{\mathbf{R}}\mathbf{T}^{-1} - \mathbf{h}^{(1)}\mathbf{h}^{(1)T}\\ &= \begin{bmatrix}\mathbf{T}^{-T}\tilde{\mathbf{R}}^T & \mathbf{h}^{(1)}\end{bmatrix}\begin{bmatrix}\mathbf{I}_N & \mathbf{0}\\ \mathbf{0} & -1\end{bmatrix}\begin{bmatrix}\tilde{\mathbf{R}}\mathbf{T}^{-1}\\ \mathbf{h}^{(1)T}\end{bmatrix}\\ &= \mathbf{T}^{-T}\begin{bmatrix}\tilde{\mathbf{R}}^T & \mathbf{T}^T\mathbf{h}^{(1)}\end{bmatrix}\begin{bmatrix}\mathbf{I}_N & \mathbf{0}\\ \mathbf{0} & -1\end{bmatrix}\begin{bmatrix}\tilde{\mathbf{R}}\\ \mathbf{h}^{(1)T}\mathbf{T}\end{bmatrix}\mathbf{T}^{-1}.\end{aligned} \tag{4.14}$$

Therefore,

$$\mathbf{T}^T\mathbf{H}_d^T\mathbf{H}_d\mathbf{T} = \begin{bmatrix}\tilde{\mathbf{R}}^T & \mathbf{T}^T\mathbf{h}^{(1)}\end{bmatrix}\begin{bmatrix}\mathbf{I}_N & \mathbf{0}\\ \mathbf{0} & -1\end{bmatrix}\begin{bmatrix}\tilde{\mathbf{R}}\\ \mathbf{h}^{(1)T}\mathbf{T}\end{bmatrix}. \tag{4.15}$$

Like the Cholesky downdating, $\begin{bmatrix}\tilde{\mathbf{R}}\\ \mathbf{h}^{(1)T}\mathbf{T}\end{bmatrix}$ can be then be recovered into the triangular form, $\tilde{\mathbf{R}}_d^{in}$, by applying a total of N hyperbolic rotations [19]. Accordingly, the new preconditioning matrix $\mathbf{T}_d$ can be found with initial parameters $\tilde{\mathbf{R}}_d^{in}$ and $\mathbf{T}$.

5. Rounding error analysis

In this section we look into the effect of the rounding errors during the updating/downdating methods on the process of the LLL algorithm, beginning with the column-wise updating in Section 3.1. It is assumed that the computed $\hat{\mathbf{R}}$ of $\mathbf{R}$ in Eqn. (3.1) has the following form as:

$$\hat{\mathbf{R}} = \mathbf{R} + \delta\mathbf{R}, \quad \|\delta\mathbf{r}_i\|_2 \le \zeta\|\mathbf{r}_i\|_2, \tag{5.1}$$

where $\delta\mathbf{R}$ is the computational error of $\mathbf{R}$ and $\delta\mathbf{r}_i$ is the ith column of $\delta\mathbf{R}$. From the results in [13, 14], the computed $\hat{\mathbf{w}}$ and $\hat{\mathbf{q}}$ in Eqn. (3.5) have the following forms as:

$$\hat{\mathbf{w}} = \mathbf{w} + \delta\mathbf{w}, \quad \|\delta\mathbf{w}\|_2 \le M\epsilon_p\|\mathbf{w}\|_2, \tag{5.2}$$

$$\hat{\mathbf{q}} = \mathbf{q} + \delta\mathbf{q}, \quad \|\delta\mathbf{q}\|_2 \le (M+1)\epsilon_p\|\mathbf{q}\|_2 \tag{5.3}$$

where $\delta\mathbf{w}$ and $\delta\mathbf{q}$ are the computational errors and ϵ_p is the unit *roundoff*. The computed initial parameter $\hat{\tilde{\mathbf{R}}}_u^{in}$ for the LLL algorithm can then be computed as follows:

$$\hat{\tilde{\mathbf{R}}}_u^{in} = \tilde{\mathbf{R}}_u^{in} + \delta\tilde{\mathbf{R}}_u^{in}, \quad \|\delta\tilde{\mathbf{r}}_i^{in}\|_2 \le \zeta'\|\tilde{\mathbf{r}}_i^{in}\|_2, \tag{5.4}$$

where $\delta\tilde{\mathbf{r}}_i^{in}$ is the ith column of $\delta\tilde{\mathbf{R}}_u^{in}$ and $\zeta' = \max(\zeta, (M+2)\epsilon_p)$.

Since the computational complexity and the rounding errors are mainly dependent on two-reduction steps, we would start from analyzing two-reduction step in Table 1 with the initial parameter (5.4). As analyzed in [20, 14], there are exact plane rotations $\hat{\mathbf{U}}_1$, ..., $\hat{\mathbf{U}}_P$ such that for any vector $\mathbf{v}$

$$fl(\mathbf{U}_P...\mathbf{U}_1\mathbf{v}) = \hat{\mathbf{U}}_P...\hat{\mathbf{U}}_1\mathbf{v} + \mathbf{g}, \tag{5.5}$$

where $\|\mathbf{g}\|_2 \lesssim 6P\|\mathbf{v}\|_2\epsilon_p$. Therefore, the error bound after one two-reduction step can be given as:

$$\hat{\tilde{\mathbf{R}}}_u^{(1)} = \tilde{\mathbf{R}}_u^{(1)} + \delta\tilde{\mathbf{R}}_u^{(1)}, \quad \|\delta\tilde{\mathbf{r}}_i^{(1)}\|_2 \leq 6(\|\tilde{\mathbf{r}}_i^{(1)}\|_2 + \|\delta\tilde{\mathbf{r}}_i^{(1)}\|_2)\epsilon_p \leq 6(1+\zeta')\epsilon_p\|\tilde{\mathbf{r}}_i^{(1)}\|_2. \tag{5.6}$$

In size-reduction step that follows, the computed $\hat{\mu}$ is given as

$$\hat{\mu} = \left\lceil \frac{\hat{r}_{ij}^{(1)}}{\hat{r}_{ii}^{(1)}} \right\rfloor = \left\lceil \frac{r_{ij}^{(1)}}{r_{ii}^{(1)}}(1 + \frac{e_{ij}^{(1)} - e_{ii}^{(1)}}{1 + e_{ii}^{(1)}}) \right\rfloor, \tag{5.7}$$

where $e_{ij}^{(1)} = \frac{\delta r_{ij}^{(1)}}{r_{ij}^{(1)}}$. Let $\frac{r_{ij}^{(1)}}{r_{ii}^{(1)}} = m + \alpha$, where m is an integer and α is a real number within $[-0.5, 0.5)$; Eqn. (5.7) can then be written as:

$$\hat{\mu} = \left\lceil m + \alpha + \left(\frac{e_{ij}^{(1)} - e_{ii}^{(1)}}{1 + e_{ii}^{(1)}} \right)(m+\alpha)) \right\rfloor. \tag{5.8}$$

Assuming that

$$\left| \alpha + \left(\frac{e_{ij}^{(1)} - e_{ii}^{(1)}}{1 + e_{ii}^{(1)}} \right)(m+\alpha) \right| < 0.5, \tag{5.9}$$

then $\hat{\mu} = m = \mu$ which implies that $\mathbf{T}$ has no computational error and the modular operation can eliminate the effect of the previous rounding errors on $\mathbf{T}$ under the condition of Eqn. (5.9). Note that we would not consider $\left| \alpha + \left(\frac{e_{ij}^{(1)} - e_{ii}^{(1)}}{1 + e_{ii}^{(1)}} \right)(m+\alpha) \right| \geq 0.5$ in this paper and leave the analysis of its effect and the method to remove it as future works.

After size-reduction step,

$$\begin{aligned} \hat{r}_{ij}^{(2)} = \hat{r}_{ij}^{(1)} - m\hat{r}_{ii}^{(1)} &= r_{ij}^{(1)} + \delta r_{ij}^{(1)} - m(r_{ii}^{(1)} + \delta r_{ii}^{(1)}), \\ &= \alpha r_{ii}^{(1)} + \delta r_{ij}^{(1)} - m\delta r_{ii}^{(1)}, \\ &= r_{ij}^{(2)} + \delta r_{ij}^{(2)}, \end{aligned} \tag{5.10}$$

where $r_{ij}^{(2)} = \alpha r_{ii}^{(1)}$ and

$$\begin{aligned} |\delta r_{ij}^{(2)}| &= |\delta r_{ij}^{(1)} - m\delta r_{ii}^{(1)}|, \\ &= |\delta r_{ij}^{(1)} - \frac{r_{ij}^{(1)}}{r_{ii}^{(1)}}\delta r_{ii}^{(1)} + \alpha\delta r_{ii}^{(1)}|, \\ &\leq 12(1+\zeta')\epsilon_p|r_{ij}^{(1)}| + 3(1+\zeta')\epsilon_p|r_{ii}^{(1)}|, \\ &\leq 15(1+\zeta')\epsilon_p \max(|r_{ij}^{(1)}|, |r_{ii}^{(1)}|). \end{aligned} \tag{5.11}$$

Therefore, if N_s^u two-reduction steps are required to achieve $\mathbf{T}_u$, the error is given as:

$$\|\delta\mathbf{r}_u\|_2 \leq 15N_s^u(1+\zeta')\epsilon_p B', \tag{5.12}$$

where $B' = \max\{\|\mathbf{h}_1\|_2, ..., \|\mathbf{h}_N\|_2\}$. Therefore, it shows that the column-wise LLL updating process is stable. As pointed in [15], the column-wise updating/downdating and row-wise updating of orthogonal decompositions are numerically stable. Similarly the process to compute the initial parameters for column-wise LLL downdating and row-wise LLL updating is also stable and the above analysis is still valid. For the row-wise LLL downdating, the hyperbolic rotations are used to compute the initial upper triangular matrix. Based on [16, 17], the hyperbolic rotation is also proved to be mixed forward-backward stable. Therefore, the above analysis can also be easily extended to see that the row-wise LLL downdating is stable.

6. Numerical results

In this section, we run the Monte-Carlo simulations to evaluate the performance of the proposed column-wise updating/downdating and the row-wise updating/downdating methods. In the simulations, each element of the lattice basis matrix $\mathbf{H}$ is drawn from the uniform distribution on the unit interval $[0,1]$. Newly updated column vector $\mathbf{h}_c$ and row vector $\mathbf{h}_r^T$ also follow the uniform distribution on the unit interval. All the simulations are run by using MATLAB on Windows XP with 2.13 GHz CPU and 2GB memory.

6.0.1. *Column-wise LLL updating and downdating.* Table 3 lists the average of $\kappa(\mathbf{H}_a)$, $\kappa(\mathbf{H}_a\mathbf{T}_1)$, and $\kappa(\mathbf{H}_a\mathbf{T}_2)$, for various column sizes when a new column is appended, where $\mathbf{H}_a = [\mathbf{H}\ \mathbf{h}_c]$, $\mathbf{H} \in \mathbb{R}^{(N+2)\times N}$, $\mathbf{h}_c \in \mathbb{R}^{(N+2)\times 1}$ and $\kappa(\mathbf{A})$ denotes the condition number of a matrix $\mathbf{A}$. The preconditioning matrices $\mathbf{T}_1$ and $\mathbf{T}_2$ are, respectively, obtained by using the updating method and by using the conventional LLL. In Table 3, the average number of two-reduction steps performed when the LLL algorithm is applied to $\mathbf{H}_a$ (denoted as N_t) and the average number of two-reduction steps performed when the proposed updating method is applied, given the reduced lattice bases of $\mathbf{H}$ (denoted as N_u) are also compared. For reference, the average number of two-reduction steps performed when the LLL algorithm is applied to $\mathbf{H}$ (denoted as N_o) is also evaluated. Finally, t_u and t_t are the average *computing time*s per one sample in milliseconds to get the preconditioning matrix with and without the proposed updating algorithm, respectively.

TABLE 3. Comparison of the average condition numbers and the average number of two-reduction steps for the column-wise updating method.

Size (N)	$\kappa(\mathbf{H}_a)$	$\kappa(\mathbf{H}_a\mathbf{T}_1)$	$\kappa(\mathbf{H}_a\mathbf{T}_2)$	N_u	N_t	$t_u(ms)$	$t_t(ms)$
4	24.35	2.65	2.65	3.30	7.71	0.17	0.35
6	40.26	3.25	3.25	5.03	15.30	0.25	0.67
8	57.37	4.14	4.14	6.84	24.28	0.35	1.08
10	78.28	5.40	5.40	8.16	33.30	0.44	1.53
12	100.20	7.05	7.05	9.14	41.67	0.52	2.00
14	123.02	9.12	9.12	10.03	49.28	0.60	2.48
16	149.60	11.69	11.69	10.49	55.46	0.68	2.96
18	171.69	14.67	14.67	10.30	60.48	0.73	3.40
20	209.06	18.11	18.11	10.47	65.23	0.81	3.88
22	234.12	21.77	21.77	10.53	69.34	0.89	4.36
$\vdots$		$\vdots$			$\vdots$		$\vdots$
50	806.54	111.40	111.40	10.51	121.31	2.04	12.42
60	1099.18	159.42	159.42	10.40	136.28	2.50	15.70
70	1286.72	214.41	214.41	10.54	151.00	3.02	19.17
80	1550.84	275.22	275.22	10.37	163.56	3.53	22.61
90	1912.26	340.88	340.88	10.76	177.44	4.17	26.53
500	22775	6343.7	6343.7	9.86	616.30	55.29	316.66
1000	68148	19950	19950	10.59	1121.1	223.13	1259.78

From the results of the condition numbers, LLL algorithms with or without the updating algorithm make the general lattice basis matrices better-conditioned and their outputs exhibit the same condition numbers. In comparison of the average numbers of the two-reduction steps, LLL algorithm with the updating method performs much less two-reduction steps, that is, the updating method removes the redundant computational complexities.

Remark 2. *In the comparison of t_u and t_t, LLL with the updating algorithm takes roughly $2 \sim 6$ times less computing time than that without updating algorithm and its ratio goes higher as the matrix size (N) becomes larger in a certain point. Interestingly, the ratio of t_u and t_t tends to be saturated as N increases, while the ratio of N_u and N_t does not. This is because the numbers of while loop iterations in both cases are, respectively, given by $N + N_u$ and $N + N_t$, which prevents the computing time ratio from going infinity. In addition, N_u is saturated (around 10) as N increases, because the LLL reduction conditions (Eqns. (3) and (4)) are checked with $\mathbf{T}_u^{in}$ and $\tilde{\mathbf{R}}_u^{in}$ of Eqn. (12) in the proposed updating algorithm.*

Table 4 shows the data for various column size when an existing column is removed. The matrix $\mathbf{H}_b$ denotes the submatrix of $\mathbf{H}$ $(= [\mathbf{H}_b\ \mathbf{h}_N] \in \mathbb{R}^{(N+1)\times N})$. The condition numbers, $\kappa(\mathbf{H}_b)$, $\kappa(\mathbf{H}_b\mathbf{T}_1)$, and $\kappa(\mathbf{H}_b\mathbf{T}_2)$, are evaluated where $\mathbf{T}_1$ and $\mathbf{T}_2$ are, respectively, obtained by the downdating method and the conventional LLL. The average numbers of two-reduction

TABLE 4. Comparison of the average condition numbers and the average number of two-reduction steps for the column-wise downdating method.

Size (N)	$\kappa(\mathbf{H}_b)$	$\kappa(\mathbf{H}_b\mathbf{T}_1)$	$\kappa(\mathbf{H}_b\mathbf{T}_2)$	N_u	N_t	$t_u(ms)$	$t_t(ms)$
4	7.31	2.13	2.08	0.93	2.28	0.07	0.12
6	14.76	2.65	2.61	2.10	7.07	0.14	0.32
8	23.98	3.28	3.26	3.22	13.69	0.22	0.61
10	33.50	4.16	4.18	4.21	21.19	0.31	0.97
12	45.49	5.35	5.39	4.89	28.99	0.39	1.35
14	59.29	6.91	6.99	5.41	36.07	0.48	1.75
16	71.79	8.83	8.94	5.94	42.06	0.56	2.16
18	85.31	11.11	11.25	6.05	47.56	0.64	2.56
20	101.70	13.79	13.91	5.97	52.51	0.71	2.97
22	117.44	16.76	16.85	5.94	56.69	0.80	3.40
$\vdots$		$\vdots$			$\vdots$		$\vdots$
50	417.85	89.70	89.83	5.83	109.18	2.09	10.47
60	547.65	129.61	129.81	5.87	124.31	2.61	13.38
70	685.25	174.98	175.34	5.79	138.95	3.16	16.46
80	841.10	225.33	225.93	5.80	151.92	3.76	19.61
90	1007.25	280.32	281.06	5.74	165.57	4.35	22.97
500	13633	5315.7	5370.8	5.90	605.89	61.61	291.26
1000	37715	16432	16356	6.13	1108.7	223.88	1013.15

steps with/without the downdating method, respectively denoted as N_u and N_t, are also compared. From the results of the condition numbers, $\mathbf{H}_b\mathbf{T}_1$ and $\mathbf{H}_b\mathbf{T}_2$ have similar condition numbers, lower than that for $\mathbf{H}_b$. In comparison of the average numbers of the two-reduction steps, the LLL algorithm with the downdating method performs much less two-reduction steps. Accordingly, its time consumption (t_u) is also less than that (t_t) for the LLL algorithm without the downdating method. Note that the average computing time of the conventional LLL algorithm (t_t) in Table 3 is slightly higher than that in Table 4 because the matrix size of $\mathbf{H}_a$ is larger than that of $\mathbf{H}_b$. In addition, as N becomes larger, the significant gap can be observed because the numbers of *while* loop iterations dominate in the computing time (see also Remark 2).

6.0.2. *Row-wise LLL updating and downdating.* Table 5 compares the condition numbers for the outputs with/without the row-wise updating method for $\mathbf{H}_c$, where $\mathbf{H}_c = [\mathbf{h}_r^T;\ \mathbf{H}]$, $\mathbf{H} \in \mathbb{R}^{(N+1)\times N}$, and $\mathbf{h}_r^T \in \mathbb{R}^{1\times N}$. Here, $\mathbf{T}_1$ and $\mathbf{T}_2$ are, respectively, obtained by the updating method and the conventional LLL. The numbers of two-reduction steps are also compared. Similarly to the case of the column-wise updating, the row-wise updating method can also reduce the computational complexities given the preconditioning matrix for the lattice basis matrix before a new row is updated.

TABLE 5. Comparison of the average condition numbers and the average number of two-reduction steps for the row-wise updating method.

Size (N)	$\kappa(\mathbf{H}_c)$	$\kappa(\mathbf{H}_c\mathbf{T}_1)$	$\kappa(\mathbf{H}_c\mathbf{T}_2)$	N_u	N_t	$t_u(ms)$	$t_t(ms)$
4	10.85	2.34	2.35	0.76	4.40	0.07	0.21
6	19.17	2.88	2.89	1.65	10.24	0.13	0.46
8	29.01	3.63	3.68	2.68	17.40	0.20	0.78
10	39.50	4.65	4.75	3.81	25.07	0.28	1.15
12	51.68	5.96	6.14	4.91	32.71	0.38	1.56
14	64.76	7.64	7.91	6.02	39.25	0.48	1.96
16	78.75	9.69	10.06	7.05	44.77	0.59	2.35
18	93.17	12.09	12.49	7.72	50.20	0.70	2.78
20	110.18	14.81	15.35	8.61	54.77	0.83	3.18
22	126.33	17.96	18.47	9.19	59.02	0.95	3.60
$\vdots$		$\vdots$			$\vdots$		$\vdots$
50	432.43	94.17	93.62	13.61	110.70	2.86	10.71
60	558.73	136.06	134.19	13.86	125.73	3.58	13.61
70	707.48	182.51	180.48	14.61	140.03	4.38	16.70
80	870.18	233.50	231.08	15.11	153.54	5.24	19.91
90	1027.72	291.84	288.09	15.66	166.44	6.15	23.24
500	13324	5400.4	5350.3	19.52	603.41	86.85	289.76
1000	38281	16890	16455	19.38	1108.9	346.82	1102.32

In Table 6, the condition numbers and the numbers of two-reduction steps are compared for the outputs with and without the row-wise downdating method for the submatrix $\mathbf{H}_d$ of $\mathbf{H}$, where $\mathbf{H} = [\mathbf{h}^{(1)T}; \mathbf{H}_d]$, $\mathbf{H} \in \mathbb{R}^{(N+3)\times N}$, and $\mathbf{h}^{(1)T} \in \mathbb{R}^{1\times N}$. Similarly, $\mathbf{T}_1$ and $\mathbf{T}_2$ are, respectively, obtained by the downdating method and the conventional LLL. It can be found that the proposed row-wise downdating method also requires much less two-reduction steps and simultaneously exhibits a similar condition numbers of LLL reduced matrices to that using the LLL algorithm without the row-wise downdating method.

7. CONCLUDING REMARKS

In this paper, we propose the LLL updating and downdating methods when a new basis column (or row) is added or when an existing basis column (or row) is removed in a given lattice basis matrix. Through the proposed updating and downdating methods, the preconditioning matrix for the original lattice basis matrix is modified to provide a suitable initial parameter holding the information of the original reduced bases, eliminating the redundant complexities in finding newly updated preconditioning matrix. Moreover, through the rounding error analysis, the LLL updating/downdating methods are shown stable. The simulation results reveal that the proposed updating and downdating methods reduce the computational complexities giving

TABLE 6. Comparison of the average condition numbers and the average number of two-reduction steps for the row-wise downdating method.

Size (N)	$\kappa(\mathbf{H}_d)$	$\kappa(\mathbf{H}_d\mathbf{T}_1)$	$\kappa(\mathbf{H}_d\mathbf{T}_2)$	N_u	N_t	$t_u(ms)$	$t_t(ms)$
4	10.94	2.33	2.34	0.65	4.40	0.07	0.21
6	18.78	2.88	2.89	1.61	10.16	0.12	0.45
8	29.22	3.64	3.68	3.03	17.50	0.20	0.78
10	40.02	4.68	4.73	4.55	25.28	0.29	1.16
12	52.14	6.07	6.15	6.31	32.48	0.39	1.54
14	64.52	7.78	7.92	7.87	39.36	0.50	1.96
16	78.94	9.85	10.04	8.90	44.69	0.61	2.34
18	92.90	12.30	12.52	9.88	50.06	0.71	2.76
20	109.73	15.05	15.31	10.55	54.48	0.82	3.16
22	127.97	18.25	18.50	11.14	59.00	0.93	3.60
⋮		⋮			⋮		⋮
50	432.02	93.40	93.88	16.08	110.55	2.62	10.72
60	555.71	133.91	133.67	16.80	125.25	3.29	13.58
70	700.48	180.54	179.74	17.47	140.07	4.00	16.70
80	858.56	231.99	231.59	18.03	153.77	4.79	19.92
90	1026.08	286.81	285.71	18.72	167.17	5.60	23.27
500	7508.1	3994.2	3959.2	19.09	592.89	64.53	255.44
1000	20403	12280	12237	18.99	1096.8	239.46	907.57

the preconditioning matrix which transforms the lattice basis matrix into a better-conditioned matrix.

ACKNOWLEDGMENT

The authors would like to thank the anonymous reviewers whose comments helped us improve this paper.

REFERENCES

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

[2] P. Nguyen and J. Stern, "Lattice reduction in cryptology: An update," in *Algorithmic number theory: 4th international symposium - ANTS-IV, Lecture Notes in Computer Sciences*, vol. 1838. Springer, 2000, pp. 85–112.

[3] C. P. Schnorr, "A hierarchy of polynomial time lattice basis reduction algorithms," *Theoretical Computer Science*, vol. 53, pp. 201–224, 1987.

[4] K. Lee, J. Chun, and L. Hanzo, "Optimal lattice-reduction aided successive interference cancellation for mimo systems," *IEEE transaction on Wireless Communications*, vol. 6, no. 7, pp. 2438–2443, July 2007.

[5] C. Windpassinger and R. F. H. Fischer, “Low-complexity near-maximum-likelihood detection and precoding for MIMO systems using lattice reduction,” in *Proc. IEEE Information Theory Workshop (ITW)*, Mar. 2003, pp. 345–348.

[6] H. Yao and G. W. Wornell, “Lattice-reduction-aided detectors for MIMO communication systems,” in *Proc. IEEE Global Telecommunication Conference, Taipei, Taiwan*, vol. 1, Nov. 2002, pp. 424–428.

[7] D. Wübben, R. Böhnke, V. Kühn, and K. D. Kammeyer, “Near-maximum-likelihood detection of MIMO systems using MMSE-based lattice-reduction,” in *Proc. IEEE International Conference on Communications*, vol. 2, June 2004, pp. 798–802.

[8] J. Park and J. Chun, “Improved lattice reduction-aided MIMO successive interference cancellation under channel estimation errors,” *IEEE Trans. Signal Processing*, vol. 60, no. 6, pp. 3346–3351, June 2012.

[9] ——, “Efficient lattice-reduction-aided successive interference cancellation for clustered multiuser MIMO system,” *IEEE transactions on Vehicular Technology*, vol. 61, no. 8, pp. 3643–3655, Oct. 2012.

[10] A. Storjohann, “Faster algorithms for integer lattice basis reduction,” Swiss Federal Institute of Technology, Departement Informatik, ETH Zurich, Tech. Rep. TR 249, July 1996.

[11] H. Koy and C. P. Schnorr, “Segment LLL-reduction of lattice bases,” in *in: Cryptograhpy and Lattices, Lecture Notes in Computer Sciences*, vol. 2146. New York: Springer, 2001, pp. 67–80.

[12] ——, “Segment LLL-reduction with floating point orthogonalization,” in *in: Cryptograhpy and Lattices, Lecture Notes in Computer Sciences*, vol. 2146. Springer, 2001, pp. 81–96.

[13] N. J. Higham, *Accuracy and Stability of Numerical Algorithms*, 2nd ed. Philadelphia: SIAM, 2002.

[14] J. H. Wilkinson, *The Algebraic Eigenvalue Problem*. Clarendon Press, Oxford, 1965.

[15] C. C. Paige, “Error analysis of some techniques for updating orthogonal decompositions,” *Mathematics of Computation*, vol. 34, no. 150, pp. 465–471, 1980.

[16] G. Hargreaves, “Topics in matrix computations: Stability and efficiency of algorithms,” Ph.D. dissertation, Univ. of Manchester, Manchester, UK, 2005.

[17] A. Bojanczyk, N. J. Higham, and H. Patel, “Solving the indefinite least squares problem by hyperbolic QR factorization,” *SIAM Journal on Matrix Analysis and Applications*, vol. 24, no. 4, pp. 914–931, 2003.

[18] M. C. Cary, “Lattice basis reduction: Algorithms and applications,” *Unpublished draft available at http://www.cs.washington.edu/homes/cary/lattice.pdf*, Feb. 2002.

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

[20] G. W. Stewart, “The effects of rounding error on an algorithm for downdating a cholesky factorization,” *Journal of the Institute of Mathematics and its Applications*, vol. 23, no. 2, pp. 203–213, 1979.