EURASIP Journal on Advances in Signal Processing
a SpringerOpen Journal

RESEARCH **Open Access**

Experimental quality evaluation of lattice basis reduction methods for decorrelating low-dimensional integer least squares problems

Peiliang Xu

Abstract

Reduction can be important to aid quickly attaining the integer least squares (ILS) estimate from noisy data. We present an improved LLL algorithm with fixed complexity by extending a parallel reduction method for positive definite quadratic forms to lattice vectors. We propose the minimum angle of a reduced basis as an alternative quality measure of orthogonality, which is intuitively more appealing to measure the extent of orthogonality of a reduced basis. Although the LLL algorithm and its variants have been widely used in practice, experimental simulations were only carried out recently and limited to the quality measures of the Hermite factor, practical running behaviors and reduced Gram-Schmidt coefficients. We conduct a large scale of experiments to comprehensively evaluate and compare five reduction methods for decorrelating ILS problems, including the LLL algorithm, its variant with deep insertions and our improved LLL algorithm with fixed complexity, based on six quality measures of reduction. We use the results of the experiments to investigate the mean running behaviors of the LLL algorithm and its variants with deep insertions and the sorted QR ordering, respectively. The improved LLL algorithm with fixed complexity is shown to perform as well as the LLL algorithm with deep insertions with respect to the quality measures on length reduction but significantly better than this LLL variant with respect to the other quality measures. In particular, our algorithm is of fixed complexity, but the LLL algorithm with deep insertions could seemingly not be terminated in polynomial time of the dimension of an ILS problem. It is shown to perform much better than the other three reduction methods with respect to all the six quality measures. More than six millions of the reduced Gram-Schmidt coefficients from each of the five reduction methods clearly show that they are not uniformly distributed but depend on the reduction algorithms used. The simulation results of the reduced Gram-Schmidt coefficients have clearly shown that our improved LLL algorithm tends to produce small reduced Gram-Schmidt coefficients near zero with a larger probability and large reduced Gram-Schmidt coefficients near both ends of 0.5
and −0.5 with a smaller probability.

Keywords: Integer linear model; Integer least squares; Closest point problem; Lattice reduction; LLL algorithm

1 Introduction

Reduction is to find the shortest basis vectors and try to make them as orthogonal as possible [1,2]. It has been revolutionarily revitalized with the publication of the landmark polynomial-time reduction method by A. Lenstra, H. Lenstra and L. Lovasz [3]. Since this reduction method was invented by the three authors with an L in all their family names, it has since been widely known as the LLL or L^3 algorithm (see e.g., [4-6]). Almost all practical algorithms of reduction are involved with the LLL algorithm at a certain stage [7]. The LLL algorithm has already had a profound impact on computational geometry of numbers and found many important applications in a variety of highly interdisciplinary subjects such as integer programming [8-10], multiple-input-multiple-output (MIMO) communication systems [11-14], learning with

Correspondence: pxu@rcep.dpri.kyoto-u.ac.jp
Disaster Prevention Research Institute, Kyoto University, Uji, Kyoto 611-0011, Japan

errors [15], cryptography [4,16], discrete tomography [17], and global navigation satellite systems (see e.g., [18-27]). As a result, even an international conference was solely dedicated to celebrate the 25th birthday of the invention of the LLL algorithm at the University of Caen in 2007, with its proceedings containing excellent review and application papers (see e.g., [6,28,29]) published in 2010 [5] (For more information on this event, the reader is referred to the conference website http://lll25.info.unicaen.fr/ and the book of proceedings [5]).

Although the LLL algorithm has been successfully applied in practice, its actual running behavior remains mysterious, is problem-dependent and cannot be precisely predicted in advance (see e.g., [6,14,29,30]), due to the fact that the swapping operation of lattice vectors is controlled by the Lovasz condition with a swapping control parameter δ (see e.g., [3,6,7,30]). Subsequent theoretical works are thus mainly focused on two aspects: (a) to understand statistical mean running behavior and average complexity of the LLL algorithm in practice and (b) to improve the efficiency and stability of the LLL algorithm. Given a lattice $\mathcal{L}$ with a complete basis $\mathbf{B}$ of full rank n, Daudé and Vallée [30] proved that the complexity $O(n^4 \log A)$ of the LLL algorithm, as given originally by Lenstra et al. [3], can be replaced by $O(n^4 \log A/a)$, which depends only on the ratio of lengths between the longest and shortest lattice vectors. Here, A is the length of the longest or maximum vector and a that of the shortest vector. By assuming a probabilistic model of unit ball for random lattice vectors (see also [31]), Daudé and Vallée [30] further obtained the statistical mean complexity of $O(n^4 \log n/2)$ for the LLL algorithm. Recently, Jaldén et al. [13] showed that the complexity of $O(n^4 \log A/a)$ [30] should only depend on the condition number κ_B of the starting lattice basis $\mathbf{B}$. In other words, A/a should be replaced by κ_B. Ling and Howgrave-Graham [32] proposed an effective LLL reduction method by relaxing the size-reduced condition of the original LLL algorithm and analyzed its complexity (see also [33]).

In addition to the probabilistic model approach, one can directly conduct random simulations to gain insight into practical running average behavior of the LLL algorithm. An excellent numerical experiment in this aspect was recently carried out by Nguyen and Stehlé [7], based on three types of random lattice bases, namely, the Goldstein-Mayer bases, the Ajtai-type bases and the knapsack-type bases. Their simulations and theoretical analysis confirmed the well-known fact that the LLL algorithm performs much better in practice than the worst-case bound of complexity. As a result, they proposed a floating-point-based L^2 algorithm [6,7]. Based on the random simulation results, Nguyen and Stehlé [7] further studied the output quality of Hermite defects and the distribution of the reduced Gram-Schmidt coefficients μ_{ij} between -0.5 and 0.5. Following the experiments on μ_{ij} by Nguyen and Stehlé [7], Schneider et al. [34] studied the mean and variance of the shortest reduced vector.

A number of approaches have been proposed in order to improve the performance and output quality of the LLL algorithm, which include (a) imposing stronger test conditions for swapping lattice vectors, (b) improving numerical stability using Householder factorization and floating point techniques, and (c) directly simplifying the LLL algorithm with fixed complexity [14]. Compared with the Lovasz swapping test of $\delta\|\mathbf{b}_i^*\|^2 \leq \|\mathbf{b}_{i+1}^* + \mu_{(i+1)i}\mathbf{b}_i^*\|^2$ with $\delta = 3/4$, a stronger swapping strategy implies using a larger value for the control parameter δ in the Lovasz condition, which can be between 0.95 and 0.999 (see e.g., [7]). Unlike the Lovasz test which involves only the two consecutive orthogonalized vectors $\mathbf{b}_i^*$ and $\mathbf{b}_{i+1}^*$, an even much stronger swapping strategy was proposed by Schnorr and Euchner [35], which is involved with all the orthogonalized vectors $(\mathbf{b}_i^*, \mathbf{b}_{i+1}^*, \ldots, \mathbf{b}_k^*)$ for all $i \leq (k-1)$. If a swapping is required, the vector $\mathbf{b}_k$ is directly *inserted* between $\mathbf{b}_{i-1}$ and $\mathbf{b}_i$. As a result, the strategy is naturally called *deep insertions* by Schnorr and Euchner [35] (see also [6]). Numerical stability and computational efficiency have been successfully attained using Householder factorization or properly selecting a floating point precision (see e.g, [6,7]). Heuristics and sorting were also demonstrated to enable to speed up reduction such as the LLL algorithm and improve its output quality [19,21-23,36-39] (For the review on recent progress of the LLL algorithm and other variants, the reader is referred to Nguyen and Vallée [5], Stehlé [6], Seysen [40] and Vallée and Vera [29]).

The purposes of this paper are threefold: (a) to extend the parallel Cholesky-based reduction for positive definite quadratic forms proposed recently by Xu [22] to the reduction of lattice basis vectors, which will be referred to as an improved LLL algorithm with fixed complexity in the remainder of this paper; (b) to propose the minimum angle as an alternative quality measure of orthogonalization of the reduced lattice basis; and (c) to conduct a large scale of random simulations in order to compare and evaluate five lattice basis reduction methods, namely, the original LLL algorithm, the deep-insertion LLL algorithm proposed by Schnorr and Euchner [35], the fixed complexity algorithm published by Vetter et al. [14], the LLL algorithm with the sorted QR ordering presented by Gan and Mow [38] (see also [33,39]) for basis vectors and by Xu [22] for positive definite quadratic forms, and the improved LLL algorithm with fixed complexity developed in this paper. The comparison and evaluation of these methods will be based on a number of reduction quality measures. In Section 2, we will first briefly outline the LLL algorithm, the deep-insertion LLL algorithm, the fixed complexity algorithm and the LLL algorithm with the sorted QR ordering for the convenience of comparison in numerical simulations, and then present our own improved LLL algorithm with fixed complexity. Section 3 will focus on quality measures of reduction of lattice vectors. In Section 4, we will conduct a large scale of random simulations to demonstrate the performance of the improved LLL algorithm with fixed complexity and compare it with the other four algorithms. Finally, we will summarize the major results in Section 5.

2 LLL-based reduction methods

Given m linearly independent vectors $\mathbf{b}_1, \mathbf{b}_2, \ldots, \mathbf{b}_m$ in $\mathcal{R}^n$, a (sub-)lattice is a discrete point set defined by (see e.g., [1,2,41])

$$\mathcal{L} = \left\{ \sum_{i=1}^{m} \mathbf{b}_i z_i \mid z_i \in \mathbb{Z} \right\}, \tag{1}$$

where $\mathcal{R}^n$ is an n-dimensional, real-valued space, and $\mathbb{Z}$ is a one-dimensional integer space. In particular, Freeden [41] has further developed harmonic lattice point theory for use in geomathematics. The vectors $\mathbf{b}_i$ $(i = 1, 2, \ldots, m)$ form a basis of the lattice $\mathcal{L}$. It is well known that the bases of the lattice $\mathcal{L}$ are not unique, since the matrix $\mathbf{B}$ right-multiplied with a unimodular matrix $\mathbf{G}$, namely, $\mathbf{BG}$, is also a basis of $\mathcal{L}$, where $\mathbf{B} = (\mathbf{b}_1, \mathbf{b}_2, \ldots, \mathbf{b}_m)$. Among an infinite number of the bases of $\mathcal{L}$, some are much more efficient for solving problems of theoretical and practical importance from pure and applied science than the others, as already implied/demonstrated clearly by the first question posed to Hendrik Lenstra from Van Emde Boas and A. Marchetti-Spaccamela in 1980 that eventually led to the invention of the celebrated LLL algorithm [28].

The question now is how to find the unimodular matrix $\mathbf{G}$ such that the new lattice basis $\mathbf{BG}$ is optimal in a certain sense of optimality. As far as senses of optimality are specified and formulated as objective functions, reduction is then equivalent to solving an integer programming problem with one and/or multiple objective functions [22]. The conventional sense of optimality in the theory of lattice reduction would be twofold: (a) that all the reduced vectors are the shortest and (b) that all the reduced vectors are mutually orthogonal. Unfortunately, this combined sense of optimality is practically impossible to achieve, except for some trial types of bases. Even worse is that the shortest vector problem itself is conjectured to be NP-hard (see e.g., [42]), not to mention that finding a good reduced basis generally is only the means to help solve problems at hand but certainly not the final goal. Thus, almost all algorithms of practical importance for lattice basis reduction are either based on the Gram-Schmidt orthogonalization or the Householder QR factorization to naturally obtain a (suboptimal) unimodular matrix $\mathbf{G}$ under a certain condition for reducing the lengths of the basis vectors. Other senses of optimality include the metric for reduction defined by Seysen [40] and the minimization of the maximum variance of the integer-transformed real-valued solution proposed recently by Zhou and Ma [43]. Since LaMacchia [44] reported that Seysen's method of reduction cannot compete with the LLL algorithm, we will not pursue this method any further in this work.

2.1 The LLL algorithm

The LLL algorithm has been well documented in the literature, often given in the form of pseudo-codes (see e.g., [4,6,7,14,29,30,35]) and can even be easily available from the internet. The algorithm consists of two essential components, namely, the Gram-Schmidt orthogonalization and the Lovasz condition. Given the basis $\mathbf{b}_1, \mathbf{b}_2, \ldots, \mathbf{b}_m$ (assumed to be linearly independent as in the above), the Gram-Schmidt orthogonalization aims at making the reduced basis as orthogonal as possible and proceeds as follows:

$$\mathbf{b}_i^* = \mathbf{b}_i - \sum_{j=1}^{i-1} \mu_{ij} \mathbf{b}_j^*, \tag{2a}$$

where the Gram-Schmidt coefficient

$$\mu_{ij} = (\mathbf{b}_i, \mathbf{b}_j^*)/(\mathbf{b}_j^*, \mathbf{b}_j^*) \tag{2b}$$

is always assumed, without loss of generality, to fall between $-1/2$ and $1/2$, with $(\cdot, \cdot)$ standing for the Euclidean inner product on $\mathcal{R}^n$. In case that $|\mu_{ij}| > 1/2$, $\mathbf{b}_i$ is replaced with $(\mathbf{b}_i - \lceil \mu_{ij} \rfloor \mathbf{b}_j)$, where $\lceil \mu_{ij} \rfloor$ is the nearest integer to μ_{ij} [3]. Actually, a basis is called size-reduced, if $|\mu_{ij}| \leq 1/2$ $(1 \leq j < i \leq m)$.

To further make the reduced vectors as short as possible, the LLL algorithm implements the Lovasz condition, namely,

$$\delta \|\mathbf{b}_i^*\|^2 \leq \|\mathbf{b}_{i+1}^* + \mu_{(i+1)i} \mathbf{b}_i^*\|^2 \tag{3}$$

to decide whether the Gram-Schmidt orthogonalization procedure (2) should be temporarily suspended and the action of swapping between $\mathbf{b}_{i+1}$ and $\mathbf{b}_i$ should be taken. If the swapping is necessary, one has to exchange $\mathbf{b}_i$ with $\mathbf{b}_{i+1}$ and then set the current stage of $(i + 1)$ back to i in (2), before the orthogonalization (2) is reactivated. Lenstra et al. [3] proved that the procedure described can always converge in polynomial time. For more details, the reader is referred to Lenstra et al. [3].

Although pseudo-codes of the LLL algorithm are readily accessible, some implementations require updating the Gram-Schmidt coefficients μ_{kj} $(j < k)$ (see e.g., [3,6,7]). Actually, it is easy to prove that updating μ_{kj} is needed only if all μ_{kj} are computed in advance. Because we compute each μ_{kj} only when its turn comes and because the loop j, as implemented at Step S4 of Algorithm 1, runs from $(k - 1)$ to 1, it is not necessary for us to update

μ_{kj}. Actually, the procedure for updating μ_{kj} has been automatically implemented by the loop from S4 to S10 in Algorithm 1. For convenience of reference, we list our pseudo-codes of the LLL algorithm in Algorithm 1. Note, however, that if $k = 2$ and if a swapping is required at step S13 of Algorithm 1, then one will have to update $\mathbf{b}_1^*$ as well.

2.2 LLL algorithms with deep insertions

LLL algorithms with deep insertions were first proposed by Schnorr and Euchner [35] and have led to many more applications and further investigations, as clearly seen from an ever increasing long list of citing articles either on the Web site of Google Scholar or the ISI Web of Science. The basic idea of LLL algorithms with deep insertions is to use the same Gram-Schmidt orthogonalization process as the LLL algorithm to achieve almost orthogonality of the reduced lattice vectors but to replace the Lovasz condition with a stronger condition of vector swapping to further reduce the lengths of the reduced lattice vectors. In fact, following Lenstra et al. [3], we know that any reduced vector is upper-bounded by the following inequality (see e.g., [3]):

$$\|\mathbf{b}_i\|^2 \leq \alpha^{i-1}\|\mathbf{b}_i^*\|^2, \tag{4}$$

where $\alpha = 1/(\delta - 1/4)$ and $1/4 < \delta < 1$. In the case of the LLL algorithm, $\delta = 3/4$ and $\alpha = 2$.

Obviously, a smaller α and/or a smaller $\|\mathbf{b}_i^*\|$ directly result in a tighter upper bound of length for the reduced vector $\mathbf{b}_i$ and potentially indicate that the length of $\mathbf{b}_i$ can be further reduced in comparison with that from the LLL algorithm. More specifically, Schnorr and Euchner [35] proposed replacing the Lovasz condition (3) with the following stronger test:

$$\delta\|\mathbf{b}_i^*\|^2 \leq \|\mathbf{b}_l^* + \sum_{j=i}^{l-1}\mu_{lj}\mathbf{b}_j^*\|^2 \tag{5}$$

for all $1 \leq i < l$. The Lovasz condition (3) is a special case of (5) by restricting i to $(l-1)$. In other words, the LLL algorithm is of insertion with a unit depth. If (5) is violated, then a minimum index i is chosen and $\mathbf{b}_l$ is inserted right before $\mathbf{b}_i$. By setting l back to i, one can then resume the reduction procedure with deep insertions.

Because (5) applies for all $1 \leq i < l$, all the values $\|\mathbf{b}_i^*\|$ $(1 \leq i \leq m)$ from the LLL algorithm with deep insertions should be smaller than those from the LLL algorithm. In addition, Schnorr and Euchner [35] also proposed using a bigger value of $\delta = 0.99$, which leads to a smaller value of α. Nguyen and Stehlé [7] set δ to 0.999 in their experimental study of the performance of the LLL algorithm. Schnorr and Euchner [35] stated that the complexity of the LLL algorithm with deep insertions is super polynomial, with the published examples showing that its practical running time is longer by a few times than the original LLL algorithm. Gama and Nguyen [45] reported that the LLL algorithm with deep insertions is of super exponential complexity. One way to control the complexity of reduction with deep insertions is to limit the depth of insertions by setting $(l - i)$ in (5) to some constant. In our experiments to be reported in Section 4, we implement the control condition (5) without any restriction on i. More specifically, the implemented variant of LLL algorithms with deep insertions is to replace steps S12 to S15 of Algorithm 1 with the following deep insertion process [35] in Algorithm 2.

2.3 The LLL algorithm with the sorted QR ordering

A different ordering of the basis vectors could affect the running time and the reduction quality of the LLL algorithm and its different variants (see e.g., [19,36,37]). Both ascending and descending orderings of the basis vectors have been used for reduction (see e.g., [19,22,36,37]). Using two variants of the LLL algorithm with deep insertions, Backes and Wetzel [36,37] have shown that

Algorithm 1 Pseudo-codes of the LLL algorithm

```
S1 Input: the basis of lattice $\mathbf{b}_1, \mathbf{b}_2, \ldots, \mathbf{b}_m$
S2 Initialize: $k = 2$ and $\mathbf{b}_1^* = \mathbf{b}_1$
S3 while $k \leq m$
S4    for $j = (k-1)$ to 1 step $-1$
S5       compute $\mu_{kj}$
S6       if $|\mu_{kj}| > 0.5$
S7          set $\mu_{kj}$ to its nearest integer $\lceil\mu_{kj}\rfloor$
S8          replace $\mathbf{b}_k = \mathbf{b}_k - \lceil\mu_{kj}\rfloor\mathbf{b}_j$ and $\mu_{kj} = \mu_{kj} - \lceil\mu_{kj}\rfloor$
S9       end
S10   end
S11   compute $\mathbf{b}_k^*$
S12   if the Lovasz test (3) is true, continue to next $k$
S13   else swap $\mathbf{b}_k$ with $\mathbf{b}_{k-1}$ and set $k = \min(k-1, 2)$
S14   end
S15 end
```

Algorithm 2 Pseudo-codes of the LLL algorithm with deep insertions. The codes are identical with Algorithm 1, except for steps S12 to S15 of Algorithm 1 replaced with the following lines

```
S12A  for $i = 1$ to $(k-1)$
S13A     if $\delta\|\mathbf{b}_i^*\|^2 > \|\mathbf{b}_k^*\|^2 + \sum_{j=i}^{k-1}\mu_{kj}^2\|\mathbf{b}_j^*\|^2$,
S14A        insert $\mathbf{b}_k$ before $\mathbf{b}_i$ and set $k$ to $i$. If $k = 1$, goto
            S2 of Algorithm 1;
            otherwise, goto S4 of Algorithm 1
S15A     else continue to next $k$
S16A     end
S17A  end
```

sorting can indeed affect the practical running behaviors of the algorithms, but the extent of effect depends on the types of lattice bases. In one case, ascending order could speed up the reduction significantly. However, in the other case, it could increase the time of reduction substantially. Xu [19,22] has shown through numerical simulations that arranging the basis vectors in ascending order could improve the quality of reduction in the sense of producing a smaller condition number.

The sorted QR ordering has been popular in communications and can be very effective in constructing a suboptimal integer estimator (see e.g., [24,25,46-48]). The terminology of sorted QR directly came from the publication by Wübben et al. [47], although such a suboptimal integer estimator was first formulated by Xu et al. [24] in 1995 (see also [25]) in the language of minimum pivoting for Gaussian and/or Cholesky decompositions and was called a one-step, non-exact solution. Actually, Xu et al. [24] went one step further than Wübben et al. [47] by implementing the reduction of positive definite quadratic forms with the sorted QR ordering into the procedure to construct the suboptimal integer solution. The basic idea of the sorted QR ordering is to arrange the unknown integer parameters in the order of maximum conditional weightings on the basis of the normal matrix. It has been recently proposed by Gan and Mow [38] (see also [33,39]) as a component of the LLL algorithm for reduction of basis vectors and by Xu [22] for reduction of positive definite quadratic forms. The random simulations of Xu [22] have clearly shown the effectiveness of the sorted QR ordering to reduce the condition number of a positive definite quadratic form. For low-dimensional problems, it can significantly reduce the running time of reduction [22,38]. The average running time and performance of the methods can also be found in Ling and Mow [39]. In the case of positive definite quadratic forms, Xu [22] focused on the performance to reduce the condition number of a positive definite matrix. Thus, in this paper, we will include the LLL algorithm with the sorted QR ordering proposed by Gan and Mow [38] (see also [33,39]) for comparison. The algorithm is essentially the same as the LLL algorithm, except for that the vector with the minimum length projected onto the complement range of the subspace spanned by the orthogonalized vectors up to the present is first picked up for reduction/orthogonalization among the unreduced basis vectors. More specifically, the LLL algorithm with the QR sorting can be readily coded by replacing step S12 of Algorithm 1 with the lines shown in Algorithm 3. More details on the algorithm can also be found in Ling and Mow [33,39]. We should note, however, that Algorithm 3 may be said to be a special case of Xu [22]. Due to the Lovasz condition (3), the final reduced positive definite matrix from this version of LLL algorithms with the QR sorting does not necessarily match that of Xu [22].

Algorithm 3 The LLL algorithm with the QR sorting. The codes are identical with Algorithm 1, except for step S12 of Algorithm 1 replaced with the following lines

```
S12A  if the Lovasz test (3) is true
S13A      project the remaining vectors onto the complement range of the sub-space spanned by the orthogonalized vectors;
S14A      pick up the vector, whose projected length is minimum, as the next $\mathbf{b}_k$ and continue;
S15A  end
```

2.4 An LLL algorithm with fixed complexity

The flow of the LLL algorithm dynamically depends on a problem at hand and the corresponding arithmetic operations cannot be estimated precisely beforehand. Except for the worst-case complexity, one cannot know exactly when the LLL algorithm will terminate. In order to make the running behavior of the LLL algorithm completely countable in advance, Vetter et al. [14] proposed a fixed complexity LLL algorithm. Obviously, the uncontrollability of the LLL algorithm is solely due to the Lovasz condition (3). As a result, in order to clear this unpredictability, Vetter et al. [14] directly eliminated the winding step, namely, $k = \min(k-1, 2)$ from Algorithm 1. However, the swapping operation remains active, if the Lovasz condition (3) is violated. To compensate for prohibiting the progress counter k to step back in the LLL algorithm, they suggested executing the above procedure repeatedly for $(m-1)$ times (simply m in our implementation). The resulted algorithm is thus called *fixed complexity LLL algorithm* by Vetter et al. [14]. Since the algorithm will be used in our numerical simulations for comparison and since a complete set of pseudo-codes is not given in Vetter et al. [14], we list the pseudo-codes of this fixed complexity LLL algorithm in Algorithm 4.

2.5 Improved LLL algorithm with fixed complexity

To start developing our improved LLL algorithm with fixed complexity, let us assume (a) that the vectors of the sublattice $\mathcal{L}$, namely, $\mathbf{B} = (\mathbf{b}_1, \mathbf{b}_2, \ldots, \mathbf{b}_m)$ in $\mathcal{R}^n$ are linearly independent, as in the literature on lattice reduction and (b) that these vectors have been orthogonalized and can be rewritten, without loss of generality, as follows:

$$\mathbf{B} = \mathbf{B}^*\mathbf{L}^T, \tag{6a}$$

where $\mathbf{B}^*$ is an orthogonal matrix consisting of mutually orthogonal column vectors, the superscript T stands for

Algorithm 4 Pseudo-codes of the fixed complexity LLL algorithm by Vetter et al. [14]

```
S1  **Input:** the basis of lattice $\mathbf{b}_1, \mathbf{b}_2, \ldots, \mathbf{b}_m$
S2  **Initialize:** countLOOP = 1
S3  **while** countLOOP $\le m$
S4      set $k = 2$ and $\mathbf{b}_1^* = \mathbf{b}_1$
S5      **while** $k \le m$
S6          **for** j = $(k-1)$ **to** 1 **step** $-1$
S7              compute $\mu_{kj}$
S8              **if** $|\mu_{kj}| > 0.5$
S9                  set $\mu_{kj}$ to its nearest integer $\lceil \mu_{kj} \rfloor$
S10                 replace $\mathbf{b}_k = \mathbf{b}_k - \lceil \mu_{kj} \rfloor \mathbf{b}_j$ and $\mu_{kj} = \mu_{kj} - \lceil \mu_{kj} \rfloor$
S11             **end**
S12         **end**
S13         compute $\mathbf{b}_k^*$
S14         **if** the Lovasz test (3) is not true,
S15             swap $\mathbf{b}_k$ with $\mathbf{b}_{k-1}$ and $\mathbf{b}_k^*$ with $\mathbf{b}_{k-1}^*$
S16         **end**
S17         continue to next $k$
S18     **end**
S19     if the Lovasz test (3) is true for all $k$, terminate
S20     continue to the next countLOOP
S21 **end**
```

transpose, and $\mathbf{L}$ is a lower triangular matrix with all its diagonal elements being equal to unity, namely,

$$\mathbf{L} = \begin{bmatrix} 1 & & & & \\ l_{21} & 1 & & & \\ l_{31} & l_{32} & 1 & & \\ \vdots & \vdots & \vdots & \ddots & \\ l_{m1} & l_{m2} & l_{m3} & \ldots & 1 \end{bmatrix}. \tag{6b}$$

The elements of $\mathbf{L}$ are essentially the original Gram-Schmidt coefficients, namely, $l_{ij} = (\mathbf{b}_i, \mathbf{b}_j^*)/(\mathbf{b}_j^*, \mathbf{b}_j^*)$, with $\mathbf{b}_j^*$ being the jth column vector of $\mathbf{B}^*$.

$\mathbf{L}$ of (6b) can be rewritten as the product of a unimodular matrix $\mathbf{G}$ and a new lower triangular matrix $\mathbf{L}_\mu$, namely,

$$\mathbf{L} = \mathbf{G}\mathbf{L}_\mu, \tag{7}$$

where

$$\mathbf{L}_\mu = \begin{bmatrix} 1 & & & & \\ \mu_{21} & 1 & & & \\ \mu_{31} & \mu_{32} & 1 & & \\ \vdots & \vdots & \vdots & \ddots & \\ \mu_{m1} & \mu_{m2} & \mu_{m3} & \ldots & 1 \end{bmatrix},$$

and all the elements μ_{ij} satisfy

$$|\mu_{ij}| \le 0.5, \quad (i > j).$$

Substituting (7) into (6a) yields

$$\mathbf{B} = \mathbf{B}^*\mathbf{L}_\mu^T\mathbf{G}^T. \tag{8}$$

By treating $\mathbf{B}^*\mathbf{L}_\mu^T$ as $\mathbf{B}$ and repeating the above procedure from (6a) to (8), we can then finally obtain the reduced basis $\mathbf{B}(= \mathbf{B}^*\mathbf{L}_\mu^T)$.

Proposition 1. Given a set of m linearly independent vectors $\mathbf{b}_1, \mathbf{b}_2, \ldots, \mathbf{b}_m$ in $\mathcal{R}^n$, the reduction by repeating the process from (6a) to (8) always converges in a finite number of iterations.

The proof of the proposition is trivial. In fact, given two linearly independent vectors $\mathbf{a}$ and $\mathbf{b}$, it is trivial to prove that $(\mathbf{b}^*, \mathbf{b}^*) \le (\mathbf{b}, \mathbf{b})$, if the size reduction is active, namely, $|r| \ge 1$, where $\mathbf{b}^* = \mathbf{b} - r\mathbf{a}$ and $r = \lceil (\mathbf{a}, \mathbf{b})/(\mathbf{a}, \mathbf{a}) \rfloor$. If we assume that the process described in the proposition does not converge, this means that there always exists, at least, one non-zero integer r to reduce the length of a vector. In other words, the determinant of the reduced basis, or equivalently, $\det\{(\mathbf{B}^*)^T\mathbf{B}^*\}$, can be arbitrarily small. However, this contradicts with the well-known fact that $\det\{\mathbf{B}^T\mathbf{B}\}$ is invariant for a given lattice. Similar work may be found in Ling and Mow [39], though they did not summarize their related work as clearly as we state in proposition 1 with the help of formulae (6a) to (8). Nevertheless, we should note that proposition 1 is still slightly different from the work of Ling and Mow [39] in two senses: (a) while Ling and Mow [39] directly implemented the sorted QR technique to re-arrange the basis vectors, we do not assume any sorting in proposition 1 and (b) as a result of (a), the proofs given here and in Ling and Mow [39] are essentially different.

From a formal point of view, proposition 1 is complete to serve as a protocol of reduction algorithm. In order to turn it into an efficient reduction algorithm, we will focus on two heuristic factors: (a) sorting the basis vectors $\mathbf{b}_1, \mathbf{b}_2, \ldots, \mathbf{b}_m$ to construct the matrix $\mathbf{L}$ of (6b) such that the lengths of the basis vectors can be maximally reduced quickly and, as a result, the running time of the algorithm can be significantly saved and (b) the complexity of the algorithm. Ling and Mow [33,39] proposed applying the sorted QR technique, as originally invented by Xu et al. [24] in 1995 to construct a suboptimal integer estimator, to sort the basis vectors. Although the QR-sorting strategy is very powerful in obtaining a suboptimal integer solution, it was shown to perform less efficiently for the reduction of positive definite quadratic forms [22]. Actually, in the development of a Cholesky-based reduction algorithm with fixed complexity for positive definite quadratic forms, Xu [22] found that two sorting strategies are very powerful to reduce the condition number of a positive definite quadratic form. One such sorting technique is to sort the vectors $\mathbf{b}_1, \mathbf{b}_2, \ldots, \mathbf{b}_m$ according to the ascending order of their lengths, which will be referred to as *ascending sorting* and abbreviated by *ASCE*. The other is to implement a perturbation to the first sorting strategy, which will be referred to as *perturbed sorting* and abbreviated by *PERT*. More precisely speaking, the second perturbed sorting technique PERT is implemented as follows: to start the reduction algorithm, we first follow the sorting strategy ASCE. In the following one or two iterations, we sort the vectors $\mathbf{b}_1, \mathbf{b}_2, \ldots, \mathbf{b}_m$ according to the ascending order of the lengths of the orthogonalized vectors $\mathbf{b}_1^*, \mathbf{b}_2^*, \ldots, \mathbf{b}_m^*$. Then we return to the sorting strategy ASCE and use it until the termination of the reduction algorithm. Since the two sorting strategies ASCE and PERT can be run in parallel, they are assembled together to construct our improved (parallel) LLL algorithm (For more details on these and other sorting techniques, the reader is referred to Xu [22]).

Now we face the same situation as in the case of the LLL algorithm, i.e., that we do not know exactly when our reduction algorithm will terminate. In order to make the arithmetic operations of the reduction algorithm predictable in advance, we can limit the number of iterations by setting a maximum value, say $K_{\max}$. By doing so, the algorithm either terminates naturally or when the iteration number hits $K_{\max}$. However, according to the experience of numerical simulations [19,22] more iterations can improve the reduction quality in terms of condition numbers slightly but can also worsen such quality measure. Therefore, we set the maximum number of iterations $K_{\max}$ to the rank of lattice m and finish constructing our improved (parallel) LLL algorithm with fixed complexity. We note, however, that a $K_{\max}$ larger than $3m$ is not recommended. In the final version used to report the results in Section 4, we set $K_{\max}$ to 15 if $m \leq 15$.

Thus, we are now in a position to assemble what we described in the above in the form of an algorithm with fixed complexity in Algorithm 5. Algorithm 5 is parallel in the sense that either step S4A or S4B can be used independently. Since condition numbers can be thought of as a combined quality measure of orthogonality and length defects, the final output reduced basis from Algorithm 5 is the one with a smaller condition number. We should note that Algorithm 5 is different from the parallel LLL-deep algorithm by Ling and Mow [39] in the sense that they used the sorted QR strategy in the lines S4A and S4B. The two sorting strategies of S4A and S4B will be shown to perform much better than the sorted QR for reduction in Section 4.

Compared with the LLL algorithm [3] and its fixed complexity variant by Vetter et al. [14], the improved LLL algorithm with fixed complexity has two significant features: (a) the LLL algorithm and its published variants perform the size reduction on its individual Gram-Schmidt coefficient μ_{ij}. This operation is also on and off without a natural smooth flow, depending on the switch control by the Lovasz test. The improved LLL algorithm with fixed complexity directly works on all the Gram-Schmidt

Algorithm 5 Pseudo-codes of the improved LLL algorithm with fixed complexity

```
S1   Input: the basis of lattice $\mathbf{b}_1, \mathbf{b}_2, \ldots, \mathbf{b}_m$
S2   Initialize: countLOOP = 1
S3   while countLOOP $\leq K_{\max}$
S4A      use the sorting strategy ASCE to sort $\mathbf{b}_1, \mathbf{b}_2, \ldots, \mathbf{b}_m$, or
S4B      use the sorting strategy PERT to sort $\mathbf{b}_1, \mathbf{b}_2, \ldots, \mathbf{b}_m$
S5       compute the Gram-Schmidt orthogonalization (6a)
S6       reduce the matrix $\mathbf{L}$ to get $\mathbf{L}_\mu$ via (7)
S7       if no reduction is possible, terminate
S8       replace $\mathbf{B}$ with $\mathbf{B}^*\mathbf{L}_\mu^T$
S9       continue to the next countLOOP
S10  end
```

coefficients μ_{ij} simultaneously. As a result, a global optimal size reduction could be achieved at each iteration. From this point of view, we might say that the LLL algorithm and its other variants are only locally optimal in size reduction; and (b) unlike the LLL algorithm and its known variants, the improved LLL algorithm with fixed complexity requires no Lovasz test. Thus, the flow of algorithmic actions is completely transparent and smooth.

3 Quality measures of lattice basis reduction

As is well known, the goals of lattice reduction are to make the reduced basis as orthogonal as possible and to make the lengths of the reduced basis vectors as short as possible. Thus, quality measures of lattice reduction should directly be associated with the goals of reduction. Three most widely used quality measures are the Hermite defect, the length defect, and the orthogonality defect, which are denoted by $\mathcal{H}(\mathbf{B})$, $l(\mathbf{B})$, and $\mathcal{O}(\mathbf{B})$, respectively, and given as follows (see e.g., [29,31]):

$$\mathcal{H}(\mathbf{B}) = \frac{\|\mathbf{b}_1\|^2}{[\det(\mathcal{L})]^{1/m}}, \tag{9a}$$

$$\ell(\mathbf{B}) = \frac{\|\mathbf{b}_1\|}{\lambda(\mathcal{L})}, \tag{9b}$$

and

$$\mathcal{O}(\mathbf{B}) = \frac{\prod_{i=1}^{m} \|\mathbf{b}_i\|}{[\det(\mathcal{L})]^{1/2}}, \tag{9c}$$

where $\mathbf{b}_1$ is the shortest reduced basis vector, $\det(\mathcal{L})$ is the determinant of the lattice $\mathcal{L}$ and is equal to $\det\{\mathbf{B}^T\mathbf{B}\}$, $\lambda(\mathcal{L})$ is the first minimum of $\mathcal{L}$. A length defect can also be defined as the ratio of the length of $\mathbf{b}_i$ to the ith successive minimum of $\mathcal{L}$ [31]. The Hermite defect (9a) and the length defect (9b) may be interpreted to evaluate the mean and absolute improvements of the length of the shortest reduced vector against the lattice $\mathcal{L}$ and its first minimum $\lambda(\mathcal{L})$, respectively. Because the Hermite defect $\mathcal{H}(\mathbf{B})$ of

(9a) remains exponential with a power roughly equal to the rank m, Nguyen and Stehlé [7] suggested replacing (9a) by the Hermite factor γ_B to measure the output quality of reduction, which can be defined through the following relationship:

$$\gamma_B^m = \frac{\|\mathbf{b}_1\|}{[\det(\mathcal{L})]^{1/(2m)}}. \tag{10}$$

The factor of two in the power of $\det(\mathcal{L})$ on the right-hand side of (10) is due to the difference in defining the determinant of a lattice. More precisely speaking, Nguyen and Stehlé [7] defined $\det(\mathcal{L})$ as the square root of $\det\{\mathbf{B}^T\mathbf{B}\}$.

Since finding $\lambda(\mathcal{L})$ is conjectured to be NP-hard, $\ell(\mathbf{B})$ of (9b) is more of theoretical value but likely is not a practical quality measure of reduction. Furthermore, both $\det(\mathcal{L})$ and $\lambda(\mathcal{L})$ are invariant for a given lattice $\mathcal{L}$. If we are concerned with the comparison of different reduction methods, we can simply focus on $\|\mathbf{b}_1\|$ only and denote

$$\ell_1(\mathbf{B}) = \|\mathbf{b}_1\|. \tag{11}$$

As an alternative quality measure to the Hermite defect and the length defect, one may define a new quality measure of length defect as the length ratio of the longest basis vector to the shortest one, namely,

$$r(\mathbf{B}) = \frac{\max\{\|\mathbf{b}_2\|, \ldots, \|\mathbf{b}_m\|\}}{\|\mathbf{b}_1\|}, \tag{12}$$

where $\mathbf{b}_1$ has been defined in (9a). Obviously, a best reduction method should result in the minimum $r(\mathbf{B})$.

For the ILS problem of minimizing $(\mathbf{z}-\mathbf{z}_f)^T\mathbf{W}_f(\mathbf{z}-\mathbf{z}_f)$, the absolute lengths of the basis vectors are not important since they can be made arbitrarily small without affecting the solution to the ILS problem [20], even though making the reduced basis as short as possible has been a goal of reduction. Here $\mathbf{z}$ and $\mathbf{z}_f$ are the unknown integer vector to be estimated and a real-valued vector, respectively, and $\mathbf{W}_f$ is a positive definite (weight) matrix. Actually, given two positive definite weight matrices $\mathbf{W}_f$ and $\alpha\mathbf{W}_f$, it is trivial to prove that both matrices lead to the same ILS estimator, no matter how small the positive scalar α is. From this point of view, a quality measure of reduction for ILS problems should emphasize the relative lengths of the reduced basis instead of their absolute lengths. In other words, a good quality measure of reduction should minimize the maximum relative length of the reduced basis vectors for an ILS problem. A natural quality measure of this type is the condition number in association with the ILS problem, which has been shown to be very powerful in evaluating the performance of reduction methods [19]. Actually, the condition number may be interpreted as a combined quality measure of length defect and orthogonality defect [22]. The smaller the condition number, the better a reduction method can be said to be. In the case of the lattice $\mathcal{L}$, the condition number can be defined as follows:

$$\kappa_B = \lambda_{\max}/\lambda_{\min}, \tag{13}$$

where $\lambda_{\max}$ and $\lambda_{\min}$ are the maximum and minimum singular values of the matrix $\mathbf{B}$, respectively.

The orthogonality defect (9c) is defined on the basis of Hadamard's inequality:

$$\det(\mathcal{L}) \le \prod_{i=1}^{m} \|\mathbf{b}_i\|^2. \tag{14}$$

If the reduced basis is mutually orthogonal, (14) becomes an identity. The idealized minimum value of the orthogonality defect $\mathcal{O}(\mathbf{B})$ is equal to unity. However, there exists no upper bound for $\mathcal{O}(\mathbf{B})$ of (9c). As is well known, the smaller $\mathcal{O}(\mathbf{B})$, the better a reduction method. The question is that we do not have any operational objective criterion to judge whether a reduction basis is sufficiently orthogonal from its value of $\mathcal{O}(\mathbf{B})$ in $[1, \infty)$. In other words, although a reduction is to make the reduced basis as orthogonal as possible, unfortunately, we cannot practically tell the extent of orthogonality of the reduced basis from the orthogonality defect $\mathcal{O}(\mathbf{B})$.

As a result, we define the minimum angle among the reduced basis vectors of $\mathcal{L}$ as an alternative quality measure of orthogonality, which can be written as follows:

$$\theta(\mathbf{B}) = \min\{\theta_{ij},\ 1 \le i < j \le m\}, \tag{15}$$

where

$$\theta_{ij} = \min\{\arccos(\rho_{ij}),\ 180^o - \arccos(\rho_{ij})\},$$

$$\rho_{ij} = \frac{(\mathbf{b}_i, \mathbf{b}_j)}{\|\mathbf{b}_i\|\|\mathbf{b}_j\|},$$

and $\arccos(\rho_{ij})$ is given in degrees. By definition, we have $0^o \le \theta(\mathbf{B}) \le 90^o$. If all the basis vectors are mutually orthogonal, $\theta(\mathbf{B}) = 90^o$. Based on the quality measure (15) of orthogonality, we can now be quite confident to say intuitively that a good reduction method should almost always guarantee an angle above 45^o (ideally 60^o in the best case) for $\theta(\mathbf{B})$ of (15). As an alternative quality measure of orthogonality, $\theta(\mathbf{B})$ of (15) may be intuitively more appealing than $\mathcal{O}(\mathbf{B})$ of (9c), since, given a value of $\theta(\mathbf{B})$, we can immediately have an idea in our mind on how orthogonal the reduced basis looks like. We should note, however, that the computation of $\theta(\mathbf{B})$ is not more difficult than that of $\mathcal{O}(\mathbf{B})$, since both $\theta(\mathbf{B})$ and $\mathcal{O}(\mathbf{B})$ are solely based on the elements of the matrix $\mathbf{B}^T\mathbf{B}$.

4 Experiments and analysis of results

4.1 Numerical simulation of random lattice bases

Broadly speaking, a lattice can be said to be random, if there exists, at least, one random element in any of the basis vectors $\mathbf{B}$. Probabilistic models such as the uniform distribution in the unit ball or on the unit sphere have played an important role in understanding the mean practical running behavior of the LLL algorithm and in the derivation of statistical mean values of quantities of the reduced basis (see e.g., [29-31]). Ajtai [49,50] demonstrated the worst-case performance of an LLL variant using the following random basis:

$$\mathbf{b}_i = f_{ii}\mathbf{e}_i + f_{i(i-1)}\mathbf{e}_{i-1}/2 + \sum_{j=1}^{i-2} f_{ij}\mu_{ij}\mathbf{e}_j, \tag{16}$$

where

$$\begin{aligned} f_{ii} &= k^{c(m-i+1)/k}, \\ f_{i(i-1)} &= f_{(i-1)(i-1)}, \\ f_{ij} &= f_{jj}, \end{aligned}$$

$\mathbf{e}_i$ is the ith standard/natural basis vector in a Euclidean space, k is an integer of the size roughly equivalent to a fractional part of m, and c is a positive constant. μ_{ij} are all assumed to be independent random variables with a uniform distribution over $[-1/2, 1/2]$. A modified version by making all the lower-triangular elements become random with a uniform distribution can be found in Nguyen and Stehlé [7] and Vallée and Vera [29] and is called random bases of the Ajtai's type. Likely, the most widely used random lattice with a lot of applications is of the knapsack type and defined by the row vectors of the following matrix (see also [4,7,29]):

$$\begin{bmatrix} a_1 & 1 & 0 & \dots & 0 \\ a_2 & 0 & 1 & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ a_m & 0 & 0 & 0 & 1 \end{bmatrix}, \tag{17}$$

where all the elements a_i are random and uniformly distributed independently.

A particular distribution tends to generate random bases with some particular statistical features and might affect lattice basis reduction without our *a priori* knowledge. Thus, as a basic principle to guide the simulation of random lattices for our experiments, we require that (a) all the elements of **B** must be random, and (b) the basis **B** be generated from a non-informative referential system. As a result, we decide to generate the random bases using the decomposition:

$$\mathbf{B} = \mathbf{U}^T\mathbf{S}\mathbf{V}, \tag{18}$$

where **U** and **V** are non-informative referential systems of different dimensions and **S** contains all the nonzero (positive) singular values of **B**. More precisely, **U** and **V** can be generated using the non-informative probabilistic model for referential systems, and the nonzero elements of **S** are generated using a uniform distribution. As a matter of fact, if all the elements of a random matrix are of identical and independent normal distributions with mean zero, then its eigenvector matrix is non-informative [51,52]. Thus we can first simulate a standard Gaussian matrix and then decompose it to obtain **U** and **V**. Actually, these guiding rules were first suggested and used by Xu [19,22].

More specifically, we simulate $10,000$ random examples of **B**, with the number of columns uniformly distributed over $[3, 60]$ and the number of rows uniformly distributed over $[m, 800]$. In our experiments, we decide to set the maximum rank of lattice to 60, since finding the exact solution to the shortest vector problem up to such a dimension is still foreseeable [45]. In particular, almost all practical applications of GPS kinematic applications are low-dimensional (see e.g., [26,27]). The condition numbers of the simulated examples range from 10 to 5×10^4.

For convenience of discussing the experiment results in the remainder of this section, we will use the abbreviations of *PROB*, *LLL*, *DEEP*, *SLLL*, *VLLL*, and *PLLL* to denote the original random examples, the LLL algorithm (Algorithm 1 with the original $\delta = 0.75$), the LLL algorithm with deep insertions (Algorithm 2 with the original $\delta = 0.99$), the LLL algorithm with the sorted QR ordering (Algorithm 3), the LLL algorithm with fixed complexity by Vetter et al. [14] (Algorithm 4), and our improved LLL algorithm with fixed complexity (Algorithm 5), respectively. We may note that we choose the original δ value in our experiments since computation time for reduction would increase significantly with the increase of δ. Nevertheless, in low-dimensional GPS applications, reduction is only an intermediate procedure for integer estimation; thus, a significant increase of reduction time is highly not desirable.

4.2 Practical running behaviors of variants of the LLL algorithm

Given an integer basis for a lattice $\mathcal{L}$, Lenstra et al. [3] have proved that the LLL algorithm requires

$$K_B = O(m^2 \log b_{\max}) \tag{19}$$

iterations to terminate in the worst case, where $b_{\max}$ is the maximum length of the integer basis **B**. Alternatively, Daudé and Vallée [30] proposed an improved worst-case bound for the number of iterations as follows:

$$K_l = O(m^2 \log(b^*_{\max}/b^*_{\min})), \tag{20}$$

where $b^*_{\max}$ and $b^*_{\min}$ are the maximum and minimum lengths of the orthogonalized basis vectors, respectively. Recently, Jaldén et al. [13] suggested replacing $(b^*_{\max}/b^*_{\min})$ in (20) with the condition number of **B** and obtained a new worst case bound:

$$K_\kappa = O(m^2 \log \kappa_B), \tag{21}$$

where κ_B is the condition number of **B**.

However, it has been widely reported that the LLL algorithm runs surprisingly much faster than the theoretical worst complexity bound predicts (see e.g., [7,29,34,45,53]). Experiments have only been carried out recently to demystify and explain nice practical behavior of the LLL algorithm in terms of running time and output quality (see also [7,34,45,53]). Nguyen and Stehlé [7] reported that the bound of iterations (19) seems to be tight for random lattices of Ajtai's type, but might be relaxed for lattice bases of Knapsack type. The experiments by Gama and Nguyen [45] clearly demonstrated that the running time of Schnorr-Euchner's algorithm of enumeration aided by the LLL algorithm with deep insertions to solve the shortest vector problem is super-exponential.

Based on the 10,000 random examples, we will continue and complement the investigation of mean practical running behaviors by Nguyen and Stehlé [7] and Gama and Nguyen [45], in the sense that: (a) they [7,45] only tested the upper bound (19) with the simulated results from LLL and DEEP, but we will test all the three upper bounds (19), (20), and (21) with the results from LLL, DEEP, and SLLL; and (b) the random bases used in our simulations are neither of Ajtai's type nor of Knapsack type, as used by Nguyen and Stehlé [7] and Gama and Nguyen [45] in their study.

To begin with, for each of the 10,000 random examples, we have recorded the numbers of iterations for the three basis reduction methods, namely, LLL, DEEP, and SLLL, which are denoted by K^i_{LLL}, K^i_{DEEP}, and K^i_{SLLL}, respectively, where the superscript i stands for the ith random example. In order to understand the practical running behavior of DEEP and compare it with that of LLL, we have computed the ratio

$$\rho^i_{\mathrm{DEEP}} = K^i_{\mathrm{DEEP}}/K^i_{\mathrm{LLL}}, \ (i = 1, 2, \ldots, 10,000). \tag{22}$$

The mean and maximum values of ρ^i_{DEEP} for each rank of lattice are shown in Figure 1. Obviously, they increase with the increase of the rank of a lattice, indicating that the practical running behavior of DEEP is exponential, at least, for the random lattices under investigation. The results of experimental complexity support the report of super-exponential complexity of the LLL algorithm with deep insertions by Gama and Nguyen [45]. In other words, DEEP may not be super-polynomial in the worst case, as otherwise mentioned by Schnorr and Euchner [35].

In order to investigate the tightness of the upper bounds of iterations (19), (20), and (21), we have computed the following indices:

$$\rho_{\mathrm{J}}(\mathbf{B}) = K_{\mathrm{J}}/(m^2 \log b_{\max}), \tag{23a}$$
$$\rho_{\mathrm{J}}(l) = K_{\mathrm{J}}/(m^2 \log(b^*_{\max}/b^*_{\min})), \tag{23b}$$
$$\rho_{\mathrm{J}}(\kappa) = K_{\mathrm{J}}/(m^2 \log \kappa_B), \tag{23c}$$

for each of the 10,000 examples, where the subscript J stands for each of the basis reduction methods, namely, LLL, SLLL and DEEP, respectively. To understand the practical tightness of K_B in (19), we have plotted $\rho_{\mathrm{LLL}}(\mathbf{B})$, $\rho_{\mathrm{SLLL}}(\mathbf{B})$ and $\rho_{\mathrm{DEEP}}(\mathbf{B})$ for all the 10,000 examples, together with their mean values at each rank of lattice, in Figure 2. Note, however, that we have encountered a few negative values of $\rho_{\mathrm{J}}(\mathbf{B})$ for small m values, since we ignore the condition of integer lattice required by the index $\rho_{\mathrm{J}}(\mathbf{B})$. These few values are simply neglected and not shown in Figure 2. In a similar manner to gain the experimental information on K_l in (20) and K_κ in (21), we have shown $\rho_{\mathrm{LLL}}(l)$, $\rho_{\mathrm{SLLL}}(l)$, and $\rho_{\mathrm{DEEP}}(l)$ in Figure 3 and $\rho_{\mathrm{LLL}}(\kappa)$, $\rho_{\mathrm{SLLL}}(\kappa)$, and $\rho_{\mathrm{DEEP}}(\kappa)$ in Figure 4, respectively.

Figures 2, 3, and 4 have clearly shown a number of patterns: (a) all the three indices to evaluate the upper bounds of iterations, namely, $\rho_{\mathrm{J}}(\mathbf{B})$ of (23a), $\rho_{\mathrm{J}}(l)$ of (23b), and $\rho_{\mathrm{J}}(\kappa)$ of (23c), behave more or less similarly for each of the three methods LLL, SLLL, and DEEP. More precisely speaking, ρ_{LLL} and ρ_{SLLL} decrease with the increase of m and seem to converge to a small constant, no matter which of $\mathbf{b}_{\max}$, $\mathbf{b}^*_{\max}/\mathbf{b}^*_{\min}$ and/or κ_B in (23) is used in association with them, as can be clearly seen from panels A and B of Figures 2, 3, and 4. Thus, we may conclude that LLL and SLLL could run much faster than the theoretical bounds of iterations, as given in (19), (20), and (21). In other words, as for LLL and SLLL, all the bounds (19), (20), and (21) are not tight for the lattices under study. In fact, we also tried to fit the green LLL and red SLLL curves to the analytical function a/m^b. The values of b are found to be between 1.7 and 1.8 for $\mathbf{b}_{\max}$ and $\mathbf{b}^*_{\max}/\mathbf{b}^*_{\min}$. The value of a from SLLL is about half of that from LLL. In the case of κ_B, the values of b are about 2.2, but with the values of a being equal to 1,572.4 and 908.0 for LLL and SLLL, respectively. These results indicate that practical running behavior of the LLL algorithm may be much better than what the statistical mean behaviors have predicted; (b) the red lines of panel D of Figures 2, 3, and 4 are all consistently below the green. This should indicate that on average, SLLL runs faster than LLL, as also consistently confirmed by the fitting results to the green and red lines; and (c) the average behavior of DEEP tends to decrease with the increase of m (compare the black lines of panel D of Figures 2, 3, and 4), implying that the average running behaviors of DEEP may be polynomial. However, the maximum values of ρ_{DEEP} clearly increase with the increase of m (compare panel C of Figures 2, 3, and 4), indicating that its worst case complexity is exponential. This observation is consistent with the statement of super-exponential complexity about DEEP by Gama and Nguyen [45].

4.3 Performance of the five lattice basis reduction algorithms

We will now compare all the five basis reduction methods, namely, LLL, DEEP, SLLL, VLLL, and PLLL, based on the 10,000 randomly simulated examples and in terms of the

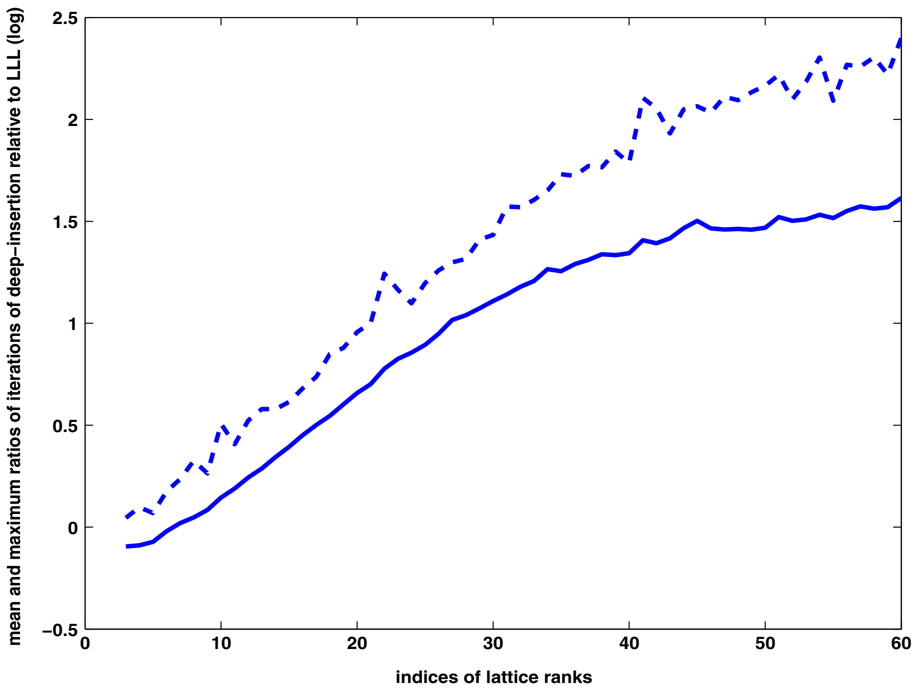

Figure 1 Practical running behaviors of LLL algorithm with deep insertions relative to original LLL algorithm. Shown in this figure are the mean and maximum values of log ρ^{i}_{DEEP} for each rank of lattice, which are displayed in solid and dashed lines, respectively.

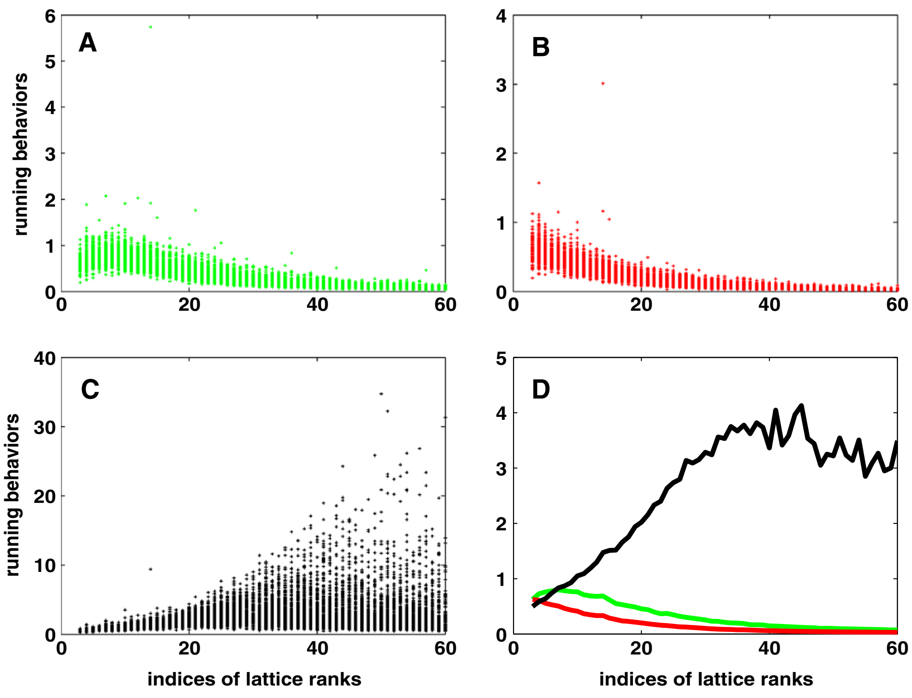

Figure 2 Tightness of K_B in (19) for three reduction methods LLL, SLLL, and DEEP with 10, 000 examples. panel **A** - $\rho_{\mathrm{LLL}}(\mathbf{B})$; panel **B** - $\rho_{\mathrm{SLLL}}(\mathbf{B})$; panel **C** - $\rho_{\mathrm{DEEP}}(\mathbf{B})$; and panel **D** - mean values of $\rho_{\mathrm{LLL}}(\mathbf{B})$ (green line), $\rho_{\mathrm{SLLL}}(\mathbf{B})$ (red line) and $\rho_{\mathrm{DEEP}}(\mathbf{B})$ (black line) for each rank of lattices.

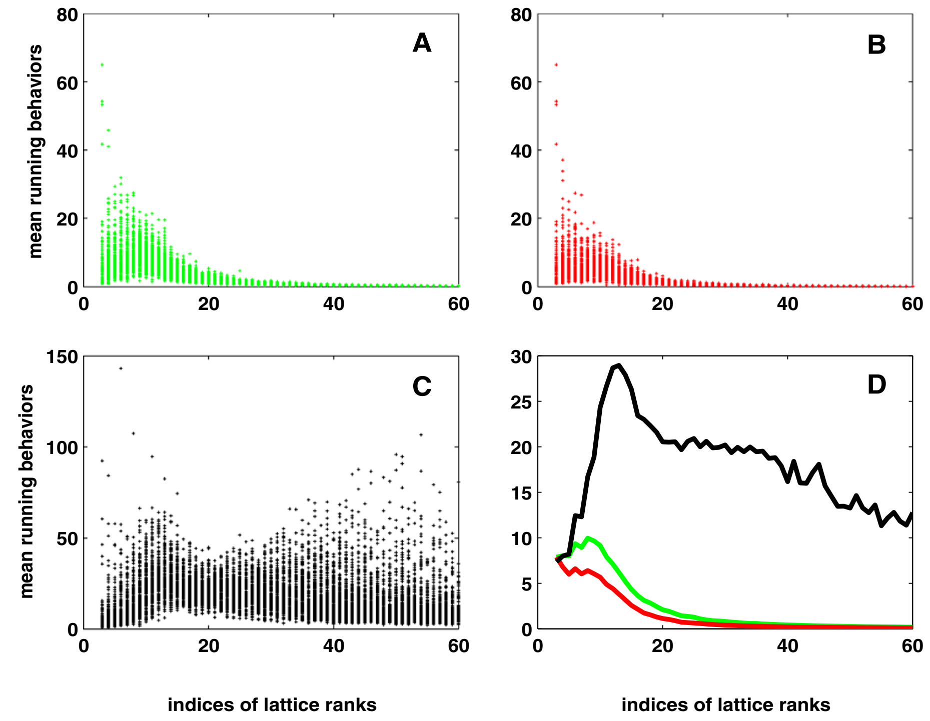

Figure 3 Tightness of K_l in (20) for three reduction methods LLL, SLLL, and DEEP with 10, 000 examples. panel **A** - $\rho_{\text{LLL}}(l)$; panel **B** - $\rho_{\text{SLLL}}(l)$; panel **C** - $\rho_{\text{DEEP}}(l)$; and panel **D** - mean values of $\rho_{\text{LLL}}(l)$ (green line), $\rho_{\text{SLLL}}(l)$ (red line), and $\rho_{\text{DEEP}}(l)$ (black line) for each rank of lattices.

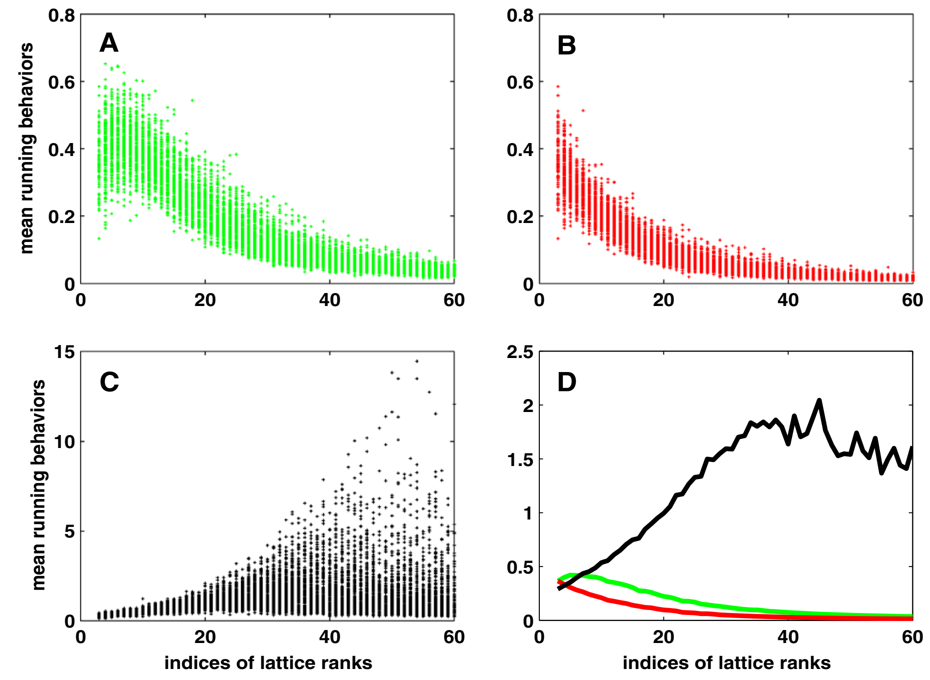

Figure 4 Tightness of K_κ in (21) for three reduction methods LLL, SLLL, and DEEP with 10, 000 examples. panel **A** - $\rho_{\text{LLL}}(\kappa)$; panel **B** - $\rho_{\text{SLLL}}(\kappa)$; panel **C** - $\rho_{\text{DEEP}}(\kappa)$; and panel **D** - mean values of $\rho_{\text{LLL}}(\kappa)$ (green line), $\rho_{\text{SLLL}}(\kappa)$ (red line), and $\rho_{\text{DEEP}}(\kappa)$ (black line) for each rank of lattices.

six quality measures of reduction discussed in Section 3. More precisely, the six quality measures used to compare the basis reduction methods are (a) the orthogonality defect $\mathcal{O}(\mathbf{B})$ of (9c); (b) the minimum angle among the reduced vectors, namely, $\theta(\mathbf{B})$ of (15); (c) the Hermite factor γ_B of (10); (d) the length $\ell_1(\mathbf{B})$ of the shortest reduced vector $\mathbf{b}_1$ in (11); (e) the maximum length ratio $r(\mathbf{B})$ of (12); and finally, (f) the condition number κ_B of (13). The first two quality measures, i.e., $\mathcal{O}(\mathbf{B})$ and $\theta(\mathbf{B})$, are related to the orthogonality of a reduced basis, the quality measures γ_B, $\ell_1(\mathbf{B})$, and $r(\mathbf{B})$ mainly reflect the length reduction of the reduced basis, while the condition number κ_B is a combined quality measure of orthogonality and length reduction. The Hermite factor has been theoretically given in Lenstra et al. [3] and recently investigated experimentally (see e.g., [7,34,45]), and the condition number κ_B of (13) as a quality measure of reduction has been substantially studied experimentally (see e.g., [19,22]). However, no experimental results on the other four quality measures have ever been reported in the literature, at least, to the best knowledge of this author.

Before we come to a particular quality measure, let us briefly explain how we compare reduction methods and compute/estimate the probabilities PBetter and PWorse listed in the succeeding tables. Let us assume that we now would like to compare two reduction methods I and J ($I, J \in \{\text{LLL}, \text{DEEP}, \text{SLLL}, \text{VLLL}, \text{PLLL}, \text{PROB}\}$) on the basis of a particular quality measure, say m_q. With the $10,000$ m_q values for each of I and J on hand, we can count the number of examples n_I with which I performs better than J and the number of examples n_J with which J performs better than I with respect to this quality measure m_q. When we compare I with J, we assign $n_I/10,000$ to PBetter and $n_J/10,000$ to PWorse in the succeeding tables. Actually, $n_I/10,000$ and $n_J/10,000$ correspond to the estimated frequency/probability with which I performs better and with which J performs better, respectively. Following this notion, we compare the results from LLL, DEEP, SLLL, VLLL, and PLLL with the original problems on the basis of the quality measures $\mathcal{O}(\mathbf{B})$, $\theta(\mathbf{B})$, $\ell_1(\mathbf{B})$, $r(\mathbf{B})$, and κ_B, and list the estimated probabilities in Table 1, where PBetter stands for the probabilities with which the five basis reduction methods improve (or perform better than) the original problems on the corresponding quality measure, respectively.

Table 1 Probabilities estimated by comparing LLL, DEEP, SLLL, VLLL, and PLLL with original problems

Measures	Methods	LLL	DEEP	SLLL	VLLL	PLLL
$\mathcal{O}(\mathbf{B})$	PBetter	0.5020	0.6152	0.5693	0.4730	0.6336
	PWorse	0.4980	0.3848	0.4307	0.5270	0.3664
$\theta(\mathbf{B})$	PBetter	0.3128	0.4120	0.3418	0.1937	0.4638
	PWorse	0.6828	0.5866	0.6540	0.8019	0.5280
$\ell_1(\mathbf{B})$	PBetter	0.4461	0.5386	0.4721	0.3870	0.5231
	PWorse	0.1130	0.0006	0.0001	0.1042	0.0141
$r(\mathbf{B})$	PBetter	0.4110	0.5571	0.4805	0.2972	0.5415
	PWorse	0.5887	0.4429	0.5195	0.7019	0.4585
κ_B	PBetter	0.9990	0.9991	0.9990	0.9929	0.9993
	PWorse	0.0010	0.0009	0.0010	0.0071	0.0007

$\mathcal{O}(\mathbf{B})$, $\ell_1(\mathbf{B})$, $r(\mathbf{B})$, κ_B, and $\theta(\mathbf{B})$ correspond to the quality measures described in Section 3. PBetter, the probability with which LLL, DEEP, SLLL, VLLL, and PLLL improve the quality indices of the problems; PWorse, the probability with which LLL, DEEP, SLLL, VLLL, and PLLL worsen the quality indices of the problems. Otherwise, these methods do not change the quality indices of the problems.

4.3.1 Orthogonality defect

For each of the $10,000$ simulated examples, we have computed the corresponding orthogonality defects from LLL, DEEP, SLLL, VLLL, and PLLL, which are collectively denoted by $\mathcal{O}^{\text{LLL}}$, $\mathcal{O}^{\text{DEEP}}$, $\mathcal{O}^{\text{SLLL}}$, $\mathcal{O}^{\text{VLLL}}$, and $\mathcal{O}^{\text{PLLL}}$, respectively. Together with the original problems, we have plotted the cumulative probability functions (cdf) of the orthogonality defects in Figure 5. Among the five basis reduction methods under study, PLLL performs the best and VLLL the worst in orthogonality defects. SLLL is consistently better than LLL (compare the red and green lines) but worse than DEEP in general. It is surprising to see from Figure 5 that none of the reduction methods can produce a smaller orthogonality defect than the original problems overwhelmingly. It is clear from row $\mathcal{O}(\mathbf{B})$ of Table 1 that, even in the best case, we still see the probability of 0.366 with which the original problems have a smaller orthogonality defect than PLLL. As will be clear, in other parts of this section, the original problems can be significantly improved. From this point of view, the orthogonality defect alone does not necessarily reflect the quality of a reduction method correctly. One should exercise great care to interpret the orthogonality defect when using it to evaluate the performance of a reduction method. It is also interesting to see that the popular LLL algorithm only shows a chance of 0.502 to produce a smaller orthogonality defect (compare $\mathcal{O}(\mathbf{B})$ of Table 1 under LLL).

Since a cdf plot does not reveal a direct comparison of each simulated example between any two methods of reduction, we have computed the differences of orthogonality defects for the $10,000$ examples. Illustrated in Figure 6 are the probability density functions (pdf) of the differences of orthogonality defects of PLLL relative to LLL, DEEP, SLLL, and VLLL. The statistics by comparing PLLL with the other four basis reduction methods are listed in Table 2. Both Figure 6 and Table 2 (row $\mathcal{O}(\mathbf{B})$) have clearly shown the outstanding performance of PLLL over LLL, DEEP, SLLL, and VLLL with respect to the quality measure of orthogonality defect. Although DEEP might be thought to produce the best results, it could

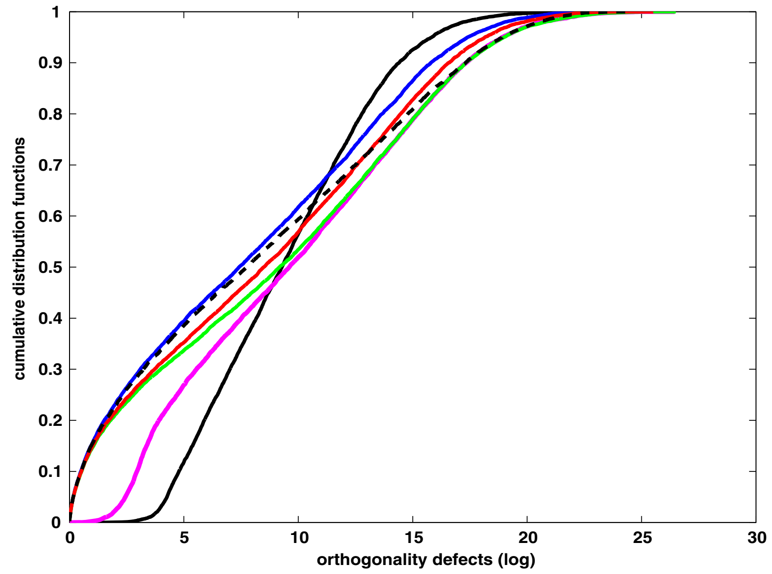

Figure 5 Cumulative probability functions of orthogonality defects (in logarithm) from 10, 000 random examples. PROB, black solid line; LLL, green line; DEEP, black dashed line; SLLL, red line; VLLL, pink line; and PLLL, blue line.

only win PLLL with a small probability of 0.125 on the orthogonality defect.

Because both LLL and DEEP are popular, we have further computed the differences of orthogonality defects of LLL relative to DEEP, SLLL, and VLLL, namely, $(\log \mathcal{O}^{\text{LLL}} - \log \mathcal{O}^{\text{DEEP}})$, $(\log \mathcal{O}^{\text{LLL}} - \log \mathcal{O}^{\text{SLLL}})$, and $(\log \mathcal{O}^{\text{LLL}} - \log \mathcal{O}^{\text{VLLL}})$, which are plotted in the pdf form in Figure 7 and summarized statistically in Table 3. It is clear from panel C of Figure 7 that LLL performs significantly better than VLLL. This might indicate that the fixed

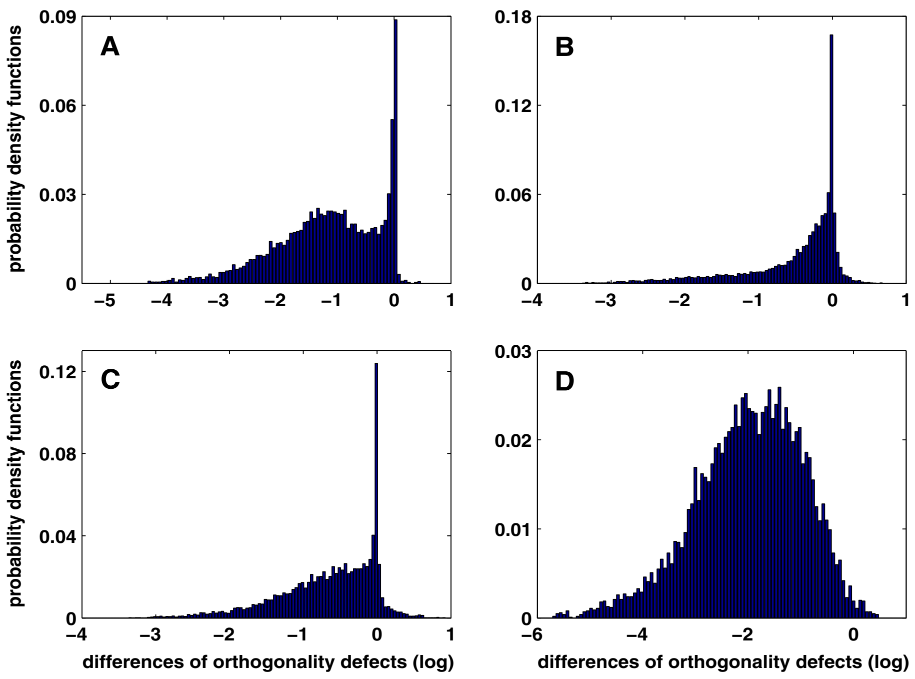

Figure 6 Probability density functions of differences of orthogonality defects (in logarithm). This figure is to compare PLLL with the other four basis reduction methods. panel **A**, $(\log \mathcal{O}^{\text{PLLL}} - \log \mathcal{O}^{\text{LLL}})$ for PLLL relative to LLL; panel **B**, $(\log \mathcal{O}^{\text{PLLL}} - \log \mathcal{O}^{\text{DEEP}})$ for PLLL relative to DEEP; panel **C**, $(\log \mathcal{O}^{\text{PLLL}} - \log \mathcal{O}^{\text{SLLL}})$ for PLLL relative to SLLL; and panel **D**, $(\log \mathcal{O}^{\text{PLLL}} - \log \mathcal{O}^{\text{VLLL}})$ for PLLL relative to VLLL. Negative difference values mean better results for PLLL.

Table 2 Probabilities estimated by comparing PLLL with LLL, DEEP, SLLL, and VLLL

Measures	Methods	LLL	DEEP	SLLL	VLLL
$\mathcal{O}(\mathbf{B})$	PBetter	0.908	0.785	0.847	0.991
	PWorse	0.003	0.125	0.088	0.009
$\theta(\mathbf{B})$	PBetter	0.731	0.547	0.666	0.830
	PWorse	0.192	0.314	0.238	0.169
$\ell_1(\mathbf{B})$	PBetter	0.305	0.024	0.183	0.562
	PWorse	0.019	0.076	0.026	0.015
$r(\mathbf{B})$	PBetter	0.774	0.361	0.584	0.882
	PWorse	0.134	0.512	0.316	0.118
κ_B	PBetter	0.845	0.808	0.796	0.951
	PWorse	0.097	0.102	0.139	0.049

$\mathcal{O}(\mathbf{B})$, $\theta(\mathbf{B})$, $\ell_1(\mathbf{B})$, $r(\mathbf{B})$, and κ_B correspond to the quality measures described in Section 3. PBetter, the probability with which PLLL performs better; PWorse, the probability with which LLL, DEEP, SLLL, and VLLL perform better than PLLL. Otherwise, PLLL produces the same results as the other reduction methods.

complexity of VLLL may finish the reduction too quickly. However, both DEEP and SLLL are surely much better than LLL, as can be seen from panels A and B of Figure 7, and the values of PWorse in row $\mathcal{O}(\mathbf{B})$ of Table 3. This should indicate that deep insertions and the sorted QR ordering help make the reduced vectors more orthogonal than the original LLL algorithm. Nevertheless, DEEP is better than SLLL on this quality measure, as can be seen from panel D of Figure 7, which displays the pdf function of the orthogonality defects from DEEP relative to those from SLLL.

4.3.2 Minimum angle $\theta(\mathbf{B})$ among the reduced vectors

Orthogonality defect has been defined and used to quantitatively measure the extent of orthogonality of a reduced lattice basis. It can take on a value from the idealized unity to infinity, which corresponds to a completely orthogonal basis with a full rank and a rank-defect sub-basis, respectively. An obvious disadvantage of orthogonality defect is that given a value of $\mathcal{O}(\mathbf{B})$, we do not have any idea about how orthogonal the reduced basis looks like. As a result, we proposed an alternative quantity $\theta(\mathbf{B})$ to measure the extent of orthogonality of a reduced basis. As the minimum angle defined in $[0^o, 90^o]$ among all the mutual vectors of a reduced basis, $\theta(\mathbf{B})$ is intuitively appealing, since we can immediately tell roughly how orthogonal the reduced basis is. Actually, the two extreme values of $\theta(\mathbf{B})$, namely, 0^o and 90^o, correspond to a degenerate reduced basis and a completely orthogonal basis, respectively. Unlike the other five quality measures, the bigger the minimum angle, the better the corresponding reduction method is with respect to $\theta(\mathbf{B})$. In this section, we

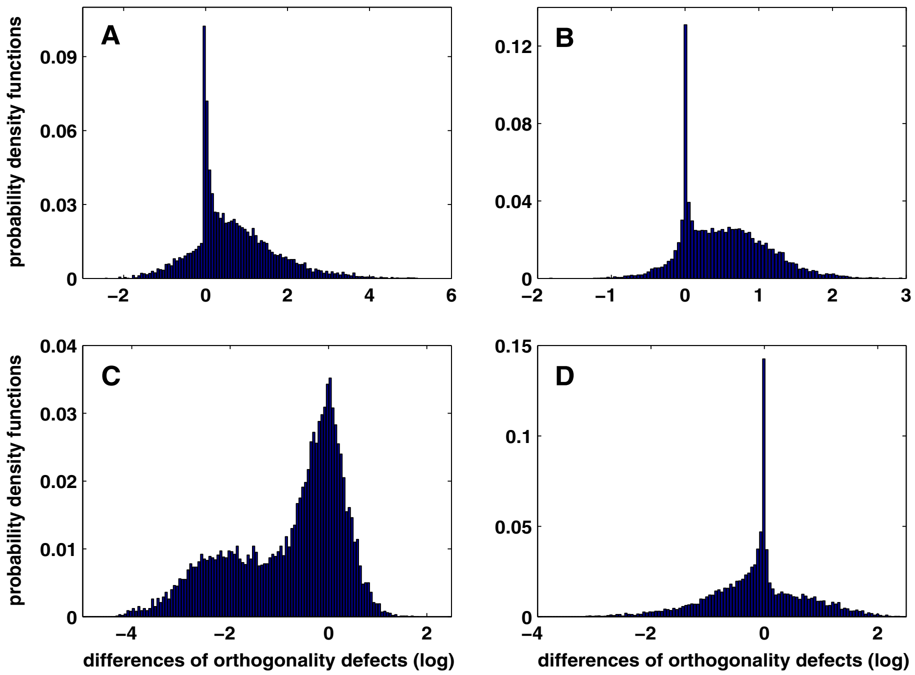

Figure 7 Probability density functions of differences of orthogonality defects (in logarithm). This figure is to compare LLL with DEEP, SLLL, and VLLL in panels **A** to **C**, and DEEP with SLLL in panel **D**. panel **A**, $(\log \mathcal{O}^{\text{LLL}} - \log \mathcal{O}^{\text{DEEP}})$ for LLL relative to DEEP; panel **B**, $(\log \mathcal{O}^{\text{LLL}} - \log \mathcal{O}^{\text{SLLL}})$ for LLL relative to SLLL; panel **C**, $(\log \mathcal{O}^{\text{LLL}} - \log \mathcal{O}^{\text{VLLL}})$ for LLL relative to VLLL; and panel **D**, $(\log \mathcal{O}^{\text{DEEP}} - \log \mathcal{O}^{\text{SLLL}})$ for DEEP relative to SLLL. Negative difference values mean better results for LLL in panels A, B and C, and better results for DEEP in panel D.

Table 3 Probabilities estimated by comparing LLL with DEEP, SLLL, and VLLL

Measures	Methods	DEEP	SLLL	VLLL
$\mathcal{O}(\mathbf{B})$	PBetter	0.202	0.152	0.725
	PWorse	0.739	0.771	0.275
$\theta(\mathbf{B})$	PBetter	0.275	0.368	0.558
	PWorse	0.648	0.527	0.347
κ_B	PBetter	0.428	0.364	0.790
	PWorse	0.514	0.559	0.210

$\mathcal{O}(\mathbf{B})$, $\theta(\mathbf{B})$ and κ_B are the same as in Table 1. PBetter, the probability with which LLL performs better; PWorse, the probability with which DEEP, SLLL, and VLLL perform better than LLL. Otherwise, LLL produces the same results as the other reduction methods.

shall use the 10, 000 random examples to investigate the effectiveness of $\theta(\mathbf{B})$ as an alternative quality measure of orthogonality defect.

As in the case of orthogonality defect, let us denote the 10, 000 minimum angles $\theta(\mathbf{B})$ from each of LLL, DEEP, SLLL, VLLL and PLLL by $\boldsymbol{\theta}^{\mathrm{LLL}}$, $\boldsymbol{\theta}^{\mathrm{DEEP}}$, $\boldsymbol{\theta}^{\mathrm{SLLL}}$, $\boldsymbol{\theta}^{\mathrm{VLLL}}$ and $\boldsymbol{\theta}^{\mathrm{PLLL}}$, respectively, with those of the original problems by $\boldsymbol{\theta}^{\mathrm{PROB}}$. The cdf curves of these minimum angles are shown in Figure 8, and the probabilities estimated by comparing $\boldsymbol{\theta}^{\mathrm{LLL}}$, $\boldsymbol{\theta}^{\mathrm{DEEP}}$, $\boldsymbol{\theta}^{\mathrm{SLLL}}$, $\boldsymbol{\theta}^{\mathrm{VLLL}}$, and $\boldsymbol{\theta}^{\mathrm{PLLL}}$ with $\boldsymbol{\theta}^{\mathrm{PROB}}$ are listed in row $\theta(\mathbf{B})$ of Table 1. We may observe from Figure 8 that (a) VLLL tends to output small minimum angles with a bigger probability than LLL, DEEP, SLLL, and PLLL, indicating that VLLL could terminate with a poorly orthogonal reduced basis with a significant probability; (b) all the other four methods of basis reduction are generally satisfactory, but the order of increasing performance to produce a bigger minimum angle is immediately visible, ranging from the least effective LLL, SLLL, and then DEEP to the most effective PLLL. In other words, PLLL is most successful in guaranteeing a big minimum angle with a biggest chance. It is most robust in avoiding a reduced basis of poor orthogonality, with an almost zero probability of 0.001 to result in a minimum angle smaller than 45^o; and (c) the problems themselves can have a bigger probability to have a minimum angle over $[51.8^o, 67.9^o]$, depending on which of LLL, DEEP, SLLL and PLLL is used to compare with PROB. A closer look at row $\theta(\mathbf{B})$ of Table 1 has shown a surprising phenomenon that none of the reduction methods can have a probability of more than 0.5 to improve the original problems with respect to this quality measure. However, the problems will be demonstrated to be significantly improved in terms of $\ell_1(\mathbf{B})$ and κ_B in this section. This should indicate that the minimum angle alone, as in the case of orthogonality defect, is not sufficient to represent the quality of a reduced basis or a reduction method, unless the lengths of the original basis are already short. Nevertheless, this quality measure also consistently indicates that both PLLL and DEEP are the best reduction methods under study.

To carry out a direct comparison of each example among LLL, DEEP, SLLL, VLLL, and PLLL, we have first computed the differences $(\boldsymbol{\theta}^{\mathrm{PLLL}} - \boldsymbol{\theta}^{\mathrm{LLL}})$, $(\boldsymbol{\theta}^{\mathrm{PLLL}} - \boldsymbol{\theta}^{\mathrm{VLLL}})$, $(\boldsymbol{\theta}^{\mathrm{PLLL}} - \boldsymbol{\theta}^{\mathrm{SLLL}})$, and $(\boldsymbol{\theta}^{\mathrm{PLLL}} - \boldsymbol{\theta}^{\mathrm{DEEP}})$, which are then depicted in Figure 9 in the form of pdf histograms and statistically summarized in row $\boldsymbol{\theta}(\mathbf{B})$ of Table 2. It is obvious from both Figure 9 and Table 2 that PLLL performs

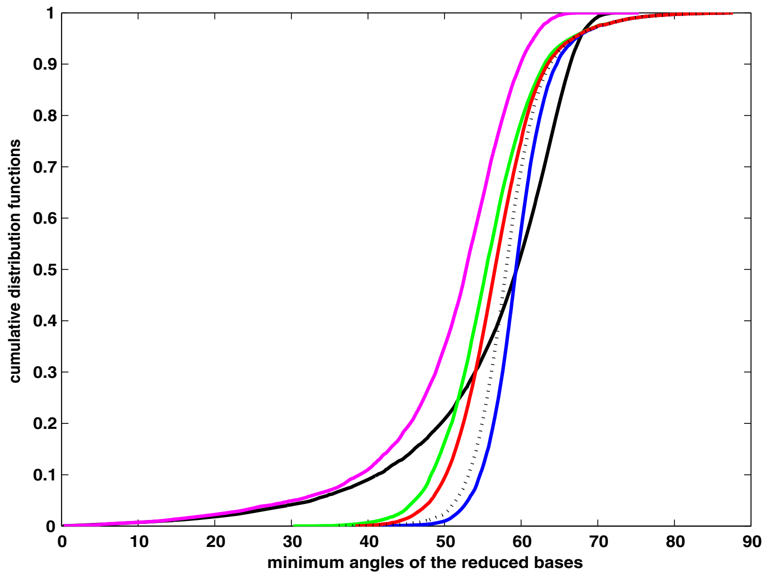

Figure 8 Cumulative probability functions of minimum angles (in degrees) from 10, 000 random examples. PROB, the solid black line; LLL, the green line; DEEP, the dotted black line; SLLL, the red line; VLLL, the pink line; and PLLL, the blue line.

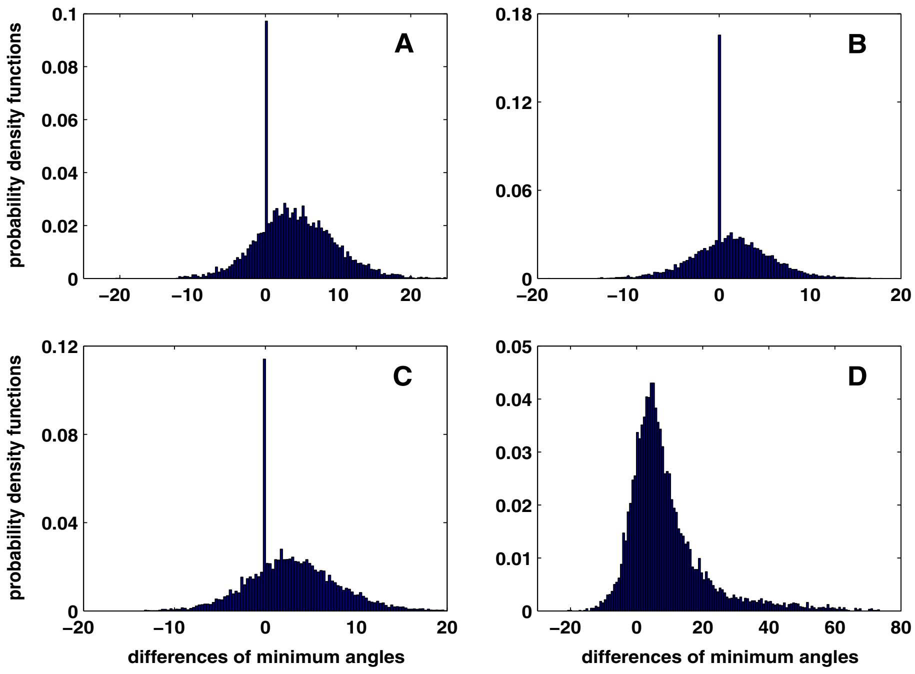

Figure 9 Histograms of probability density functions of minimum angles. This figure is to compare PLLL with LLL, DEEP, SLLL, and VLLL. Positive values indicate better results for PLLL. panel **A**, ($\theta^{\text{PLLL}} - \theta^{\text{LLL}}$); panel **B**, ($\theta^{\text{PLLL}} - \theta^{\text{DEEP}}$); panel **C**, ($\theta^{\text{PLLL}} - \theta^{\text{SLLL}}$); and panel **D**, ($\theta^{\text{PLLL}} - \theta^{\text{VLLL}}$).

significantly better than any of the other four reduction methods in producing a bigger minimum angle of the reduced basis. Although we may infer from row $\boldsymbol{\theta}(\mathbf{B})$ of Table 2 that the next best method of reduction is DEEP, followed by SLLL and LLL, with VLLL in the bottom of performance with respect to this quality index, we decide to show the direct evidence by comparing the popular LLL algorithm with the other three methods, namely, VLLL, SLLL, and DEEP. More precisely, we have depicted the pdf histograms of the quantities ($\boldsymbol{\theta}^{\text{LLL}} - \boldsymbol{\theta}^{\text{VLLL}}$), ($\boldsymbol{\theta}^{\text{LLL}} - \boldsymbol{\theta}^{\text{SLLL}}$) and ($\boldsymbol{\theta}^{\text{LLL}} - \boldsymbol{\theta}^{\text{DEEP}}$) in Figure 10 and summarized them statistically in Table 3. LLL is clearly better than VLLL but worse than SLLL and DEEP. Panel D of Figure 10 also shows that DEEP performs better than SLLL, more precisely, with a probability of 0.569 to 0.330 for DEEP.

4.3.3 Hermite factor γ_B

The Hermite factor is an important quality measure of a reduction method and theoretically reflects the upper bound of the shortest reduced vector through the following inequality (see e.g., [3]):

$$\|\mathbf{b}_1\| \leq \beta^{(m-1)/4}[\det(\mathcal{L})]^{1/2m}, \tag{24}$$

where β is equal to $4/(4\delta - 1)$. $\beta^{1/4}$ can be rewritten as γ_B, which is then referred to as the Hermite factor in the literature [7,34,45]. In the case of LLL, $\beta = 2$ and $\gamma_B = 1.189$. When δ approaches to one, $\beta \approx 4/3$ and $\gamma_B \approx 1.075$ [7,45]. Experimental studies [7,45] have shown that γ_B can be practically much smaller than theoretically expected. γ_B can be as small as 1.02 in the case of LLL and 1.01 in the case of DEEP (compare Table one of Gama and Nguyen [45]). Based on the experimental pdf of the Gram-Schmidt coefficients μ_{ij} and the assumption of a Weibull distribution to probabilistically describe the Lovasz condition, Schneider et al. [34] obtained the expectation value of 1.019 for γ_B after the LLL reduction, which is slightly smaller than 1.0219 obtained experimentally by Gama and Nguyen [45].

Based on the experiments on the 10, 000 random examples, we obtain the Hermite factors γ_B after the reductions by LLL, PLLL, VLLL, SLLL, and DEEP. Because the experiments by Nguyen and Stehlé [7] have shown the convergence of logarithm of γ_B to a certain constant, instead of computing γ_B, we have directly computed $\log(\gamma_B)$, which is denoted by η_B and given as follows:

$$\eta_B = \log(\gamma_B) = \frac{1}{m} \log \frac{\|\mathbf{b}_1\|}{[\det(\mathcal{L})]^{1/(2m)}}. \tag{25}$$

All the η_B values after the reductions are shown in Figure 11, together with those of the original random problems. Indeed, η_B from any of the reductions (LLL, PLLL, VLLL, SLLL, and DEEP) tend to stabilize after rank 15 and converge to some constant. The statistics of γ_B ($m \geq 15$) for each of the five reductions are listed in Table 4. It is clear from Table 4 that both PLLL and

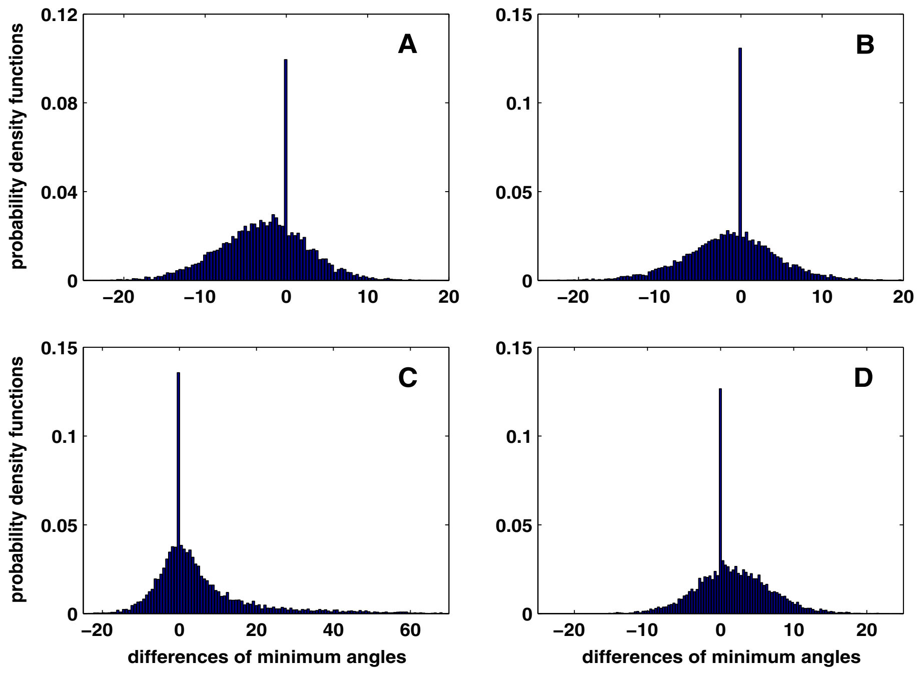

Figure 10 Histograms of probability density functions of minimum angles. This figure is to compare LLL with DEEP, SLLL, and VLLL. Also shown in panel **D** of this figure is the pdf histogram of DEEP relative to SLLL. Positive values indicate better results for LLL in panels **A**, **B**, and **C**, and better result for DEEP in panel **D**. Panel **A**, $(\theta^{\text{LLL}} - \theta^{\text{DEEP}})$; panel **B**, $(\theta^{\text{LLL}} - \theta^{\text{SLLL}})$; panel **C**, $(\theta^{\text{LLL}} - \theta^{\text{VLLL}})$; and panel **D**, $(\theta^{\text{DEEP}} - \theta^{\text{SLLL}})$.

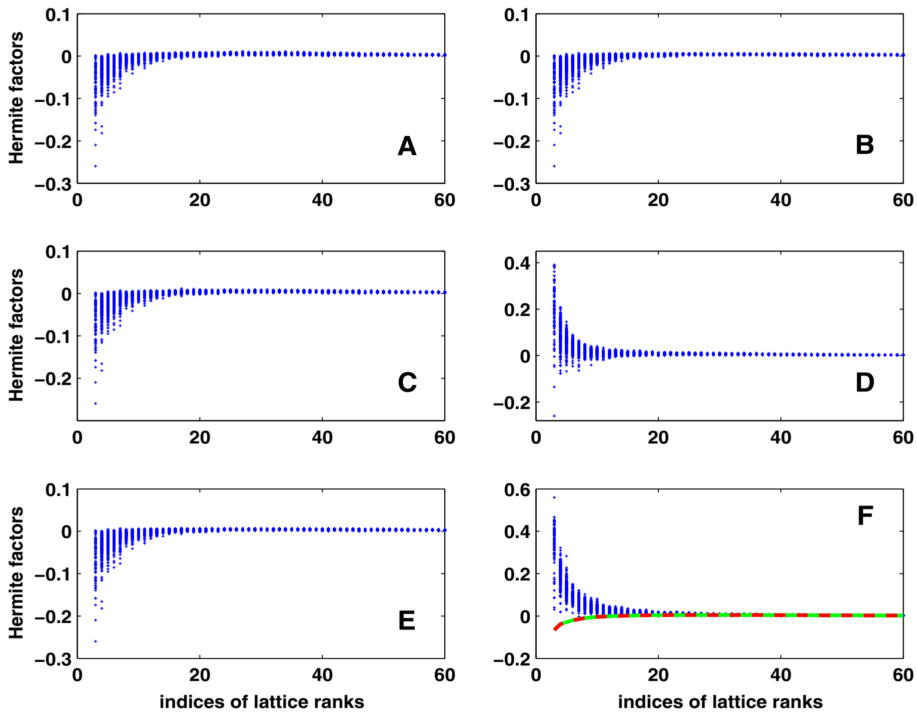

Figure 11 Hermite factors of $10,000$ random examples (in logarithm) after reductions. Panel **A**, LLL; panel **B**, DEEP; panel **C**, SLLL; panel **D**, VLLL; and panel **E**, PLLL. Shown in panel **F** are the Hermite factors of the random problems (in logarithm with symbol $+$) and the mean values of η_B for each rank of lattice, with the green line for PLLL and the dashed red line for DEEP.

Table 4 Statistics of Hermite factors with $m \geq 15$ from five reduction methods LLL, DEEP, SLLL, VLLL, and PLLL

Methods	LLL	DEEP	SLLL	VLLL	PLLL
Mean	1.0100	1.0090	1.0096	1.0122	1.0090
Max	1.0271	1.0178	1.0277	1.0513	1.0178
Min	0.9858	0.9858	0.9858	0.9931	0.9858

Mean, mean values of γ_B; *Max*, maximum values of γ_B; *Min*, minimum values of γ_B.

DEEP perform most excellently, followed by SLLL and LLL. Actually, the curves of mean values from PLLL (the green line) and DEEP (the red-dotted line) in panel F of Figure 11 reveal that these two methods essentially produce the same results of γ_B on average. The relative error of the mean γ_B of PLLL to that of DEEP is negligibly equal to 0.007%. The mean value of γ_B after DEEP is consistent with the report of 1.01 by Gama and Nguyen [45], though negligibly smaller by 0.001. VLLL results in the biggest mean value of the Hermite factors. Even worse, although all the other four methods have produced relatively large negative values of η_B for a small rank m, VLLL tends to maintain all the same large positive values of η_B as the original problems (compare panels D and F of Figure 11).

Our experiments have shown that a smaller mean value of γ_B is not impossible for the LLL algorithm, which might be related to types and randomness of lattice bases. On the other hand, the η_B values of the original problems also converge to a small constant, irrelevant to any differences in the problems themselves, as can be seen in panel F of Figure 11. This might indicate that a quality measure with a power function could only reveal a rough aspect of quality of a reduction method but hide all detailed important features of a problem and/or a reduction method.

4.3.4 Length $\ell_1(\mathbf{B})$ of the shortest reduced vector

Reduction is to make the reduced basis vectors of lattice as short as possible, but the effect of reduction is dependent on methods of reduction and their control parameters. Since solving the shortest vector problem of a lattice is conjectured to be NP-hard, we will focus on the shortest reduced vectors obtained after applying the five methods of reduction, i.e., LLL, DEEP, SLLL, VLLL, and PLLL, based on the 10, 000 simulated random lattices. More precisely, let us denote the 10, 000 lengths of $\ell_1(\mathbf{B})$ after the reductions of LLL, DEEP, SLLL, VLLL, and PLLL by the vectors ℓ_1^{LLL}, ℓ_1^{DEEP}, ℓ_1^{SLLL}, ℓ_1^{VLLL}, and ℓ_1^{PLLL}, respectively. For a clear visualization, we have plotted the cdf functions of $(\ell_1^{\text{PLLL}} - \ell_1^{\text{LLL}})$, $(\ell_1^{\text{PLLL}} - \ell_1^{\text{DEEP}})$, $(\ell_1^{\text{PLLL}} - \ell_1^{\text{SLLL}})$, and $(\ell_1^{\text{PLLL}} - \ell_1^{\text{VLLL}})$ in Figure 12. It is clear from Figure 12 that DEEP performs the best in outputting the shortest reduced vectors, which are theoretically expected, since DEEP employs a strongest swapping condition to reinforce reducing the lengths of the reduced vectors [35]. Panel B of Figure 12 has shown that PLLL is almost as good as DEEP in producing the shortest reduced vectors.

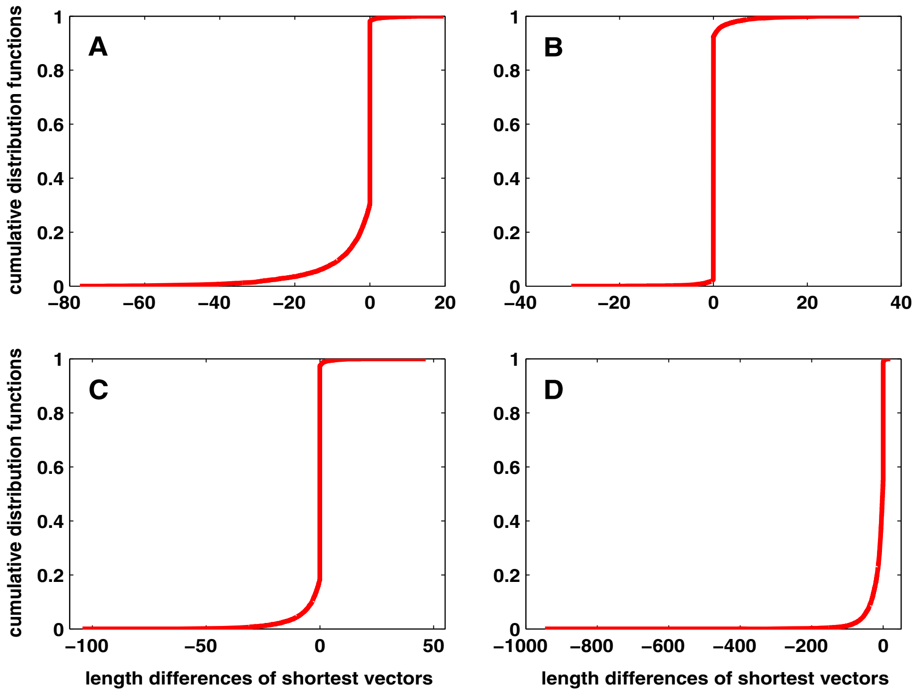

Figure 12 Curves of cumulative distribution functions of length differences of shortest reduced vectors. This figure is to compare PLLL with LLL, DEEP, SLLL, and VLLL. panel **A**, $(\ell_1^{\text{PLLL}} - \ell_1^{\text{LLL}})$; panel **B**, $(\ell_1^{\text{PLLL}} - \ell_1^{\text{DEEP}})$; panel **C**, $(\ell_1^{\text{PLLL}} - \ell_1^{\text{SLLL}})$; and panel **D**, $(\ell_1^{\text{PLLL}} - \ell_1^{\text{VLLL}})$.

Table 5 Probabilities estimated by comparing DEEP with LLL, SLLL, and VLLL

Measures	Methods	LLL	SLLL	VLLL
$\ell_1(\mathbf{B})$	PBetter	0.319	0.201	0.576
	PWorse	0.005	0.006	0.001
$r(\mathbf{B})$	PBetter	0.802	0.668	0.915
	PWorse	0.104	0.223	0.085

PBetter, the probability with which DEEP performs better; PWorse, the probability with which LLL, SLLL, and VLLL perform better. Otherwise, DEEP produces the same results as the other reduction methods.

Looking at the two numbers of probability in column DEEP of row $\ell_1(\mathbf{B})$ of Table 2, we can see that both PLLL and DEEP produce exactly the same results of $\ell_1(\mathbf{B})$ with a probability of 0.9, or equivalently, with 90% of the examples. In the remaining 10%, each of PLLL and DEEP performs better than the other with 2.4% and 7.6% of the examples, respectively. Keeping in mind that DEEP has a super-exponential complexity [45] in the worst case, PLLL is remarkable to find a shortest possible $\ell_1(\mathbf{B})$ at a fixed complexity. We can also see from Figure 12 that although both LLL and SLLL cannot compete with PLLL (see panels A and C and Table 2), they are much better than VLLL (see panel D).

Since DEEP has been known for its great ability to output the nearly shortest reduced vector as a direct consequence of deep insertions [35], we have made a further comparison of DEEP with LLL, VLLL, and SLLL. Based on the above results of $\ell_1(\mathbf{B})$, we have computed and listed the probabilities of DEEP in comparison with LLL, VLLL, and SLLL in Table 5 and shown the cdf curves of $(\ell_1^{\text{DEEP}} - \ell_1^{\text{LLL}})$, $(\ell_1^{\text{DEEP}} - \ell_1^{\text{SLLL}})$, and $(\ell_1^{\text{DEEP}} - \ell_1^{\text{VLLL}})$ in Figure 13. Indeed, DEEP outperforms any of these three methods of reduction, as also obviously shown in column $\ell_1(\mathbf{B})$ of Table 5 and Figure 13. A significant, positive impact of the sorted QR ordering on $\ell_1(\mathbf{B})$ can also be inferred by comparing the columns LLL and SLLL of Table 5 and panels A and C of both Figure 12 and Figure 13.

4.3.5 Length ratio r(**B**)

Success of reduction is supposed to shorten the lengths of the reduced basis vectors. Theoretical results of reduction guarantee the following inequality of bound

$$\|\mathbf{b}_k\|^2/\lambda^2(\mathcal{L}) \leq \alpha^{m-1} \tag{26}$$

for $k \geq 1$ [35]. If we replace $\|\mathbf{b}_k\|$ by $\max\{\|\mathbf{b}_2\|, \ldots, \|\mathbf{b}_m\|\}$ and α by $\alpha_r = \alpha^{1/2}$ in (26), then we can rewrite (26) as follows:

$$\frac{\max\{\|\mathbf{b}_2\|, \ldots, \|\mathbf{b}_m\|\}}{\lambda(\mathcal{L})} \leq \alpha_r^{m-1}, \tag{27}$$

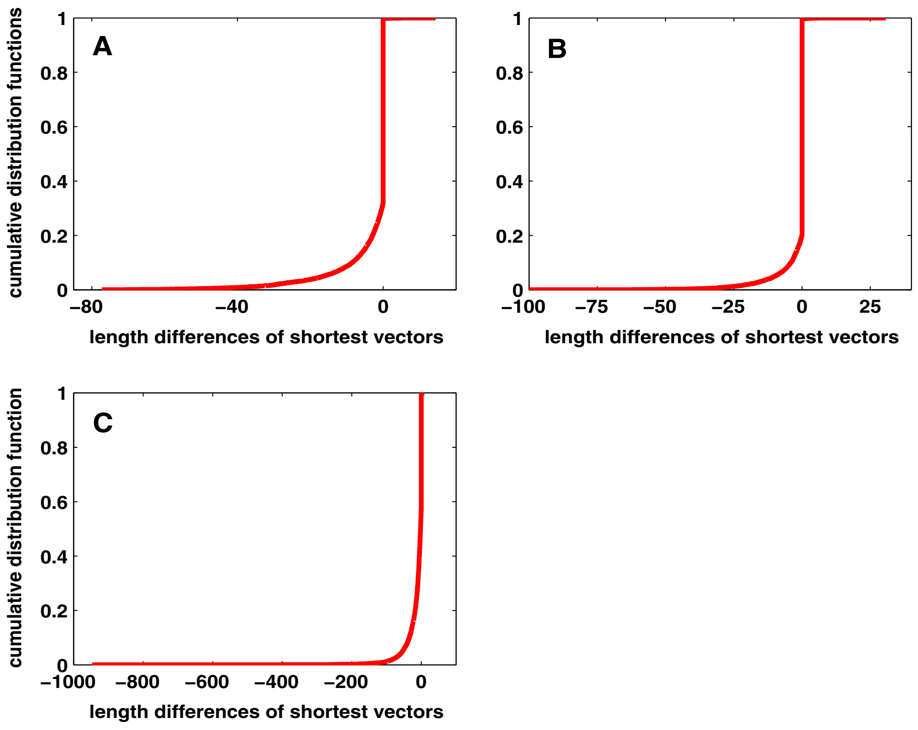

Figure 13 Curves of cumulative distribution functions of length differences of shortest reduced vectors. This figure is to compare DEEP with LLL, SLLL, and VLLL. panel **A**, $(\ell_1^{\text{DEEP}} - \ell_1^{\text{LLL}})$; panel **B**, $(\ell_1^{\text{DEEP}} - \ell_1^{\text{SLLL}})$; and panel **C**, $(\ell_1^{\text{DEEP}} - \ell_1^{\text{VLLL}})$.

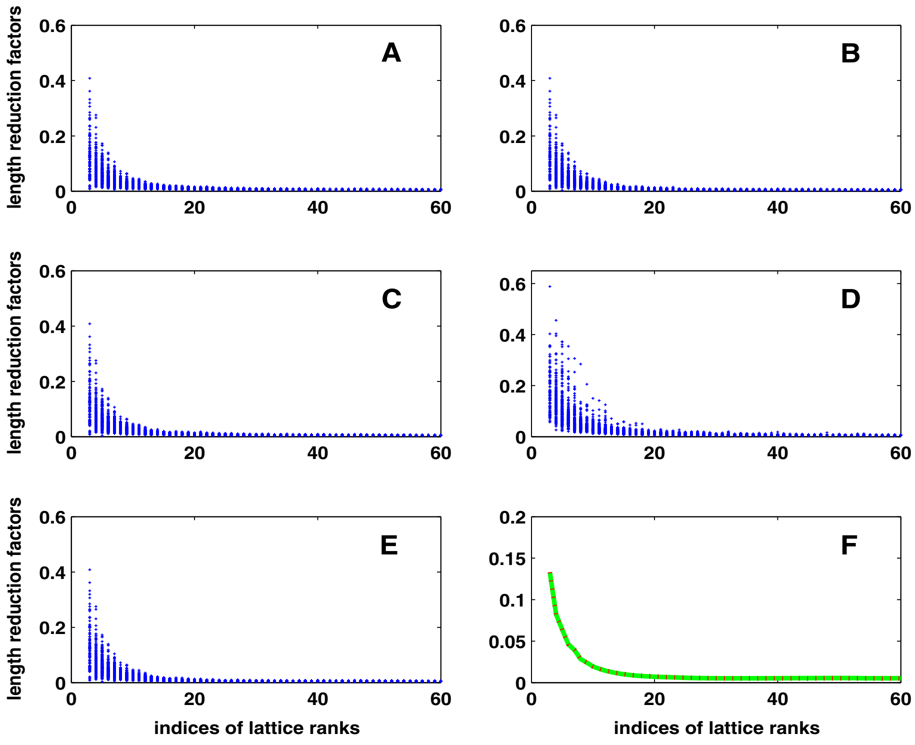

Figure 14 Length reduction factors of 10, 000 random examples (in logarithm), or equivalently, values of η_r, after reductions. panel **A**, LLL; panel **B**, DEEP; panel **C**, SLLL; panel **D**, VLLL; and panel **E**, PLLL. Shown in panel **F** are the mean values of η_r for each rank of lattices, with the green line for PLLL and the red dotted line for DEEP.

where α_r will be referred to as the length reduction factor in the remainder of this paper. Remembering that $\lambda(\mathcal{L})$ cannot be directly attainable as a result of any reduction algorithm, we will replace it with $\|\mathbf{b}_1\|$. Thus, the left-hand side of (27) exactly becomes the length ratio as defined in (12). In the similar way to defining η_B in connection with the Hermite factor, we may define

$$\eta_r = \log \alpha_r = \frac{1}{m} \log r(\mathbf{B}), \tag{28}$$

which can be directly estimated from the length ratio $r(\mathbf{B})$.

After reduction, we can obtain the 10, 000 length ratios of $r(\mathbf{B})$ for each of LLL, DEEP, SLLL, VLLL, and PLLL, which are collectively denoted by the vectors $\mathbf{r}^{\text{LLL}}$, $\mathbf{r}^{\text{DEEP}}$, $\mathbf{r}^{\text{SLLL}}$, $\mathbf{r}^{\text{VLLL}}$, and $\mathbf{r}^{\text{PLLL}}$. The results of η_r are shown in Figure 14 for each rank of lattices and the statistics of the length reduction factors α_r are listed in Table 6. As in the case of η_B for the Hermite factor γ_B, Figure 14 has clearly shown that the η_r values converge for all the five reduction methods under study. By looking at panels A, B, C, and E of Figure 14, one may conclude that except for VLLL, the other four methods of reduction, i.e., LLL, DEEP, SLLL, and PLLL work equally well with respect to the length ratio $r(\mathbf{B})$. A closer examination of Table 6 reveals that (a) DEEP performs the best. This should be theoretically expected, since DEEP was designed to further reduce the lengths of a reduced basis [35]; (b) the second best method of reduction is PLLL, which is almost as good as DEEP, followed by SLLL and LLL; and (c) VLLL has the poorest performance.

Let us now come to a direct comparison of the length ratios $r(\mathbf{B})$ for each simulated random example among the five reduction methods. Theoretically, we expect that a good reduction method should result in a small value of $r(\mathbf{B})$, unless the basis $\mathbf{B}$ of a lattice is already orthogonal. Shown in Figure 15 are the cdf curves of $(\log \mathbf{r}^{\text{PLLL}} - \log \mathbf{r}^{\text{LLL}})$, $(\log \mathbf{r}^{\text{PLLL}} - \log \mathbf{r}^{\text{DEEP}})$, $(\log \mathbf{r}^{\text{PLLL}} - \log \mathbf{r}^{\text{SLLL}})$, and $(\log \mathbf{r}^{\text{PLLL}} - \log \mathbf{r}^{\text{VLLL}})$. The probabilities computed from these quantities are summarized in row $r(\mathbf{B})$ of Table 2. It is obvious from Figure 15 that PLLL outperforms LLL, SLLL, and VLLL with respect to this quality measure, as can also be confirmed by looking at row $r(\mathbf{B})$ of Table 2. The last column of row $r(\mathbf{B})$ in Table 2 shows

Table 6 Statistics of length reduction factors with $m \geq 15$ from reduction methods LLL, DEEP, SLLL, VLLL, and PLLL

Methods	LLL	DEEP	SLLL	VLLL	PLLL
Mean	1.0164	1.0133	1.0148	1.0182	1.0136
Max	1.0533	1.0435	1.0456	1.1456	1.0435
Min	1.0082	1.0067	1.0057	1.0070	1.0064

Mean, mean values of α_r; *Max*, maximum values of α_r; *Min*, minimum values of α_r.

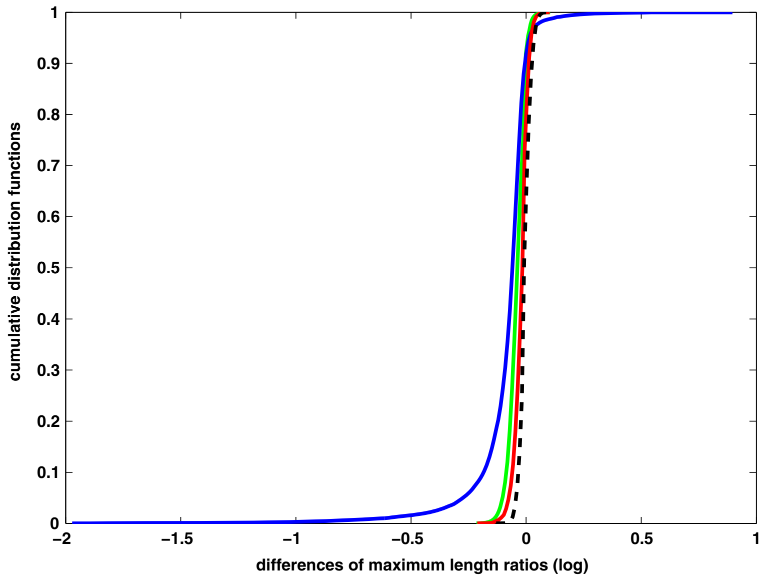

Figure 15 Curves of cumulative distribution functions of length ratios. This figure is to compare PLLL with LLL, DEEP, SLLL, and VLLL. The green line indicates ($\log \mathbf{r}^{\mathrm{PLLL}} - \log \mathbf{r}^{\mathrm{LLL}}$); the dashed black line indicates ($\log \mathbf{r}^{\mathrm{PLLL}} - \log \mathbf{r}^{\mathrm{DEEP}}$); the red line indicates ($\log \mathbf{r}^{\mathrm{PLLL}} - \log \mathbf{r}^{\mathrm{SLLL}}$); and the blue line indicates ($\log \mathbf{r}^{\mathrm{PLLL}} - \log \mathbf{r}^{\mathrm{VLLL}}$).

that DEEP performs significantly better than PLLL with a probability of 0.512 but worse only with a probability of 0.361 in terms of $r(\mathbf{B})$. To further compare DEEP with LLL, SLLL, and VLLL, we have computed the quantities, i.e., ($\log \mathbf{r}^{\mathrm{DEEP}} - \log \mathbf{r}^{\mathrm{LLL}}$), ($\log \mathbf{r}^{\mathrm{DEEP}} - \log \mathbf{r}^{\mathrm{SLLL}}$), and ($\log \mathbf{r}^{\mathrm{DEEP}} - \log \mathbf{r}^{\mathrm{VLLL}}$), shown the cdf curves of these quantities in Figure 16 and summarized their performance probabilities in row $r(\mathbf{B})$ of Table 5. Both Figure 16 and Table 5 clearly show the superior performance of DEEP over LLL, SLLL, and VLLL.

4.3.6 Condition number κ_B

Both the shortest vector and closest point problems of a lattice defined by a basis $\mathbf{B}$ are associated with its positive definite quadratic form. The shape of the corresponding searching ellipsoid is strongly determined by

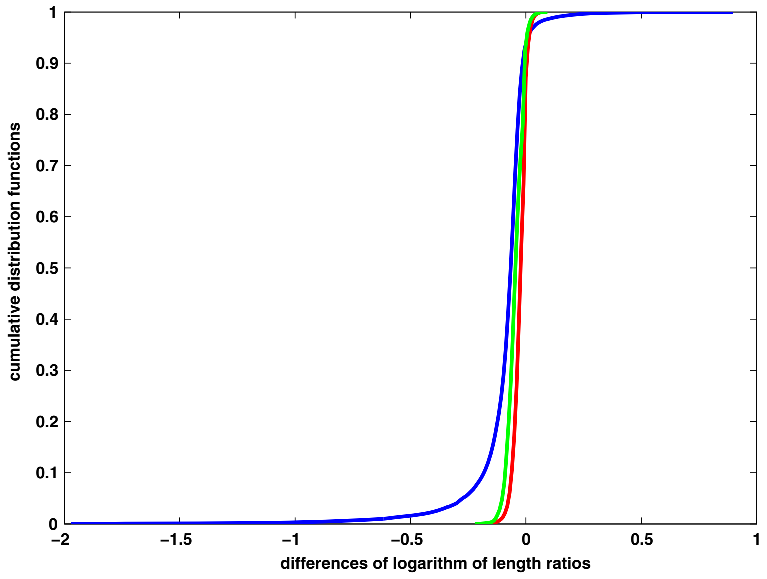

Figure 16 Curves of cumulative distribution functions of length ratios. This figure is to compare DEEP with LLL, SLLL, and VLLL. The green line indicates ($\log \mathbf{r}^{\mathrm{DEEP}} - \log \mathbf{r}^{\mathrm{LLL}}$); the red line indicates ($\log \mathbf{r}^{\mathrm{DEEP}} - \log \mathbf{r}^{\mathrm{SLLL}}$); and the blue line indicates ($\log \mathbf{r}^{\mathrm{DEEP}} - \log \mathbf{r}^{\mathrm{VLLL}}$).

the condition number of its associated positive definite matrix, or equivalently, the condition number of **B** [22]. As an additional quality measure of lattice basis reduction to the Hermite factor and orthogonality defect, condition numbers were only used to compare the performance of reduction methods for positive definite quadratic forms recently (see e.g., [19,22]). Following Xu [19,22] for the reduction of positive definite matrices, we will use the concept of condition numbers to compare the performances of reduction methods for lattice vectors.

Having applied LLL, DEEP, SLLL, VLLL, and PLLL to the 10, 000 randomly simulated examples, we have obtained the 10, 000 condition numbers for each of these methods, which are denoted by κ_B^{LLL}, κ_B^{DEEP}, κ_B^{SLLL}, κ_B^{VLLL}, and κ_B^{PLLL}, respectively. The condition numbers of the original 10, 000 random examples are collected in the vector κ_B^{PROB}. The cdf curves of all these condition numbers, namely, κ_B^{LLL}, κ_B^{DEEP}, κ_B^{SLLL}, κ_B^{VLLL}, and κ_B^{PLLL}, together with κ_B^{PROB}, are shown in Figure 17. This figure has illustrated that (a) all the methods of reduction are effective to reduce the condition numbers of problems, as can also be seen from row κ_B of Table 1. VLLL could worsen the condition number of a problem significantly (compare the pink line of Figure 17). However, we should note that the success of a reduction method to reduce the condition number of a problem can depend on the original condition number and the rank of a lattice [19,22]; (b) PLLL consistently outperforms all the other four methods of reduction, namely, LLL, DEEP, SLLL, and VLLL, for almost all the problems; and (c) LLL, DEEP, and SLLL perform much better than VLLL.

In order to have a closer look at the random simulation results on condition numbers, we have carried out a direct comparison of each example among the five methods of reduction under study. More specifically, we have computed the differences of the condition numbers (in logarithm) $(\log\kappa_B^{\mathrm{PLLL}} - \log\kappa_B^{\mathrm{LLL}})$, $(\log\kappa_B^{\mathrm{PLLL}} - \log\kappa_B^{\mathrm{DEEP}})$, $(\log\kappa_B^{\mathrm{PLLL}} - \log\kappa_B^{\mathrm{SLLL}})$, and $(\log\kappa_B^{\mathrm{PLLL}} - \log\kappa_B^{\mathrm{VLLL}})$. The pdf histograms of these differences are shown in Figure 18, and the estimated probabilities from comparing PLLL with LLL, DEEP, SLLL, and VLLL are summarized in row κ_B of Table 2. This direct comparison of condition numbers for each example reaffirms the outstanding performance of PLLL over LLL, DEEP, SLLL, and VLLL in reducing the condition number of a problem, but unlike the cdf curves of Figure 17, Table 2, together with Figure 18, has shown that LLL, DEEP, SLLL, and VLLL could still be more successful than PLLL to reduce the condition number of a problem, though with a small probability from 0.05 to 0.14. Figure 18 has once again illustrated that VLLL is least effective to reduce condition numbers of problems.

The popular LLL algorithm is then compared with DEEP, SLLL, and VLLL. In a similar manner, we have computed the differences $(\log\kappa_B^{\mathrm{LLL}} - \log\kappa_B^{\mathrm{DEEP}})$, $(\log\kappa_B^{\mathrm{LLL}} - \log\kappa_B^{\mathrm{SLLL}})$, and $(\log\kappa_B^{\mathrm{LLL}} - \log\kappa_B^{\mathrm{VLLL}})$, whose pdf histograms are shown in Figure 19. The estimated probabilities from comparing LLL with DEEP, SLLL, and VLLL in terms of condition numbers are listed in row κ_B of Table 3. It is very clear from both Figure 19 and Table 3 that LLL performs much better than VLLL but worse than DEEP and SLLL in terms of condition numbers. Although

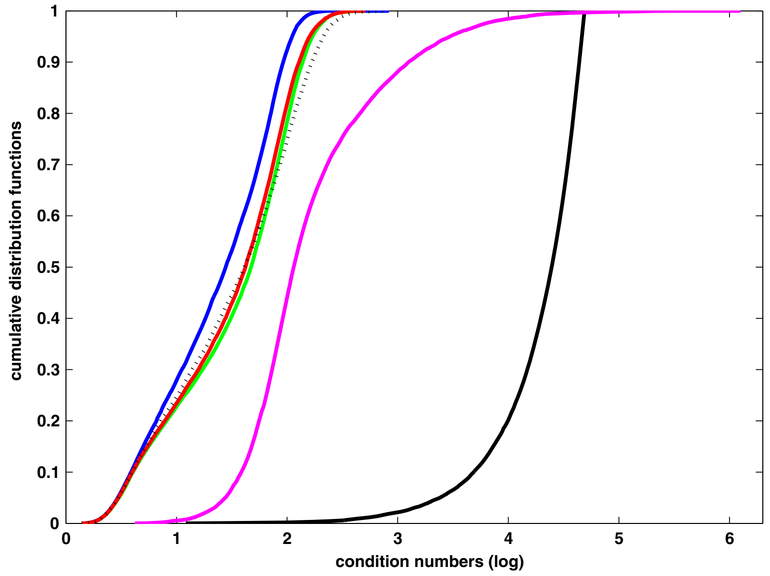

Figure 17 Cumulative distribution functions of condition numbers (in logarithm) from 10, 000 random examples. PROB, black line; LLL, green line; DEEP, dotted black line; SLLL, red line; VLLL, pink line; and PLLL, blue line.

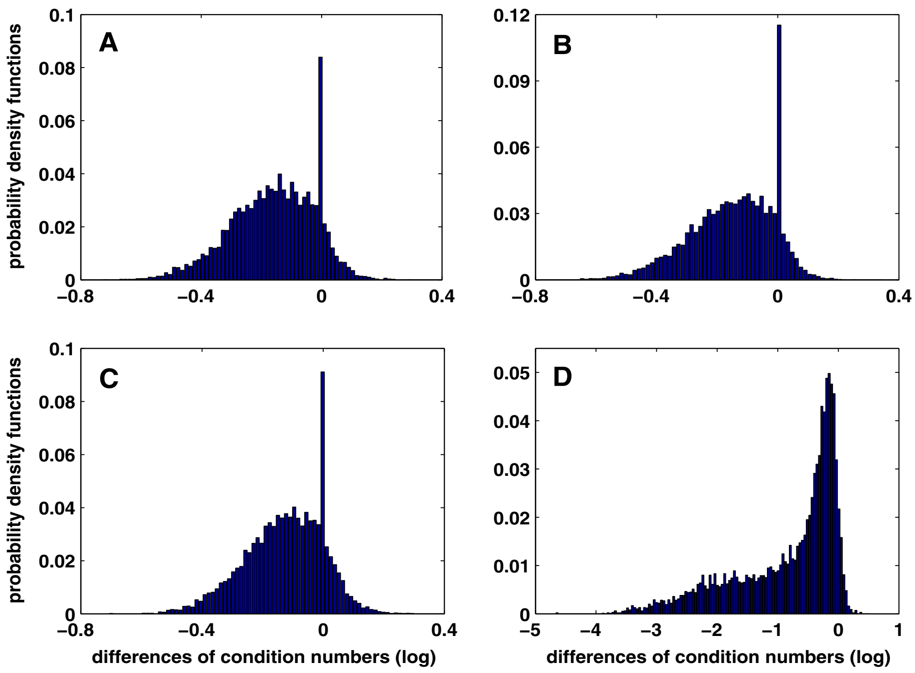

Figure 18 Histograms of probability density functions of condition numbers. This figure is to compare PLLL with LLL, DEEP, SLLL, and VLLL. Negative values indicate better results for PLLL. Panel **A**, ($\log \kappa_B^{\mathrm{PLLL}} - \log \kappa_B^{\mathrm{LLL}}$); panel **B**, ($\log \kappa_B^{\mathrm{PLLL}} - \log \kappa_B^{\mathrm{DEEP}}$); panel **C**, ($\log \kappa_B^{\mathrm{PLLL}} - \log \kappa_B^{\mathrm{SLLL}}$); and panel **D**, ($\log \kappa_B^{\mathrm{PLLL}} - \log \kappa_B^{\mathrm{VLLL}}$).

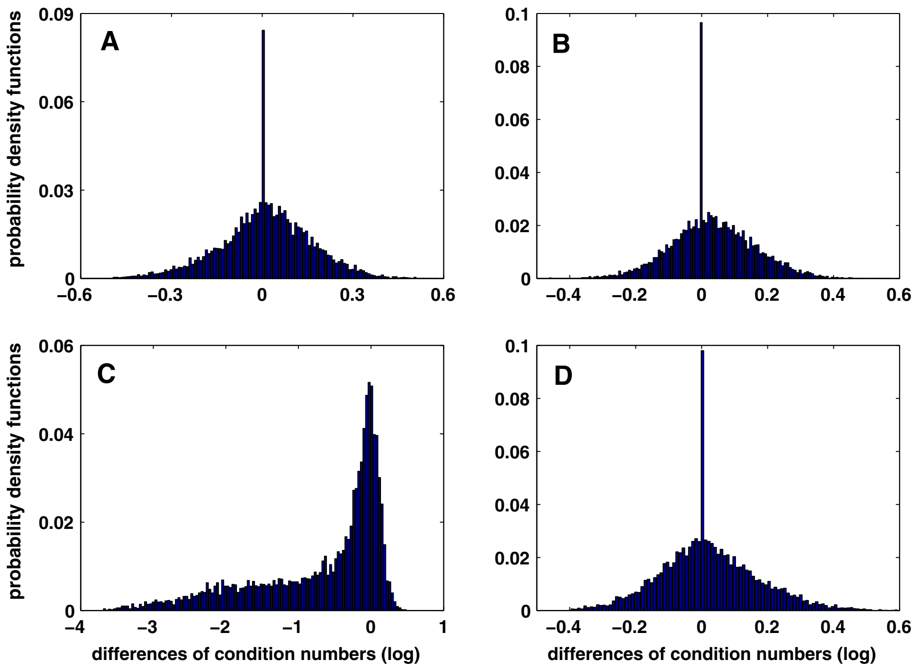

Figure 19 Histograms of probability density functions of condition numbers. This figure is to compare LLL with DEEP, SLLL, and VLLL in panels **A** to **C**, and DEEP with SLLL in panel **D**. Negative values indicate better results for LLL in panels **A, B,** and **C**, and better results for DEEP in panel **D**. Panel **A**, ($\log \kappa_B^{\mathrm{LLL}} - \log \kappa_B^{\mathrm{DEEP}}$); panel **B**, ($\log \kappa_B^{\mathrm{LLL}} - \log \kappa_B^{\mathrm{SLLL}}$); panel **C**, ($\log \kappa_B^{\mathrm{LLL}} - \log \kappa_B^{\mathrm{VLLL}}$); and panel **D**, ($\log \kappa_B^{\mathrm{DEEP}} - \log \kappa_B^{\mathrm{SLLL}}$).

Tables 2 and 3 might be used to conclude that SLLL is more effective than DEEP, we decide to provide the direct evidence by showing the pdf histogram of $(\log\kappa_B^{\text{DEEP}} - \log\kappa_B^{\text{SLLL}})$ in panel D of Figure 19. More precisely, SLLL performs better than DEEP to reduce the condition numbers of problems with 50.5% of the examples but worse than DEEP with 42.4% of the examples.

4.4 Quality of the Gram-Schmidt coefficients after reduction

The Gram-Schmidt coefficients μ_{ij} are known to satisfy the inequality $|\mu_{ij}| \leq 0.5$ for all $1 \leq j < i$ after reduction. Theoretical analysis of the LLL algorithm by Lenstra et al. [3] assumes the upper bound of $1/4$ for μ_{ij}^2 to derive all the lower and upper bounds on the reduced vectors $\mathbf{b}_i$. In order to investigate probabilistic behaviors of the LLL algorithm and its variants, one often assumes that all the Gram-Schmidt coefficients μ_{ij} after reduction are independent and uniformly distributed over $[-0.5, 0.5]$ (see e.g., [49,50,53]). It is only recently that numerical experiments were carried out by Nguyen and Stehlé [7], which have revealed that μ_{ij} are not necessarily distributed uniformly over $[-0.5, 0.5]$. Actually, the first experiments to gain the practical knowledge on μ_{ij} found that the distribution of $\mu_{i(i-1)}$ looks like a sunken basin, independent of the reduction algorithms (the LLL algorithm and its variant with deep insertions) and the used random bases [7]. They also found that with the increase of the gap of $(i-j)$, the distribution of μ_{ij} tends to have a uniform distribution over $[-0.5, 0.5]$. Similar experiments were followed by Schneider et al. [34]. They confirmed the same sunken shape of distribution for $\mu_{i(i-1)}$ after the LLL reduction as in Nguyen and Stehlé [7], went on to fit experimental data with a symmetrical polynomial function and finally used the fitted distribution to derive the mean values and variances of the shortest reduced vector.

Based on our own experiments on the 10,000 random lattices, we have obtained, in total, 6,149,762 μ_{ij} $(1 \leq j < i \leq m)$, and 304,941 $\mu_{i(i-1)}$ for each of the five reduction methods under study. The pdf histograms of 6,149,762 μ_{ij} are shown in Figure 20. It is obvious from Figure 20 that except for DEEP, the pdf curves of the other four reduction methods are symmetrical with one peak at zero, which are clearly not uniform. The pdf function of DEEP looks flatter than the other pdf curves but shows three peaks, two at the end points of -0.5 and 0.5 and one at zero, respectively. The pdf curves of μ_{ij} have shown a clear dependence on reduction methods, which is inconsistent with the observation by Nguyen and Stehlé [7] that the distributions of μ_{ij} seem to be method-independent. We

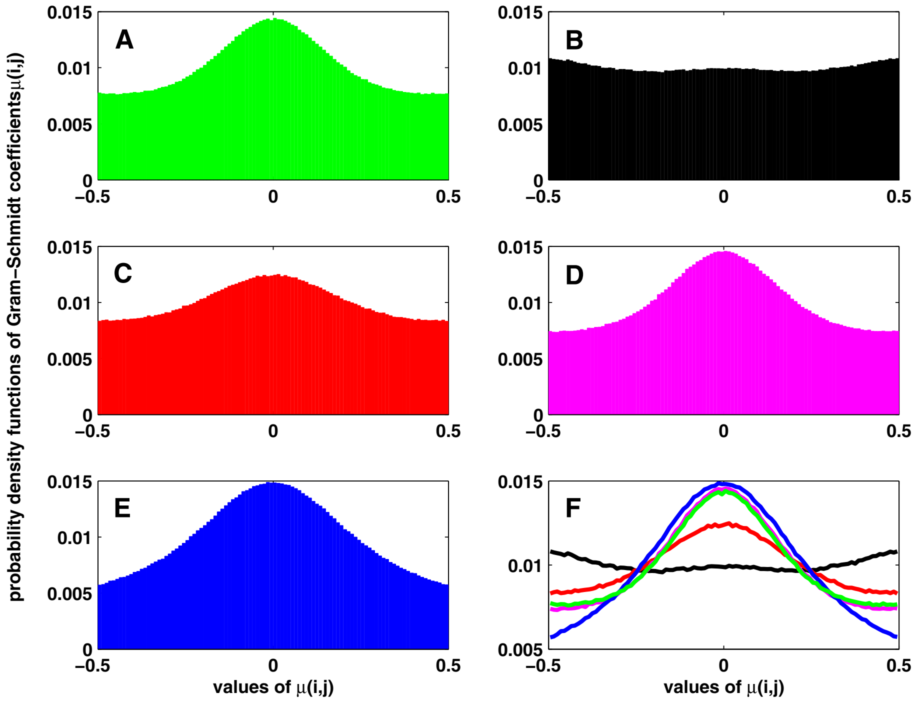

Figure 20 Pdf histograms of Gram-Schmidt coefficients μ_{ij} after reductions by LLL, DEEP, SLLL, VLLL, and PLLL. Panel **A**, LLL; panel **B**, DEEP; panel **C**, SLLL; panel **D**, VLLL; and panel **E**, PLLL. In order to show a direct comparison of these methods, we have also plotted all the pdf curves together in panel **F**: the green line indicates LLL; the black line indicates DEEP; the red line indicates SLLL; the pink line indicates VLLL; and the blue line indicates PLLL.

can also see from panel F of Figure 20 that PLLL results in most μ_{ij} around zero and least μ_{ij} close to the two end points of -0.5 and 0.5.

The pdf histograms of the Gram-Schmidt coefficients $\mu_{i(i-1)}$ are depicted for each of LLL, DEEP, SLLL, VLLL, and PLLL in Figure 21. By comparing Figure 21 with Figure 20, we can see that except for PLLL, the shapes of the pdf functions change dramatically for all the other four methods. Now, DEEP shows a clear shape of a sunken basin, as consistently found in Nguyen and Stehlé [7] and Schneider et al. [34]. SLLL results in the shape of a valley. Likely, the condition of deep insertions would force the Gram-Schmidt coefficients μ_{ij} with all $i - j > 1$ to take smaller values. As a result, the percentage of $|\mu_{i(i-1)}|$ closer to 0.5 might be relatively increased to turn the more or less flat pdf curve of DEEP in Figure 20 into a deeply sunken basin in Figure 21, if we limit ourselves to $\mu_{i(i-1)}$. The pdf functions for both LLL and VLLL have three peaks, two at the end points of -0.5 and 0.5 and one at zero. These should indicate that all these four methods of reduction, i.e., LLL, DEEP, SLLL and VLLL, tend to produce more $\mu_{i(i-1)}$ around the two end points of -0.5 and 0.5. It is, however, interesting to see that the pdf pattern of $\mu_{i(i-1)}$ from PLLL remains unchanged, still with a lot more number of points close to zero and a less number of points around -0.5 and 0.5. A great number of μ_{ij} close to zero, together with a small number of μ_{ij} around the two end points of -0.5 and 0.5, may also explain the excellent performance of PLLL, as seen in the performance analysis of Section 4.3.

5 Conclusion

Reduction is to make a reduced basis of lattice as orthogonal as possible and as short as possible. A breakthrough came with the invention of the LLL algorithm by Lenstra et al. [3]. Its variants have since been substantially developed and applied widely to solve highly interdisciplinary problems (see e.g., [5]). We have extended the parallel reduction method developed recently by Xu [22] for positive definite quadratic forms to lattice basis vectors, which is referred to as the improved LLL algorithm with fixed complexity in this paper. The proposed parallel algorithm consists of three basic components: (a) sorting the lattice vectors. Here, we implement two versions, one directly in ascending order of the lengths of the vectors and the other by perturbing the ascending sorting strategy in the first one or two iterations with the order of the lengths of the orthogonalized vectors; (b) the Gram-Schmidt orthogonalization process; and (c) a complete size reduction on the whole Gram-Schmidt coefficient

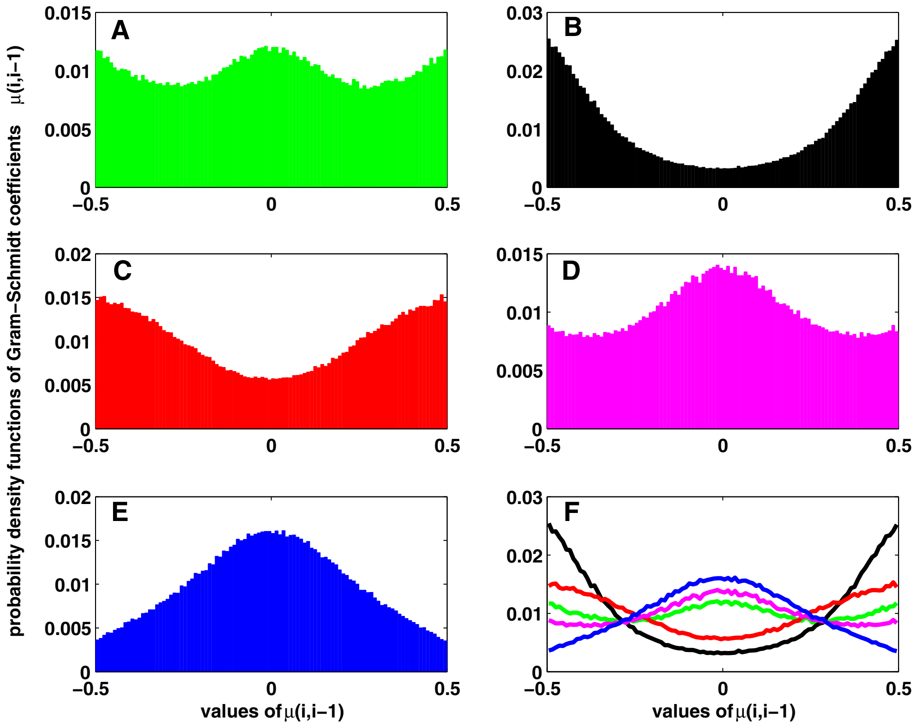

Figure 21 Pdf histograms of Gram-Schmidt coefficients $\mu_{i(i-1)}$ after reductions by LLL, DEEP, SLLL, VLLL and PLLL. Panel **A**, LLL; panel **B**, DEEP; panel **C**, SLLL; panel **D**, VLLL; and panel **E**, PLLL. In order to show a direct comparison of these methods, we have also plotted all the pdf curves together in panel **F**: the green line indicates LLL; the black line indicates DEEP; the red line indicates SLLL; the pink line indicates VLLL; and the blue line indicates PLLL.

matrix. The complexity of the algorithm is fixed by setting a maximum number of iterations.

Reduction is to make the reduced basis as short as possible and as orthogonal as possible. For an ILS problem, the absolute lengths of the basis, or equivalently, the magnitudes of the diagonal elements of the weight matrix of the ILS problem, are not important, since they can be made arbitrarily small without affecting the solution to the ILS problem [20]. From this point of view, a good quality measure of reduction for an ILS problem should minimize the maximum relative length of the reduced basis. On the other hand, although the orthogonality defect $\mathcal{O}(\mathbf{B})$ has been defined and widely used to evaluate the quality of a reduction method, it basically does not help much to tell the extent of orthogonality of the reduced basis at all. We have proposed the minimum angle of a reduced basis as an alternative quality measure of orthogonality, which may be intuitively more straightforward than the orthogonality defect.

We have carried out a large scale of random experiments to investigate the output quality and practical running behaviors of the five reduction methods for low-dimensional GPS applications: (a) the original LLL algorithm, (b) its variant with deep insertions, (c) the LLL algorithm with sorted QR ordering; (d) an LLL algorithm with fixed complexity proposed recently by Vetter et al. [14]; and finally, (e) the improved LLL algorithm with fixed complexity proposed in this paper. The five reduction methods have been extensively compared on the basis of six quality measures of reduction, namely, the orthogonality defect, the minimum angle, the Hermite factor, the length of the shortest reduced vector, the maximum length ratio, and the condition number of the reduced basis. The improved LLL algorithm with fixed complexity has been shown to perform as well as the LLL algorithm with deep insertions with respect to the quality measures of the length of the shortest reduced vector and the Hermite factor, to be slightly less efficient with respect to the maximum length ratio but otherwise to outperform deep insertions on the other three quality measures of orthogonality and condition numbers.

The random experiments have clearly shown that the LLL variant with deep insertions is very powerful in producing a uniformly short reduced basis, as might be expected theoretically. However, it is much less efficient than the improved LLL algorithm with fixed complexity in turning the reduced basis as orthogonal as possible. As a consequence, the former cannot compete with the latter from the combined point of view of length and orthogonality defects. In particular, since the LLL algorithm with deep insertions can be super exponential in complexity, as confirmed by the experiment results in this paper, the fixed complexity of our improved LLL algorithm can be profoundly more efficient computationally than this variant of LLL. Our LLL algorithm with fixed complexity has been shown to perform significantly better than the other four methods of reduction. The random simulations have also clearly demonstrated that the QR sorting can be a significant plus to the LLL algorithm on all the account of six quality measures. The LLL algorithm with fixed complexity proposed recently by Vetter et al. [14] has been shown to have the worst performance on the account of all the six quality measures.

The random experiments have illustrated that both quality measures of orthogonality, i.e., the orthogonality defect and the minimum angle, are not sufficient to properly reflect the quality of a reduction method. They should be interpreted with caution and used together with the quality measures of the shortest reduced vector and/or the condition number in order to properly evaluate the quality or performance of a reduction method. Condition numbers are appropriate to correctly reflect the combined effect/quality of a reduction method with respect to orthogonality and length defects.

Based on the 10, 000 random examples, we have also investigated the mean running behaviors of the LLL algorithm and its two variants with the sorted QR ordering and deep insertions. The LLL variant with deep insertions is confirmed to have a super-exponential complexity, as stated by Gama and Nguyen [45]. The simulations have also supported the widely spread belief that the LLL algorithm performs much better in practice than theoretically expected. Actually, three theoretical formulae on the upper bound of mean number of iterations tend to converge to a small constant with the increase of lattice ranks.

The simulation results on the distribution of Gram-Schmidt coefficients after reduction have reaffirmed that Gram-Schmidt coefficients are not uniformly distributed, as observed in the excellent experimental study by Nguyen and Stehlé [7] and otherwise often assumed for studying probabilistic behaviors of the LLL algorithm and its variants (see e.g., [49,50,53]). The distribution of $\mu_{i(i-1)}$ with deep insertions shows a shape of deeply sunken basin, which is consistent with the reports by Nguyen and Stehlé [7] and Schneider et al. [34]. However, our distribution results on the whole Gram-Schmidt coefficients have clearly shown that the distributions of μ_{ij} are reduction-dependent, which contradicts with the statement by Nguyen and Stehlé [7] that these distributions seem to be independent of both reduction methods and lattice models. The improved LLL algorithm with fixed complexity has been shown to have a consistent distribution either for all μ_{ij} or for $\mu_{i(i-1)}$ only. In particular, the distributions of both μ_{ij} and $\mu_{i(i-1)}$ indicate that our improved LLL algorithm with fixed complexity tends to produce more Gram-Schmidt coefficients closer to zero and less Gram-Schmidt coefficients closer to the two end points of −0.5 and 0.5 than any other methods of reduction under study. Finally, we should note, however, that we have conducted all these experiments with low-dimensional GPS applications in mind. If the reader is interested more in very high-dimensional (say a few hundreds and above) cryptographic applications, then further work of random simulations may be necessary.

Competing interests
The author declares that he has no competing interests.

Acknowledgements
The author would like to thank the two anonymous reviewers for their constructive comments, which help clarify some of the points in the paper. This work is partially supported by a Grant-in-Aid for Scientific Research (C25400449).

Received: 9 September 2012 Accepted: 29 July 2013
Published: 19 August 2013

References

1. JWS Cassels, *An Introduction to the Geometry of Numbers*. (Springer, Berlin, 1971)
2. PM Gruber, CG Lekkerkerker, *Geometry of Numbers*. (North-Holland, Amsterdam, 1987)
3. AK Lenstra, HW Lenstra, L Lovász, Factoring polynomials with rational coefficients. Math. Ann. **261**, 515–534 (1982)
4. JC Lagarias, AM Odlyzko, Solving low-density subset sum problems. J. ACM. **32**, 229–246 (1985)
5. PQ Nguyen, B Vallée (eds.), *The LLL Algorithm — Survey and Applications* (Springer, Berlin, 2010)
6. D Stehlé, in *The LLL Algorithm*. Floating-point LLL: theoretical and practical aspects. ed. by PQ Nguyen, B Vallée, (Springer, Berlin, 2010), pp. 179–213
7. PQ Nguyen, D Stehlé, in *ANTS 2006, LNCS 4076*. LLL on the average. ed. by F Hess, S Pauli, M Pohst, (Springer, Berlin, 2006), pp. 238–256
8. K Aardal, F Eisenbrand, in *The LLL Algorithm*. The LLL algorithm and integer programming. ed. by PQ Nguyen, B Vallée (Springer, Berlin, 2010), pp. 293–314
9. U Fincke, M Pohst, Improved methods for calculating vectors of short length in a lattice, including a complexity analysis. Math. Comput. **44**, 463–471 (1985)
10. HW Lenstra, Integer programming with a fixed number of variables. Math. Oper. Res. **8**, 538–548 (1983)
11. E Agrell, T Eriksson, A Vardy, K Zeger, Closest point search in lattices. IEEE Trans Inf. Theory. **48**, 2201–2214 (2002)
12. H Artés, D Seethaler, F Hlawatsch, Efficient detection algorithms for MIMO channels: a geometrical approach to approximate ML detection. IEEE Trans Signal Proc. **51**, 2808–2820 (2003)
13. J Jaldén, D Seethaler, G Matz, in *Proc IEEE Int. Conference on Acoustics, Speech, Signal Processing (ICASSP)*. Worst- and average-case complexity of LLL lattice reduction in MIMO wireless systems (Las Vegas, March 30 - April 4 2008), pp. 2685–2688
14. H Vetter, V Ponnampalam, M Sandell, PA Hoeher, Fixed complexity LLL algorithm. IEEE Trans. Signal Proc. **57**, 1634–1637 (2009)
15. O Regev, On Lattices, learning with errors, random linear codes, and cryptography. J. ACM. **56**, 34:1–34:40 (2009)
16. A Joux, J Stern, Lattice reduction: a toolbox for the cryptanalyst. J. Cryptol. **11**, 161–185 (1998)
17. GT Herman, A Kuba (eds.), *Advances in Discrete Tomography and its Applications* (Birkhäuser, Boston, 2007)
18. B Hofmann-Wellenhof, H Lichtenegger, J Collins, *GPS — Theory and Practice*. (Springer-Verlag, Berlin, 1992)
19. PL Xu, Random simulation and G P S decorrelation. J. Geod. **75**, 408–423 (2001)
20. PL Xu, Voronoi cells, probabilistic bounds and hypothesis testing in mixed integer linear models. IEEE Trans. Inf. Theory. **52**, 3122–3138 (2006)
21. PL Xu, in *Handbook of Geomathematics*. Mixed integer linear models. ed. by W Freeden, Z Nashed, T Sonar (Springer, Berlin, 2010), pp. 1129–1157
22. PL Xu, Parallel Cholesky-based reduction for the weighted integer least squares problem. J. Geod. **86**, 35–52 (2012). doi:10.1007/s00190-011-0490-y
23. C PL Xu, JN Shi, Liu, Integer estimation methods for GPS ambiguity resolution: an applications-oriented review and improvement. Surv. Rev. **44**, 59–71 (2012)
24. PL Xu, E Cannon, G Lachapelle, Mixed integer programming for the resolution of GPS carrier phase ambiguities. Paper presented at IUGG95 Assembly, Boulder (July 2–14 1995). arXiv preprint arXiv:1010.1052, 2010
25. PL Xu, E Cannon, G Lachapelle, Mixed Integer Observation Models, GPS Decorrelation and Integer Programming. Technical Report Nr.2000.2, Geodetic Institute, Stuttgart University, 2000
26. XW Chang, X Yang, T Zhou, MLAMBDA: a modified LAMBDA method for integer least-squares estimation. J. Geod. **79**, 552–565 (2005)
27. XW Chang, T Zhou, MILES: MATLAB package for solving mixed integer least squares problems. GPS Solut. **11**, 289–294 (2007)
28. I Smeets, in *The LLL Algorithm*. The history of the LLL-algorithm. eds. PQ Nguyen, B Vallée (Springer, Berlin, 2010), pp. 1–17
29. B Vallée, A Vera, in *The LLL Algorithm*. Probabilistic analyses of lattice reduction algorithms. ed. by PQ Nguyen, B Vallée (Springer, Berlin, 2010), pp. 71–144
30. H Daudé, B Vallée, An upper bound on the average number of iterations of the LLL algorithm. Theor. Comput. Sci. **123**, 95–115 (1994)
31. A Akhavi, Random lattices, threshold phenomena and efficient reduction algorithms. Theor. Comput. Sci. **287**, 359–385 (2002)
32. C Ling, N Howgrave-Graham, in *Proc. Int. Symp. Inform. Theory(ISITĄ2007)*. Effective LLL reduction for lattice decoding (Nice France, June 24–29, 2007), pp. 196–200
33. C Ling, WH Mow, N Howgrave-Graham, Reduced and fixed-complexity variants of the LLL algorithm for communications. IEEE Trans. Commun. **61**, 1040–1050 (2013)
34. M Schneider, J Buckmann, R Lindner, in *Proc. WEWoRC*, vol. 2009. Probabilistic analysis of LLL reduced bases (Springer, Berlin, 2010)
35. CP Schnorr, M Euchner, Lattice basis reduction: improved practical algorithms and solving subset sum problems. Math. Prog. **66**, 181–199 (1994)
36. W Backes, S Wetzel, Heuristics on lattice basis reduction in practics. ACM J. Exp. Algorithmics. **7**, 1–21 (2002)
37. W Backes, S Wetzel, *The effect of sorting on lattice basis reduction. Paper presented at LLL+25 Conference*. (Poster Session, Caen, France, 29 Jun - 1 Jul 2007). http://www.cs.stevens.edu/~wbackes/paper/paper_lll25_wb_sw.pdf
38. YH Gan, WH Mow, Novel joint sorting and reduction techniques for delay-constrained LLL-aided MIMO detection. IEEE Signal Proc. Lett. **15**, 194–197 (2008)
39. C Ling, WH Mow, in *Proc. 2009 IEEE Information Theory Workshop*. A unified view of sorting in lattice reduction: from V-BLAST to LLL and beyond (Taormina, Sicily Italy, Oct 11-16, 2009), pp. 529–533
40. M Seysen, Simultaneous reduction of a lattice basis and its reciprocal basis. Combinatorica. **13**, 363–376 (1993)
41. W Freeden, *Metaharmonic Lattice Point Theory*. (CRC Press, London, 2011)
42. G Nemhauser, L Wolsey, *Integer and Combinatorial Optimization*. (Wiley, New York, 1988)
43. Q Zhou, X Ma, Element-based lattice reduction algorithms for large MIMO detection. IEEE J. Sel. Areas Comm. **31**, 274–286 (2013)
44. BA LaMacchia, *Basis reduction algorithms and subset sum problems*. (MIT, Master thesis, 1991)
45. N Gama, PQ Nguyen, in *EUROCRYPT 2008, LNCS*. Predicting lattice reduction. vol. 4965, (Springer, Heidelberg, 2008), pp. 31–51
46. DW Waters, JR Barry, in *Proc. 2005 Int. Conf. Wireless Networks, Communications and Mobile Computing*. A reduced-complexity lattice-aided decision-feedback detector (Maui, HI, 13-16 June 2005), pp. 845–850
47. D Wübben, R Böhnke, J Rinas, V Kühn, KD Kammeyer, Efficient algorithm for decoding layered space-time codes. Electronics Lett. **37**, 1348–1350 (2001)
48. D Wübben, R Böhnke, V Kühn, KD Kammeyer, in *Proc. IEEE 58th Vehicular Technology Conference, VTC 2003-Fall*. MMSE extension of V-BLAST based on sorted QR decomposition. vol. 5, (Orlando, Florida, 6-9 Oct 2003), pp. 508–512

49. M Ajtai, in *Proc. 35th Annual ACM Symposium on Theory of Computing*. The worst-case behavior of Schnorr's algorithm approximating the shortest nonzero vector in a lattice (ACM, New York, 2003), pp. 396–406
50. M Ajtai, Optimal lower bounds for the Korkine-Zolotareff parameters of a lattice and for Schnorr's algorithm for the shortest vector problem. Theory of Computing. **4**, 21–51 (2008)
51. PL Xu, Spectral theory of constrained second-rank symmetric random tensors. Geophys. J. Int. **138**, 1–24 (1999)
52. PL Xu, Isotropic probabilistic models for directions, planes and referential systems. Proc. Roy. Soc. London. **A458**, 2017–2038 (2002)
53. M Madritsch, B Vallée, in *LATIN 2010*. Modelling the LLL algorithm by sandpiles. LNCS. vol. 6034, ed. by A López-Ortiz (Springer, Berlin, 2010), pp. 267–281

doi:10.1186/1687-6180-2013-137
Cite this article as: Xu: **Experimental quality evaluation of lattice basis reduction methods for decorrelating low-dimensional integer least squares problems.** *EURASIP Journal on Advances in Signal Processing* 2013 **2013**:137.