J Geod (2012) 86:35–52
DOI 10.1007/s00190-011-0490-y

ORIGINAL ARTICLE

Parallel Cholesky-based reduction for the weighted integer least squares problem

Peiliang Xu

Received: 10 November 2010 / Accepted: 28 May 2011 / Published online: 12 June 2011

Abstract The LLL reduction of lattice vectors and its variants have been widely used to solve the weighted integer least squares (ILS) problem, or equivalently, the weighted closest point problem. Instead of reducing lattice vectors, we propose a parallel Cholesky-based reduction method for positive definite quadratic forms. The new reduction method directly works on the positive definite matrix associated with the weighted ILS problem and is shown to satisfy part of the inequalities required by Minkowski's reduction of positive definite quadratic forms. The complexity of the algorithm can be fixed a priori by limiting the number of iterations. The simulations have clearly shown that the parallel Cholesky-based reduction method is significantly better than the LLL algorithm to reduce the condition number of the positive definite matrix, and as a result, can significantly reduce the searching space for the global optimal, weighted ILS or maximum likelihood estimate.

Keywords Global positioning system (GPS) · Integer linear model · Integer least squares · Closest point problem · Reduction of quadratic forms · LLL reduction · Multiple-input–multiple-output

1 Introduction

Consider the following integer linear model:

$$\mathbf{y} = \mathbf{B}\mathbf{z} + \boldsymbol{\epsilon}, \tag{1}$$

where $\mathbf{y}$ is an n-dimensional vector of real-valued data, $\mathbf{B}$ is an $(n \times m)$ real-valued matrix of full column rank, and $\boldsymbol{\epsilon}$ is the random error vector of the measurements $\mathbf{y}$ with the variance–covariance matrix $\mathbf{W}^{-1}\sigma^2$. The matrix $\mathbf{W}$ has often been called a *weight matrix* of the measurements $\mathbf{y}$ in the engineering literature (see, e.g. Koch 1999; Xu 2006), and σ^2 is generally assumed to be an unknown positive scalar. A basic problem in connection with the model (1) is to estimate the unknown integer vector $\mathbf{z}$ from the real-valued data $\mathbf{y}$, i.e., $\mathbf{z} \in \mathbb{Z}^m$, and $\mathbb{Z}^m$ is defined as an m-dimensional integer space.

The integer linear model (1) has been an interdisciplinary subject of study, ranging, for example, from the geometry of numbers and integer programming to multiple-input–multiple-output (MIMO) communication systems, learning with errors, cryptography, crystallography, and global navigation satellite systems (GNSS). Without the term of random errors $\boldsymbol{\epsilon}$, the deterministic part of the model (1), namely, $\mathbf{y} = \mathbf{B}\mathbf{z}$ with $\mathbf{z} \in \mathbb{Z}^m$, defines a lattice of discrete points in $\mathbb{R}^n$ with the basis of m linearly independent column vectors of $\mathbf{B}$ and is the starting point of the geometry of numbers (see, e.g. Conway and Sloane 1999; Gruber and Lekkerkerker 1987), where $\mathbb{R}^n$ is an n-dimensional real-valued space. In a MIMO communication system or a problem of learning with errors, the key issue is to correctly estimate the integer unknown vector $\mathbf{z}$ from the noise-contaminated received data $\mathbf{y}$, which can be respectively interpreted in the language of decoding and/or codewords (see, e.g. Agrell et al. 2002; Regev 2009). In cryptography, a cryptographer attempts to hide secret information on the basis of the NP-hard complexity of solving the integer linear model (1) on one hand and uses approaches to solving the same problem (1) to attack or disclose a cryptosystem (see, e.g. Joux and Stern 1998; Regev 2009) on the other hand. However, a crystallographer would be concerned with reconstructing the lattice structure of a crystalline by solving for the integer coordinates from a set of X-ray projections along different lattice directions;

P. Xu (✉)
Disaster Prevention Research Institute, Kyoto University,
Uji, Kyoto 611-0011, Japan
e-mail: pxu@rcep.dpri.kyoto-u.ac.jp

this problem is also known as a discrete tomography (see, e.g. Brunetti and Daurat 2003; Gardner et al. 1999). In GNSS, one is interested in correctly finding the integer unknown vector $\mathbf{z}$, which is essential for precise GNSS positioning (see, e.g. Teunissen 1993; Hofmann-Wellenhof et al. 1992; Xu et al. 1995; Grafarend 2000). In fact, in the case of GNSS precise positioning, we have to deal with a slightly more complicated model with both types of unknown parameters as follows:

$$\mathbf{y} = \mathbf{A}\boldsymbol{\beta} + \mathbf{B}\mathbf{z} + \boldsymbol{\epsilon}, \tag{2}$$

where $\mathbf{A}$ is an $(n \times t)$ real-valued matrix of full column rank, and $\boldsymbol{\beta}$ is a real-valued nonstochastic vector, i.e., $\boldsymbol{\beta} \in \mathbb{R}^t$ with $\mathbb{R}^t$ being a t-dimensional real-valued space. The model (2) has also been called a mixed integer linear model (see, e.g. Xu et al. 1995; Grafarend 2000; Xu 1998, 2006, 2010). By projecting the problem (2) onto the orthogonal complement of the range of $\mathbf{A}$ with the projection matrix $[\mathbf{I} - \mathbf{A}(\mathbf{A}^{\mathrm{T}}\mathbf{W}\mathbf{A})^{-1}\mathbf{A}^{\mathrm{T}}\mathbf{W}]$ (Teunissen 1993) or by following the two-step approach of Xu et al. (1995), we can reduce the mixed integer linear model (2) to the integer linear model (1).

Applying the weighted least squares (LS) criterion to the integer linear model (1), we have the following minimization model:

$$\min_{\mathbf{z}\in\mathbb{Z}^m}: \quad \mathrm{F}(\mathbf{z}) = (\mathbf{y} - \mathbf{B}\mathbf{z})^{\mathrm{T}}\mathbf{W}(\mathbf{y} - \mathbf{B}\mathbf{z}), \tag{3}$$

which is also called the *integer least squares* (*ILS*) *problem*. If the random errors $\boldsymbol{\epsilon}$ are assumed to be normally distributed, the integer optimization problem (3) can naturally be derived from the maximum likelihood (ML) principle (see, e.g. Artés et al. 2003; Damen et al. 2003). The ILS problem (3) can be rewritten equivalently as follows:

$$\min_{\mathbf{z}\in\mathbb{Z}^m}: \quad \mathrm{F}(\mathbf{z}) = (\mathbf{z} - \mathbf{z}_f)^{\mathrm{T}}\mathbf{W}_f(\mathbf{z} - \mathbf{z}_f), \tag{4}$$

where

$$\mathbf{W}_f = \mathbf{B}^{\mathrm{T}}\mathbf{W}\mathbf{B},$$
$$\mathbf{z}_f = (\mathbf{B}^{\mathrm{T}}\mathbf{W}\mathbf{B})^{-1}\mathbf{B}^{\mathrm{T}}\mathbf{W}\mathbf{y} = \mathbf{W}_f^{-1}\mathbf{B}^{\mathrm{T}}\mathbf{W}\mathbf{y},$$

(see, e.g. Teunissen 1993; Xu et al. 1995; Xu 2006). If $\mathbf{W} = \mathbf{I}$, then (3) has been better known as the *closest vector problem* (see, e.g. Nemhauser and Wolsey 1988) or as the *closest point problem* (see, e.g. Agrell et al. 2002; Babai 1986). Furthermore, if $\mathbf{y} = \mathbf{0}$, $\mathbf{W} = \mathbf{I}$ and $\mathbf{z} \neq \mathbf{0}$, then (3) has been well known as the *shortest vector problem* (see, e.g. Grötschel et al. 1988; Nemhauser and Wolsey 1988).

The ILS problem (3) is obviously a standard quadratic integer programming model (see, e.g. Taha 1975; Nemhauser and Wolsey 1988; Li and Sun 2006). In principle, one can use any appropriate integer programming methods, as documented, for example, in the above mentioned books, to find the global optimal integer solution to (3). In general, one may classify the approaches to solve (3) into two types: exact and approximate. Finding the exact solution to (3) is known to be NP-hard (see, e.g. Agrell et al. 2002; Grötschel et al. 1988; Nemhauser and Wolsey 1988; Conway and Sloane 1999). Almost all efficient algorithms to find the exact solution to (3) are designed by the combination of reduction methods with a proper enumeration strategy. The most widely used (exact) method of enumeration was originally proposed by Pohst (1981) and detailed in Fincke and Pohst (1985). The efficiency of searching for the minimum solution was further improved by Schnorr and Euchner (1994), which culminated with a celebrated algorithm for many practical applications (see, e.g. Agrell et al. 2002; Damen et al. 2003). The second type of methods is mainly focused on finding an approximate solution to (3). The simplest approximate approach can either solve (3) as if it were a real-valued problem and then round the floating point solution to its nearest integer (see, e.g. Taha 1975) or sequentially fix the floating point component to its nearest integer at each step of iteration. However, the most widely used method to find an approximate solution may be based on lattice reduction, in particular, the polynomial-time reduction method invented by Lenstra et al. (1982) which has now been popularized as the LLL algorithm after the initials of the family names of its three inventors.

Actually, reduction is not only a method to help quickly find an approximate solution to the ILS problem (3) but can also be a key component in efficiently speeding up finding the exact minimum integer solution, as can be clearly seen in Pohst (1981), Fincke and Pohst (1985) and Schnorr and Euchner (1994). Obviously, correctly estimating the integer unknown parameters $\mathbf{z}$ is the only part of statistical inference in connection with the integer linear model (1). If the reader would like to know more beyond the integer estimation, he or she should consult Shannon (1959) for a lower error probabilistic bound for incorrectly estimating $\mathbf{z}$, and Xu (2006) both for lower error probabilistic bounds better than that by Shannon (1959) and for statistical hypothesis testing in the mixed integer linear model (2). In this contribution, we will focus on reduction. More specifically, we will develop an integer parallel Cholesky-based reduction method for solving the ILS problem (3) in Sect. 2. In Sect. 3, we will briefly discuss the quality measures of reduction to be used for performance comparison of different reduction methods. We then design experiments to demonstrate the parallel Cholesky-based reduction method, to investigate the effect of sorting on reduction and to compare the parallel reduction method with the popular LLL reduction algorithm in Sect. 4. We will conclude this contribution with some remarks in Sect. 5.

2 Reduction of lattice vectors and positive definite quadratic forms

2.1 Reduction of lattice vectors

Given m independently linear vectors $\mathbf{b}_i (i = 1, 2, \ldots, m)$ of dimension n, a lattice is the discrete point set in the real-valued subspace of $\mathbb{R}^n$ and is defined as follows:

$$\mathcal{L} = \left\{ \sum_{i=1}^{m} \mathbf{b}_i z_i \mid z_i \in \mathbb{Z} \right\}, \tag{5}$$

(see, e.g. Conway and Sloane 1999; Gruber and Lekkerkerker 1987). Here the vectors $\mathbf{b}_i (i = 1, 2, \ldots, m)$ are called a basis of the lattice $\mathcal{L}$, and m is the rank of $\mathcal{L}$.

Reduction is an important problem in the geometry of numbers. The purpose of reduction is mainly twofold: (i) to make all the basis vectors of $\mathcal{L}$ become the shortest, or at least, as short as possible; and (ii) to make all the basis vectors of $\mathcal{L}$ as orthogonal as possible. Given a lattice $\mathcal{L}$, its generated basis is not unique, however. In fact, given a basis $\mathbf{B} = (\mathbf{b}_1, \mathbf{b}_2, \ldots, \mathbf{b}_m)$ of $\mathcal{L}$, any basis of the form $\mathbf{B}_r = \mathbf{BG}$ will exactly reproduce the same lattice $\mathcal{L}$, where $\mathbf{G}$ is a unimodular matrix with all its elements being integers and the absolute value of its determinant being equal to unity.

If $\mathbf{W} = \mathbf{I}$ and by substituting $\mathbf{B} = \mathbf{B}_r\mathbf{G}^{-1}$ into the ILS problem (4), we have

$$\min_{\mathbf{z}_\mathrm{N} \in \mathbb{Z}^m}: \quad \mathrm{F}(\mathbf{z}_\mathrm{N}) = (\mathbf{z}_\mathrm{N} - \mathbf{z}_r)^\mathrm{T}\mathbf{W}_r(\mathbf{z}_\mathrm{N} - \mathbf{z}_r), \tag{6a}$$

where

$$\mathbf{z}_\mathrm{N} = \mathbf{G}^{-1}\mathbf{z}, \tag{6b}$$

$$\mathbf{z}_r = \mathbf{G}^{-1}\mathbf{z}_f, \tag{6c}$$

$$\mathbf{W}_r = \mathbf{B}_r^\mathrm{T}\mathbf{B}_r. \tag{6d}$$

Ideally, if all the column vectors of $\mathbf{B}_r$ are mutually orthogonal, then $\mathbf{W}_r$ of (6a) or (6d) becomes diagonal. In this idealized case, the global optimal integer solution $\mathbf{z}_\mathrm{N}$ can be trivially obtained by simply rounding the real-valued vector $\mathbf{z}_r$ to its nearest integer (see, e.g. Taha 1975). Thus, the solution to the quadratic convex integer programming problem (3) can be readily found through the integer transformation $\mathbf{z} = \mathbf{G}\mathbf{z}_\mathrm{N}$. Unfortunately, $\mathbf{B}_r$ is generally not (column-) orthogonal. Thus, the best possible one can hope is to find a unimodular matrix $\mathbf{G}$ to reduce $\mathbf{B}$ such that $\mathbf{B}_r$ is as orthogonal as possible.

Fincke and Pohst (1985) showed that reduction methods can be very powerful in aiding enumeration to quickly find the exact integer solution to (3). Among the reduction methods compared, they reported that the LLL algorithm of reduction by Lenstra et al. (1982) is the fastest. With a further significant improvement by Schnorr and Euchner (1994) to incorporate a new strategy of enumeration, this hybrid, reduction-aided method of enumeration developed by the authors mentioned here has found wide spread applications, as can be seen, e.g. in Agrell et al. (2002); Damen et al. (2003) and a long list of references therein and an even longer list of their citing articles. Thus, in order to show the performance of our reduction method to be developed in the next Sect. 2.2, we will confine ourselves to the most widely used LLL reduction method for comparison. For more reduction methods such as Hermite reduction, Korkine–Zolotarev reduction and Minkowski reduction, the reader may consult, e.g. Afflerbach and Grothe (1985); Banihashemi and Khandani (1998); Helfrich (1985); Nguyen and Stehle (2004) and Seysen (1993).

Since the LLL reduction has been well documented in the literature (see, e.g. Lenstra et al. 1982; Nguyen and Stehlé 2009), we will only outline briefly the method for convenience of comparison. To realize the two basic requirements of size reduction and almost mutual orthogonality of lattice vectors, the LLL reduction method is essentially based on two key components: the Gram–Schmidt orthogonalization process and Lovász condition for vector swapping. Like any other reduction methods, given the basis vectors $\mathbf{b}_i$ $(i = 1, 2, \ldots, m)$ with a full lattice rank m in $\mathbb{R}^n$, the LLL reduction method starts with the Gram–Schmidt orthogonalization as follows:

$$\mathbf{b}_i^* = \mathbf{b}_i - \sum_{j=1}^{i-1} \mu_{ij}\mathbf{b}_j^*, \tag{7a}$$

$$\mu_{ij} = \frac{\mathbf{b}_i^\mathrm{T}\mathbf{b}_j^*}{\|\mathbf{b}_j^*\|^2}, \tag{7b}$$

where $\|\cdot\|$ stands for the Euclidean L_2-norm of a vector. To guarantee size reduction, it is required that $|\mu_{ij}| \le 1/2$ for all $1 \le j < i \le m$. For any $j < i$, if $|\mu_{ij}| > 1/2$, then $\mathbf{b}_i$ is replaced with $(\mathbf{b}_i - \lceil \mu_{ij} \rfloor \mathbf{b}_j)$, with $\lceil \mu_{ij} \rfloor$ being the integer nearest to μ_{ij}.

To decide whether the orthogonalization process (7) can continue to the next vector $\mathbf{b}_{i+1}$, Lenstra et al. (1982) compared the current $\mathbf{b}_i^*$ against the previous $\mathbf{b}_{i-1}^*$ and required them to satisfy Lovász condition:

$$\|\mathbf{b}_{i-1}^*\|^2 \le \delta^2 \|\mathbf{b}_i^* + \mu_{i(i-1)}\mathbf{b}_{i-1}^*\|^2 \tag{8}$$

for all $1 < i \le m$, where $\delta \in (1, 2)$. In the original LLL algorithm, δ was set to $2/\sqrt{3}$ (Lenstra et al. 1982). In case that Lovász condition (8) is violated, Lenstra et al. (1982) suggested swapping $\mathbf{b}_i$ with $\mathbf{b}_{i-1}$, decreasing the index i to $(i - 1)$, and then continuing the Gram-Schmidt orthogonalization process (7). For more details of algorithmic implementation of the LLL reduction method, the reader may refer to Lenstra et al. (1982); Daudé and Vallée (1994); Jaldén et al. (2008); Nguyen and Stehlé (2009) and Vetter et al. (2009).

In the case of $m = n$ and with the extra assumption $\mathbf{b}_i \in \mathbb{Z}^m (i = 1, 2, \ldots, m)$, Lenstra et al. (1982) reported that the LLL reduction will finish in polynomial time with an estimated number of $O(m^2 \log b_{\max})$ iterations or $O(m^3 n \log b_{\max})$ arithmetic operations, where $b_{\max}$ is the maximum length of the lattice basis $\mathbf{B}$. Daudé and Vallée (1994) showed that an upper bound of number of iterations depended only on the relative length of the lattice basis, namely, $O(m^2 \log_\delta \{b^*_{\max}/b^*_{\min}\})$, where $b^*_{\max}$ and $b^*_{\min}$ are the maximum and minimum lengths of the orthogonalized basis $\mathbf{B}^*$. Jaldén et al. (2008) proposed using the condition number of $\mathbf{B}$ to replace $(b^*_{\max}/b^*_{\min})$. Substantial work has since been focused on improving the complexity of the LLL algorithm (see, e.g. Jaldén et al. 2008; Nguyen and Stehlé 2009; Vetter et al. 2009). Nguyen and Stehlé (2009) showed that the LLL complexity can be significantly improved by a factor of $(\log b_{\max})$ in terms of bit operations. In practical applications of the LLL algorithm, a precise number of arithmetic operations is impossible to know in advance, since it cannot be determined by the size of a problem. In fact, the number of arithmetic operations strongly depends on the flow counter i and Lovász condition (8) which decides i to either increase or to decrease by an increment of one. To avoid such an unpredictable feature, fixed complexity algorithms of approximate LLL reduction are also proposed by Vetter et al. (2009).

Before finishing this subsection, we should like to note that if the weight matrix $\mathbf{W}$ in (3) is not an identity matrix, namely, $\mathbf{W} \neq \mathbf{I}$, then reduction algorithms for lattice vectors should be applied to $\mathbf{W}^{1/2}\mathbf{B}$ instead of $\mathbf{B}$ itself. Otherwise, even if $\mathbf{B}_r$ is completely orthogonal, the transformed normal matrix $\mathbf{W}_r = \mathbf{B}_r^{\mathrm{T}}\mathbf{W}\mathbf{B}_r$ can be arbitrarily far from diagonal after the integer transformation (6b). In fact, premultiplying $\mathbf{B}$ by $\mathbf{W}^{1/2}$ is statistically equivalent to whitening the color noise of the measurements $\mathbf{y}$ in (1). Also if the number of data is much bigger than the rank of the lattice, namely, $n \gg m$, as in the case of GNSS precise positioning, then one should first compute $\mathbf{W}_f$ and then apply the LLL reduction algorithm to $\mathbf{W}_f^{1/2}$. By doing so, the number of data n in the complexity of the LLL algorithm can then be replaced by a much smaller number of lattice rank m. Reduction of lattice vectors must not be directly applied to $\mathbf{W}_f$, however; otherwise, the almost orthogonality of $\mathbf{B}_r$ in $\mathbf{W}_f = \mathbf{B}_r\mathbf{G}^{-1}$ is simply useless for solving the ILS problem (4), because the integer transformation (6b) cannot be simultaneously done without destroying the symmetry of the elegant weighted integer least squares problem. After the LLL reduction and by substituting $\mathbf{W}_f^{1/2} = \mathbf{B}_r\mathbf{G}^{-1}$ into (3) or (4), together with the integer transformation (6b), we obtain the LLL-reduced positive definite matrix $\mathbf{W}_L = \mathbf{B}_r^{\mathrm{T}}\mathbf{B}_r$, where the subscript L of $\mathbf{W}_L$ is designated to emphasize that $\mathbf{W}_L$ is obtained using the LLL algorithm.

2.2 Parallel Cholesky-based reduction of positive definite quadratic forms

Although the integer linear model (1) defines the noise-contaminated lattice, the ILS or ML solution of $\mathbf{z}$ solely depends on the ILS or the weighted closest point problem (3), given a particular set of measurements $\mathbf{y}$. Thus, instead of applying lattice reduction to the lattice matrix $\mathbf{B}$, we should try to directly apply reduction techniques to (3). More specifically, we should directly reduce the positive definite matrix $\mathbf{W}_f$ of (4). Advantages of directly working on the reduction of positive definite quadratic forms are obvious: (i) since the dimension of $\mathbf{W}_f$ can be much smaller than the number of the measurements $\mathbf{y}$, at least, as in the case of GNSS precise positioning, computational complexity of reduction should accordingly decrease significantly; (ii) the weight matrix $\mathbf{W}$ of $\mathbf{y}$ has been naturally part of $\mathbf{W}_f$. No extra care is required about $\mathbf{W}$, as in the case of $\mathbf{W}^{1/2}\mathbf{B}$; and (iii) as is well-known, the ILS estimation of $\mathbf{z}$ has nothing to do with the scaling of $\mathbf{W}_f$, given the real-valued floating solution $\mathbf{z}_f$ (Xu 2006).

Reduction of positive definite quadratic forms is to find a unimodular matrix $\mathbf{G}$ such that the newly transformed positive definite matrix $\mathbf{W}_N = \mathbf{G}^{\mathrm{T}}\mathbf{W}_f\mathbf{G}$ possesses certain desirable properties of optimality, for example, in the sense of a minimum condition number or a minimum sphere covering in the equivalent class of positive definite quadratic forms of $\mathbf{W}_f$. Actually, almost all reduction methods for positive definite quadratic forms focus on the reduction domain or cone defined by the corresponding positive definite matrix $\mathbf{W}_f$ (see, e.g. Mahler 1938; Dickson 1972; Tammela 1976; Tammela 1979, 1985; Ryshkov 1976; Gruber and Lekkerkerker 1987). For example, given a real-valued positive definite matrix $\mathbf{W}_f$ and an integer vector $\mathbf{z}$, the positive definite quadratic form

$$f(\mathbf{z}) = \mathbf{z}^{\mathrm{T}}\mathbf{W}_f\mathbf{z} \tag{9}$$

is called *Minkowski-reduced* (see, e.g. Mahler 1938; Tammela 1976; Tammela 1979; Gruber and Lekkerkerker 1987), if the following inequalities

$$f(\mathbf{z}) \geq w_{ii}^f \tag{10a}$$

hold true for any i, given all integer vectors $\mathbf{z}$ and with the greatest common divisor of integer elements $z_i, z_{i+1}, \ldots, z_m$ of $\mathbf{z}$ being equal to unity, namely,

$$g.c.d.\ (z_i, z_{i+1}, \ldots, z_m) = 1, \tag{10b}$$

where w_{ii}^f of (10a) is the ith diagonal element of $\mathbf{W}_f$ and $g.c.d.(\cdot)$ in (10b) stands for greatest common divisor.

In particular, as part of a great number of inequalities defined by (10a) in the sense of Minkowski's reduction, we have

$$w_{ii}^f \leq w_{jj}^f, \quad (i < j) \tag{11a}$$

and

$$|w_{ij}^f| \le w_{jj}^f/2, \quad (i \ne j). \tag{11b}$$

For more details of deriving the set of inequalities (11), the reader can refer to Gruber and Lekkerkerker (1987).

As the second example of reduction, a positive definite quadratic form can be said to be type-II Voronoi-reduced, if the quadratic form (9) satisfies the following inequality:

$$V_0 = \{\mathbf{x} \mid f(\mathbf{x}) \le f(\mathbf{x}-\mathbf{z}),\ \mathbf{z} \in \mathbb{Z}^m\}, \tag{12}$$

(see, e.g. Dickson 1972). In fact, the subset V_0 of (12) defines exactly the Voronoi cell in association with the ILS problem (3) (Xu 2006). Although the Voronoi theory of reduction is to partition the space $\mathbb{R}^m$, it is very important for the determination of the lattice sphere covering (Dickson 1968). One may find the unimodular matrix $\mathbf{G}$ such that the radius of sphere covering is minimized in the equivalent class of Voronoi cells defined by $\mathbf{W}_f$.

On the other hand, Seysen (1993) defined a new concept of so-called S-reduction for positive definite quadratic forms. A positive definite matrix $\mathbf{W}_f$ is said to be S-reduced, if the following inequality

$$S(\mathbf{W}_f) \le S(\mathbf{G}^{\mathrm{T}}\mathbf{W}_f\mathbf{G}) \tag{13}$$

holds true for all unimodular matrices $\mathbf{G}$, where the function $S(\mathbf{W}_f)$ is defined by

$$S(\mathbf{W}_f) = \sum_{i=1}^{m} w_{ii}^f \widetilde{w}_{ii}^f, \tag{14}$$

with $\widetilde{w}_{ii}^f$ being the ith diagonal element of the inverse of $\mathbf{W}_f$. The S-reduction method is to find a unimodular matrix $\mathbf{G}$ such that the function $S(\mathbf{G}^{\mathrm{T}}\mathbf{W}_f\mathbf{G})$ is minimized. Intuitively, one may alternatively minimize the condition number of $(\mathbf{G}^{\mathrm{T}}\mathbf{W}_f\mathbf{G})$ to construct the optimal unimodular matrix $\mathbf{G}$.

Since the objective function $S(\mathbf{G}^{\mathrm{T}}\mathbf{W}_f\mathbf{G})$ is highly nonlinear with respect to the elements of the unimodular matrix $\mathbf{G}$, finding the global optimal integer solution $\mathbf{G}$ to minimize $S(\mathbf{G}^{\mathrm{T}}\mathbf{W}_f\mathbf{G})$ subject to the unimodularity constraint can be difficult and time-consuming. As a result, a local optimization algorithm was developed by Seysen (1993) by confining $\mathbf{G}$ to a matrix of type $(\mathbf{I} - g_{ij}\mathbf{E}_{ij})$ $(i \ne j)$, where g_{ij} is the only unknown integer to be determined by minimizing Seysen's reduction objective function, $\mathbf{E}_{ij}$ is a zero matrix except for the unity element at row i and column j, namely, $e_{ij} = 1$. This local algorithm was called S_2-reduction. However, the simulation results of LaMacchia (1991) conclude: (i) that it can be an effective reduction method only for problems with a low dimension up to 35 and (ii) that it cannot compete with the LLL algorithm. For these reasons, it will not be included for comparison in this paper.

A good unimodular matrix $\mathbf{G}$, however, is only a means to but surely not the goal of solving the closest point problem (3). From this point of view, it is not desirable to spend too much time in solving a highly nonlinear integer optimization problem to attain the global optimal solution $\mathbf{G}$ in a certain sense, not to mention that such a sense of optimality might not be necessary to help quickly find the optimal solution to (3). Therefore, as in the case of LLL reduction algorithms for lattice vectors, it is more feasible to search for methods to reduce positive definite quadratic forms, which should work effectively and should be less complex computationally.

In the remainder of this section, we will develop a parallel reduction algorithm for positive definite quadratic forms to satisfy part of Minkowski's constraint. More specifically, we are only concerned with the Minkowski's constraints (11a) and (11b). Thus, we should obtain such a $\mathbf{G}$ much more easily than Minkowski's optimal unimodular matrix. In the two-dimensional case, the algorithm is equivalent to the Gaussian reduction (see, e.g. Vallée 1991).

Starting with the Cholesky decomposition of $\mathbf{W}_f$:

$$\mathbf{W}_f = \mathbf{L}\mathbf{D}\mathbf{L}^{\mathrm{T}}, \tag{15}$$

the parallel Cholesky-based reduction algorithm to be developed in this paper consists of two basic components: (i) the size reduction of the matrix $\mathbf{L}$, and (ii) using different strategies of sorting the diagonal elements of $\mathbf{W}_f$ in parallel for achieving best results of reduction. Here $\mathbf{D}$ is diagonal with all its elements being positive and $\mathbf{L}$ is lower-triangular, namely,

$$\mathbf{L} = \begin{bmatrix} 1 & & & & \\ l_{21} & 1 & & & \\ l_{31} & l_{32} & 1 & & \\ \vdots & \vdots & \vdots & \ddots & \\ l_{m1} & l_{m2} & l_{m3} & \dots & 1 \end{bmatrix}. \tag{16}$$

Before we present the Cholesky-based reduction algorithms for positive definite quadratic forms, we state the following proposition of size reduction.

Proposition 1 *For any real-valued lower-triangular matrix* $\mathbf{L}$ *of type* (16)*, there exists a unimodular matrix* $\mathbf{G}$ *such that*

$$\mathbf{L} = \mathbf{G}\mathbf{L}_\mu, \tag{17a}$$

where

$$\mathbf{L}_\mu = \begin{bmatrix} 1 & & & & \\ \mu_{21} & 1 & & & \\ \mu_{31} & \mu_{32} & 1 & & \\ \vdots & \vdots & \vdots & \ddots & \\ \mu_{m1} & \mu_{m2} & \mu_{m3} & \dots & 1 \end{bmatrix}, \tag{17b}$$

and all the elements μ_{ij} *satisfy*

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

The logic proof of Proposition 1 can be simply described in Algorithm 1. Note, however, that the elements of $\mathbf{G}$ are

computed from the integer $intL$ within the loop but are not generated explicitly for output. Algorithm 1 also replaces $\mathbf{L}$ with $\mathbf{L}_\mu$. Actually, when the LLL algorithm terminates it naturally produces the size-reduced matrix $\mathbf{L}_\mu$ (see, e.g. Daudé and Vallée 1994; Lenstra et al. 1982; Schnorr 1987). However, we note that the LLL algorithm dynamically size-reduces $\mathbf{L}$ partially up to a changing row i and with a changing sub-basis before its termination, depending on the Lovász condition. Our parallel reduction algorithm will repeatedly perform the size reduction of the full matrix $\mathbf{L}$ all the time. Inserting the integer transformation (17a) into (15) and ignoring the unimodular matrix $\mathbf{G}$, we obtain the reduced positive definite matrix of $\mathbf{W}_f$, which is denoted by $\mathbf{W}_N$ and given as follows,

$$\mathbf{W}_N = \mathbf{L}_\mu \mathbf{D} \mathbf{L}_\mu^{\mathrm{T}}. \tag{18}$$

Algorithm 1: Reducing $\mathbf{L}$ to $\mathbf{L}_\mu$

```
for i = m to 2 step -1
  for j = i - 1 to 1 step -1
    if |L(i, j)| > 0.5
      intL = nearest integer to L(i, j);
      L(i, j) = L(i, j) - intL;
      Update the elements L(i, 1 : j − 1) by computing
      L(i, 1 : j − 1) = L(i, 1 : j − 1) − intL ∗ L(j, 1 : j − 1);
    end
  end
end
```

Now we will focus on the second component of the parallel reduction algorithm, namely, the strategies of sorting the diagonal elements of $\mathbf{W}_f$ to achieve a maximum effect of size reduction. By comparing (16) with (17b), we may see that the elements $|l_{ij}|$ $(i < j)$ should be as large as possible in order to reduce the sizes of the (large) diagonal elements of $\mathbf{W}_f$ quickly. Two intuitively attractive sorting techniques to re-arrange the diagonal elements of $\mathbf{W}_f$ are: (i) to sort the diagonal elements in natural ascending order, namely, $w_{11}^f \le w_{22}^f \le \cdots \le w_{mm}^f$, and (ii) the complete pivoting by selecting the smallest diagonal element at each step in the process of constructing the matrix $\mathbf{L}$ of (16). The former is straightforward and easy to implement, and the latter can be summarized in Algorithm 2 for convenience of reference and implementation. Algorithm 2 has been shown to be very effective in constructing suboptimal (approximate) solutions to (1). It is now well-known as the sorted QR decomposition in the MIMO literature (see, e.g. Wübben et al. 2001, 2003, 2004; Ling and Mow 2009; Waters and Barry 2005a,b), although it was first presented by Xu et al. (1995) to reduce $\mathbf{W}_f$ by extending the Gaussian reduction to a general m-dimensional case and further to derive a suboptimal integer solution for GPS ambiguity resolution. Keeping in mind the popularity of the terminology of sorted QR decomposition, in this paper, we will refer to these two sorting strategies as natural ascending and sorted QR orderings, respectively.

Algorithm 2: Cholesky-decomposition with complete minimum pivoting

```
Set L to an identity matrix;
for i = 1 to m − 1
  get the smallest w_kk^f among w_jj^f (i ≤ j ≤ m);
  if i ≠ k
    Swap L(i, 1 : i − 1) with L(k, 1 : i − 1);
    Swap the elements of submatrix W_f(i : m, i : m),
      both at the ith and kth row and column;
  end
  Compute L(i + 1 : m, i) = W_f(i + 1 : m, i)/w_kk^f;
  D(i, i) = w_kk^f;
  Update W_f(i + 1 : m, i + 1 : m) with L(i + 1 : m, i) and w_kk^f;
end
D(m, m) = W_f(m, m).
```

In addition, we have also explored the feasibility of using the information on the diagonal elements of $\mathbf{D}$ to construct an alternative sorting strategy for reduction. Simulations have shown that re-arranging the diagonal elements of $\mathbf{W}_f$ according to the ascending order of the diagonal elements of $\mathbf{D}$ cannot be used as an independent sorting strategy. More on this will be discussed in the next section on the experiments. However, such information can be extremely useful to work together with the natural ascending sorting as a perturbation technique. More precisely, after the first iteration with the natural ascending sorting of $\mathbf{W}_f$, we switch to the ascending ordering of $\mathbf{D}$ to re-arrange the matrix $\mathbf{W}_f$ in the second and third iterations of reduction. Starting from the fourth iteration, we return to the natural ascending sorting of $\mathbf{W}_f$ until the algorithm terminates. This process will be referred to as the (third) perturbed ascending sorting strategy.

By treating $\mathbf{W}_N$ as $\mathbf{W}_f$ and repeating the above procedures of reduction with different sorting strategies until $|l_{ij}| \le 0.5$, $(i < j)$ for all the elements of $\mathbf{L}$ in (16), we can finally attain the reduced positive definite matrix $\mathbf{W}_N$, with part of the Minkowski's constraints (11a) and (11b) automatically satisfied.

We are now in a position to assemble the results of Proposition 1, Algorithm 1 and the three sorting strategies together for a test parallel Cholesky-based reduction algorithm for positive definite quadratic forms. Focusing on the final reduced positive definite matrix without the care of permutations, the test algorithm can be described in Algorithm 3. Since we will only compare the performances of reduction by the proposed parallel Cholesky-based and LLL reduction methods, it is not necessary to keep the information on the permutations and $\mathbf{G}$ in this study, as is in the case of LLL reduction. However, we should note that recording the permutations and the final unimodular matrix $\mathbf{G}$ is important in correctly reporting the solution to the ILS problem (3). Algorithm 3 is said to be parallel, since each of the three sorting techniques, as described in Steps 1A, 1B and 1C, can be run independently in parallel. Finally, we should note that

although each of the components of the parallel reduction algorithm might be said to be known, the parallel reduction algorithm by putting them together will be shown in Sect. 4.3 to be very successful and perform much better than any individual component.

Algorithm 3: Parallel Cholesky-based reduction of $\mathbf{W}_f$

Given a positive definite matrix $\mathbf{W}_f$;
Step 1A: Apply the natural ascending sorting for $\mathbf{LDL}^{\mathrm{T}}$; or
Step 1B: Apply Algorithm 2 for $\mathbf{LDL}^{\mathrm{T}}$; or
Step 1C: Apply the perturbed ascending sorting for $\mathbf{LDL}^{\mathrm{T}}$;
Step 2: **if** $|L(i, j)| \leq 0.5$ for all $(1 \leq j < i \leq m)$
Step 3: Terminate;
Step 4: **end**
Step 5: Apply Algorithm 1 to get $\mathbf{L}_\mu$;
Step 6: Overwrite $\mathbf{W}_f = \mathbf{L}_\mu \mathbf{D} \mathbf{L}_\mu^{\mathrm{T}}$;
Step 7: Goto Step 1.
Output the final reduced matrix $\mathbf{W}_f$;

2.3 Fixed complexity for the Cholesky-based reduction method

According to Hadamard's inequality, namely, $\det\{\mathbf{W}_f\} \leq \prod_{i=1}^{m} w_{ii}^f$, it is trivial to prove that the Cholesky-based reduction method with any of the three sorting strategies converges, where $\det\{\cdot\}$ stands for determinant. However, given a particular problem in hand, we do not know exactly when Algorithm 3 terminates. A fixed complexity version can be obtained by setting a maximum number of iterations before Step 1 to artificially terminate Algorithm 3. As is well-known, each iteration of Algorithm 3 with one of the sorting strategies will approximately take $m^3/3$ arithmetic operations to complete the Cholesky decomposition $\mathbf{LDL}^{\mathrm{T}}$ and about the same arithmetic operations to reconstruct $\mathbf{W}_f = \mathbf{L}_\mu \mathbf{D} \mathbf{L}_\mu^{\mathrm{T}}$ for the next iteration. Reduction of $\mathbf{L}$ could require about the same number of operations in the beginning in the worst case. Thus if we set the maximum number of iterations between m and $3m$, we have a new version of Algorithm 3 with a fixed complexity of $O(m^4)$ arithmetic operations. Algorithm 3 can terminate naturally as a result of Steps 2 to 4 before the maximum number of iterations is hit, however. More importantly, the simulations to be reported in Sect. 4.3 have shown that the probability that the sorted QR ordering further improves the condition numbers from the combination of the natural and perturbed ascending sorting techniques by more than 50% is very small, ranging from 0.0006 in the third experiment to 0.0037 in the first experiment. Thus in the final codes of the parallel algorithm, we implement $3m$ as the maximum number of iterations without the sorted QR ordering of Step 1B. All the results to be reported in Sect. 4.3 are based on this final fixed complexity version of the parallel Cholesky-based reduction algorithm.

3 Measures of reduction quality for performance comparison

As is well-known, reduction methods are expected to achieve two goals in an idealized situation: (i) the reduced lattice vectors are the shortest; and (ii) they are mutually orthogonal. However, in reality, given a set of independently linear lattice vectors $\mathbf{B}$, the idealized situation is either not expected for the closest point problem due to the hardness of computing the shortest vector or generally impossible to attain the complete mutual orthogonality among the reduced lattice vectors. There are two popular measures to evaluate the quality of reduced lattice vectors for a reduction method (see, e.g. Vetter et al. 2009; Kannan 1987; Akhavi 2002; Schnorr 2006): one to measure length defects and the other to measure orthogonality defects.

A length defect of a lattice basis can be defined by $\|\mathbf{b}_1\|/r_1$ (see, e.g. Akhavi 2002; Schnorr 2006), where r_1 is the length of the shortest non-zero lattice vector. The most widely used measure of orthogonality defects is based on Hadamard's inequality. Given a lattice basis $\mathbf{B}$, we always have

$$d(\mathcal{L}) \leq \prod_{i=1}^{m} \|\mathbf{b}_i\|, \tag{19}$$

where

$$d(\mathcal{L}) = \sqrt{\det\{\mathbf{B}^{\mathrm{T}}\mathbf{B}\}}$$

is invariant with respect to bases of the lattice $\mathcal{L}$. Both sides of Hadamard's inequality (19) will become identical, if and only if the basis vectors of the lattice are mutually orthogonal. As a result, a natural quality measure of reduction to describe orthogonality defects can be defined as follows:

$$\mathcal{O}(\mathcal{L}) = \prod_{i=1}^{m} \|\mathbf{b}_i\|/d(\mathcal{L}), \tag{20}$$

(see, e.g. Kannan 1987; Akhavi 2002; Vetter et al. 2009).

However, usefulness of these two measures of reduction quality in association with the ILS problem (3) seems questionable for two reasons: (i) since our major target is to solve (3), we should avoid spending too much time in solving the shortest vector problem of lattice, which is conjectured to be NP-hard (see, e.g, Conway and Sloane 1999; Grötschel et al. 1988); and (ii) near orthogonality of lattice vectors does not directly provide any useful information on the constrained ellipsoid or sphere in which the solution of (3) is enclosed. Even worse, the shape of the Voronoi cell associated with (3) can be very sensitive to small changes in near orthogonality, as clearly shown by Xu (2006).

As a result, Xu (2001) used the concept of condition numbers to measure the quality of reduction methods. For the noise-contaminated lattice model (1) with a weight matrix $\mathbf{W}$, we can compute the condition number of $\mathbf{W}_f$, which is

denoted by κ_B and given as follows:

$$\kappa_B = \lambda_{\max}/\lambda_{\min}, \tag{21}$$

where $\lambda_{\max}$ and $\lambda_{\min}$ are the maximum and minimum eigenvalues of $\mathbf{W}_f (= \mathbf{B}^{\mathrm{T}}\mathbf{W}\mathbf{B})$, respectively. Accordingly, with the reduced positive definite matrix $\mathbf{W}_N$ after applying a reduction method either to the lattice basis ($\mathbf{W}^{1/2}\mathbf{B}$) or to the positive definite matrix $\mathbf{W}_f$, we can also compute the condition number of $\mathbf{W}_N$ as follows:

$$\kappa_N = \lambda^N_{\max}/\lambda^N_{\min}, \tag{22}$$

where $\lambda^N_{\max}$ and $\lambda^N_{\min}$ are the maximum and minimum eigenvalues of $\mathbf{W}_N$, respectively. Note, however, that if $\mathbf{W}_f$ is diagonal, or equivalently, if the basis vectors of the lattice are already mutually orthogonal, then no methods of reduction can reduce the condition number of $\mathbf{W}_f$. Fortunately, in this idealized case, it is trivial to find the exact solution to the ILS problem (4) (Taha 1975) without the need of reduction.

Geometrically, a condition number represents the most important information on the shape of the ellipsoid to contain the solution of the ILS problem (3) and can significantly affect the efficiency of finding the solution. Ideally, if $\kappa_N = 1$, then the reduced positive definite matrix $\mathbf{W}_N$ completely defines a sphere, and the solution to (3) is trivial to obtain. For a general problem ($\kappa_N \neq 1$), if one would directly search for the global optimal integer solution within an ellipsoid, one can readily use the combination of algorithms developed by Fincke and Pohst (1985) and Schnorr and Euchner (1994).

To illustrate how a condition number affects the time of searching for the global optimal integer solution, let us take the searching within a sphere (or sphere searching) as an example. Assuming that an ellipsoid defined by $\mathbf{W}_f$ requires an initial length of r_f in the minor axis to contain, at least, one integer point inside the ellipsoid, then we will have to search for the solution within a sphere with the minimum radius of $r_f \kappa_B^{1/2}$, where r_f can either be obtained using Babai's (1986) technique or simply derived from the ellipsoidal equation $\mathbf{x}^{\mathrm{T}}\mathbf{W}_f\mathbf{x} = c_f$ by setting $\mathbf{z}_f$ to its nearest integer, namely, $c_f = \mathrm{F}(\lceil \mathbf{z}_f \rfloor)$. After reduction, the minimum radius of the corresponding searching sphere is then equal to $r_N \kappa_N^{1/2}$, where r_N can be readily found in the similar manner to r_f. In general, rounding the real-valued solution to its nearest integer after reduction should be more reliable and, as a result, r_N should be closer to the shortest length for the ellipsoid to contain, at least, one integer point. A condition number obviously reflects directly the complexity of finding the solution to (3) in sphere searching. The smaller the condition number κ_N, the better the corresponding reduction method. Geometrically, a good reduction method should turn a (badly-shaped) ellipsoid into a sphere as much as possible, as illustrated in Fig. 1. As a result, it is more advantageous to use condition numbers to measure the quality of reduction in association with the weighted closest point problem (3) than the measures of length and orthogonality defects. More examples and consequences to support the use of condition numbers can be found, e.g. in Xu (2001); Artés et al. (2003) and Luk and Tracy (2008). Thus, in this contribution, we will compare different reduction methods on the basis of condition numbers.

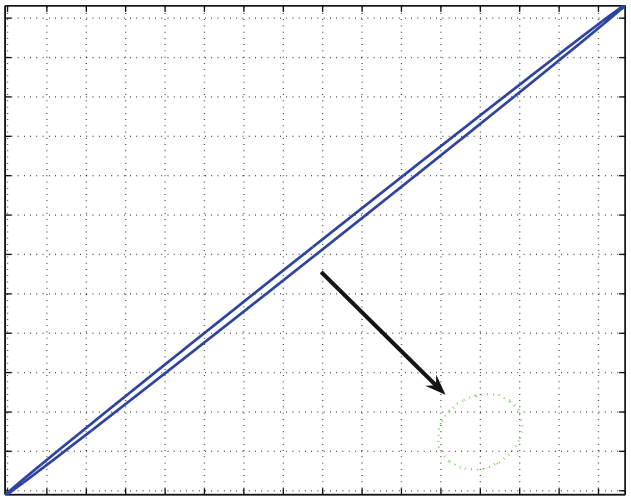

Fig. 1 Illustration of successful reduction: *blue solid line*—the original ILS problem corresponding to a badly shaped ellipsoid; *green dotted line*—the shape of ellipsoid after a successful reduction. The centers of the ellipsoids have been shifted for a clear display

4 Experiments and simulation results

4.1 Experiment design

Experiment examples in this section should serve three purposes: (i) to investigate the effect of sorting on reduction; (ii) to demonstrate the parallel Cholesky-based reduction method; and (iii) to compare the performances of the LLL algorithm and the parallel Cholesky-based reduction method. In order to avoid the effect of a particular example on performance comparison, we will have to simulate a great number of random examples. To begin with the experiments, we need to simulate the design matrix $\mathbf{B}$ and the weight matrix $\mathbf{W}$. Since the weight matrix $\mathbf{W}$ has either been incorporated into the lattice vectors $\mathbf{W}^{1/2}\mathbf{B}$ in the case of the LLL algorithm or the normal matrix $\mathbf{W}_f$ in the case of reduction of positive definite quadratic forms, without loss of generality, we can set $\mathbf{W} = \mathbf{I}$ in our simulations. Thus we will focus on randomly designing the matrix $\mathbf{B}$ such that the noise-contaminated integer linear model (1) could reflect a wide range of problems. More specifically, the basic ingredients of designing $\mathbf{B}$ include the number of integer unknowns and the condition number of the matrix $\mathbf{W}_f$. Additionally, we

should also allow the number of measurements **y** to vary over a fairly wide range.

As a result of all the above considerations, we design three scenarios of experiments, each with 10,000 random examples, as follows:

- the first set of examples is randomly generated, with the number of integer unknowns **z** uniformly varying between 3 and 20 and the number of measurements **y** uniformly between 200 and 1,000. Each random matrix **B** is first generated using the standard Gaussian normal distribution $N(0, 1)$ and decomposed using the singular value decomposition technique. Then the singular values of **B** are replaced by the positive random numbers from a uniform distribution to reconstruct **B** such that the condition number of $\mathbf{W}_f$ randomly falls into [100, 40,000]. Roughly speaking, this set of examples may be compared to kinematic positioning with all satellite navigation systems such as American GPS, European Galileo and Chinese COMPASS in operation. The maximum number of iterations is set to $3m$ to terminate the parallel reduction algorithm, if it does not stop naturally;
- the second set of examples is designed with the same setting as the first set of examples, except for the number of integer unknowns **z** to uniformly distribute between 21 and 50; and
- finally, the last set of examples is to simulate highly ill-conditioned problems. This set of examples is motivated by the reported bad consequences of ill-conditioning on decoding by Artés et al. (2003) (see also Xu et al. 1999) and used to demonstrate how the issue of ill-conditioning can be overcome by lattice reduction of vectors and/or positive definite quadratic forms. As in the second set of examples, the dimension of **z** for this set of examples is between 21 and 50. However, the numbers of measurements range from 21 to 1,000 under the condition of $n \geq m$, and the condition numbers fall randomly between 10^4 and (1.5×10^9).

4.2 The effect of sorting on reduction

Sorting has been shown to play an important role in finding an improved suboptimal solution to the integer unknowns **z** of the integer linear model (1). The basic idea is to optimally design an ordering of integer parameters for a sequential estimation of each integer. The two most widely used techniques of sorting are the sorted QR decomposition (see, e.g. Xu et al. 1995; Wübben et al. 2001) and the BLAST ordering (see, e.g. Waters and Barry 2005b). The former can either be applied directly to $\mathbf{W}_f$ (Xu et al. 1995) or to the geometrical structure of the system (1), namely, the **B** matrix of (1) (see, e.g. Ling and Mow 2009; Wübben et al. 2001), while the latter is essentially equivalent to finding the optimal index/ordering such that the corresponding real-valued, sequential estimate of each integer unknown has the minimum conditional variance (see, e.g. Waters and Barry 2005b).

Unlike suboptimal integer estimation techniques which try to maximally use the most precise (conditional) information at each stage to sequentially estimate integers, lattice reduction attempts to whiten the colored noise of the (real-valued) floating solution $\mathbf{z}_f$ and thus indirectly increases the conditional probability of sequential integer estimation. Waters and Barry (2005a) showed that a simple implementation of lattice reduction can significantly improve the performance of a suboptimal estimator/detector. The effect of sorting is less obvious for lattice reduction than for suboptimal integer estimation, however.

Thus our major question of concern is how sorting would affect reduction. We investigate this question through numerical simulations. More precisely, we will compare the effects of the three sorting strategies in Sect. 2.2 on reduction, namely, (i) the natural ascending sorting; (ii) the sorted QR ordering or the Cholesky-decomposition with complete pivoting detailed in Algorithm 2; and (iii) the perturbed ascending sorting. For convenience of comparison, we will use the abbreviations *ASCE*, *SEQR* and *PERT* to denote these three sorting strategies, respectively.

With the implementation of the sorting strategies *ASCE*, *SEQR* and *PERT* in Algorithm 3, we can then compute and obtain the condition numbers after reduction for all the random examples in each of the three experiments. In the case of the first experiment with the 10,000 low-dimensional random examples, for instance, we have 10,000 condition numbers for each of the sorting strategies *ASCE*, *SEQR* and *PERT*, which are denoted by the vectors $\boldsymbol{\kappa}_{\mathrm{ACE}}, \boldsymbol{\kappa}_{\mathrm{SQR}}$ and $\boldsymbol{\kappa}_{\mathrm{PRT}}$, namely,

$$\boldsymbol{\kappa}_{\mathrm{ACE}} = (\kappa^1_{\mathrm{ACE}}, \kappa^2_{\mathrm{ACE}}, \ldots, \kappa^{10,000}_{\mathrm{ACE}}), \tag{23a}$$

$$\boldsymbol{\kappa}_{\mathrm{SQR}} = (\kappa^1_{\mathrm{SQR}}, \kappa^2_{\mathrm{SQR}}, \ldots, \kappa^{10,000}_{\mathrm{SQR}}), \tag{23b}$$

and

$$\boldsymbol{\kappa}_{\mathrm{PRT}} = (\kappa^1_{\mathrm{PRT}}, \kappa^2_{\mathrm{PRT}}, \ldots \kappa^{10,000}_{\mathrm{PRT}}), \tag{23c}$$

respectively. If $\kappa^i_J < \kappa^i_K$, we say that strategy J performs better than strategy K for the ith example, where the subscripts J and K stand for one of the strategies *ASCE*, *SEQR* and *PERT*. As a result, when comparing strategy J with strategy K with the first experiment of 10,000 examples, we can count the total numbers of examples either satisfying $\kappa^i_J < \kappa^i_K$ or $\kappa^i_J > \kappa^i_K$, which are denoted by n_b and n_w, respectively. Accordingly, we can readily compute the percentage of examples with which strategy J performs better than strategy K by $n_b/10{,}000 \times 100$ and the percentage of examples with which strategy K performs better than strategy J by $n_w/10{,}000 \times 100$, respectively. In the same manner, we can obtain the same statistics for the

Table 1 Performance statistics of the effects of sorting strategies *ASCE*, *SEQR* and *PERT* on reduction

Experiment	Methods	ASCE_SEQR	ASCE_PERT	PERT_SEQR
1	BExamples	40.52	20.04	41.21
	WExamples	27.21	21.63	26.64
2	BExamples	79.91	48.52	80.03
	WExamples	20.08	47.06	19.96
3	BExamples	85.88	49.69	86.10
	WExamples	14.12	50.31	13.90

Experiments 1, 2 and 3 are referred to the first, second and third sets of random examples, respectively. BExamples—number of examples with an improved smaller condition number (in %); WExamples—number of examples with a deteriorated larger condition number (in %)

second and third experiments of 10,000 examples. All the statistics are listed in Table 1. It is obvious from columns ASCE_SEQR and PERT_SEQR of Table 1 that the sorting strategies *ASCE* and *PERT* consistently perform much better than the strategy *SEQR* for all the three experiments. In particular, in the second and third experiments, the sorting strategies *ASCE* and *PERT* are able to produce a smaller condition number than the sorting strategy *SEQR* for 79.91 to 86.10% of the examples. Nevertheless, the strategy *SEQR* still performs better than *ASCE* and *PERT* with examples from 13.90% in the third experiment to 27.21% in the first experiment. However, these numbers drop to a maximum of 0.95% in the first experiment, 1.63% in the second experiment and 1.14% in the third experiment, respectively, if we only count all the examples such that *SEQR* is able to improve the condition numbers from *ASCE* or *PERT* by 50%, namely, $\kappa_{\text{SQR}} \leq 0.5\kappa_{\text{ACE}}$ or $\kappa_{\text{SQR}} \leq 0.5\kappa_{\text{PRT}}$. Furthermore, if we assemble the sorting strategies *ASCE* and *PERT* to construct a parallel reduction algorithm, then the probability for $\kappa_{\text{SQR}} \leq 0.5\kappa_{\text{APT}}$ significantly drops to 0.0006 in the third experiment to 0.0037 in the first experiment, where κ_{APT} is the reduced condition number from the parallel reduction algorithm with the sorting techniques *ASCE* and *PERT*.

As for the sorting strategies *ASCE* and *PERT*, they perform about equally well in all the three experiments (compare column ASCE_PERT of Table 1). Since sorting by arranging the diagonal elements of $\mathbf{D}$ in (15) in ascending order as a perturbation component of *PERT* has been shown from the above simulations to produce satisfactory results of reduction, one may ask whether it is feasible to fully implement this sorting technique as an independent sorting strategy for reduction; this full version of sorting the diagonal elements of $\mathbf{D}$ will be referred to as *SRTD* in the remainder of this section. Indeed, we have tried to repeat all the test computations with *SRTD* in the three experiments. Among all the 10,000 examples in the first experiment, *SRTD* has performed slightly better than *ASCE*, *SEQR* and *PERT* by 4.74, 16.51 and 3.92 more percent of the examples, respectively. However, a deep analysis has shown that *SRTD* has a much bigger chance to produce a bigger condition number than *ASCE*, *SEQR* and *PERT*. Actually, the probabilities for $\kappa_{\text{STD}} \geq 1.5\kappa_{\text{ACE}}$, $\kappa_{\text{STD}} \geq 1.5\kappa_{\text{SQR}}$ and $\kappa_{\text{STD}} \geq 1.5\kappa_{\text{PRT}}$ are equal to 0.079, 0.050 and 0.082, while those for $\kappa_{\text{STD}} \leq 0.5\kappa_{\text{ACE}}$, $\kappa_{\text{STD}} \leq 0.5\kappa_{\text{SQR}}$ and $\kappa_{\text{STD}} \leq 0.5\kappa_{\text{PRT}}$ are merely 0.018, 0.046 and 0.017, respectively, where κ_{STD} is the condition number after reduction with the sorting *SRTD*. In the second experiment, *SRTD* has significantly worsened the condition numbers of 58.51% of the original problems, with a maximum deterioration of condition number by an order of magnitude 12.247. Even worse, this sorting strategy completely failed with the first example in the third experiment (dimension: 49 and condition number: 5.1352×10^6) at the iteration of 58, because the condition number at this intermediate stage of reduction becomes far too big such that the Cholesky decomposition breaks down. Because of its instability and poor performance, we do not implement this sorting technique as an independent sorting strategy for reduction.

To investigate the practical running behaviors of the three sorting strategies *ASCE*, *SEQR* and *PERT*, we have recorded their numbers of iterations for each example in all the three experiments, which are denoted by I_{ACE}, I_{SQR} and I_{PRT}, respectively. Thus for each experiment, we can compute the iteration indices, namely, $\alpha_{\text{ACE}} = I_{\text{ACE}}/I_{\max}$, $\alpha_{\text{SQR}} = I_{\text{SQR}}/I_{\max}$ and $\alpha_{\text{PRT}} = I_{\text{PRT}}/I_{\max}$, for each of the sorting strategies *ASCE*, *SEQR* and *PERT*, where $I_{\max}$ is the maximum number of iterations and is set to $3m$ in the simulations. Based on the 10,000 iteration indices α_{ACE}, α_{SQR} and α_{PRT} from each experiment, we estimate their cumulative distribution functions (cdf), which are plotted in Fig. 2. The reduction with any of the three sorting strategies *ASCE*, *SEQR* and *PERT* terminates in less than m iterations for all the 10,000 examples in the first experiment (compare panel a of Fig. 2). The sorting *SEQR* converges most quickly with a probability of 0.95. Although *SEQR* performs almost as well as *ASCE* for almost all the examples in the second experiment (compare the green and red lines in panel b of Fig. 2), it takes a significantly larger number of iterations to terminate, though with a very small probability, as shown by the long tail of the green line in panel b. Nevertheless, all the 10,000 examples converge in $2m$ iterations with *SEQR*. In the third experiment, *SEQR* runs the fastest with a probability slightly bigger than 0.5 but quickly turns out to become the slowest for about 30% of the examples. It does not terminate naturally but has to be stopped by the set maximum number of iterations $3m$ (compare the green line of panel c in Fig. 2) with a probability of 0.055. The perturbed sorting strategy *PERT* generally takes more iterations than *ASCE* to converge or terminate, as can be clearly seen from the red and black-dotted lines of panels a, b and c of Fig. 2).

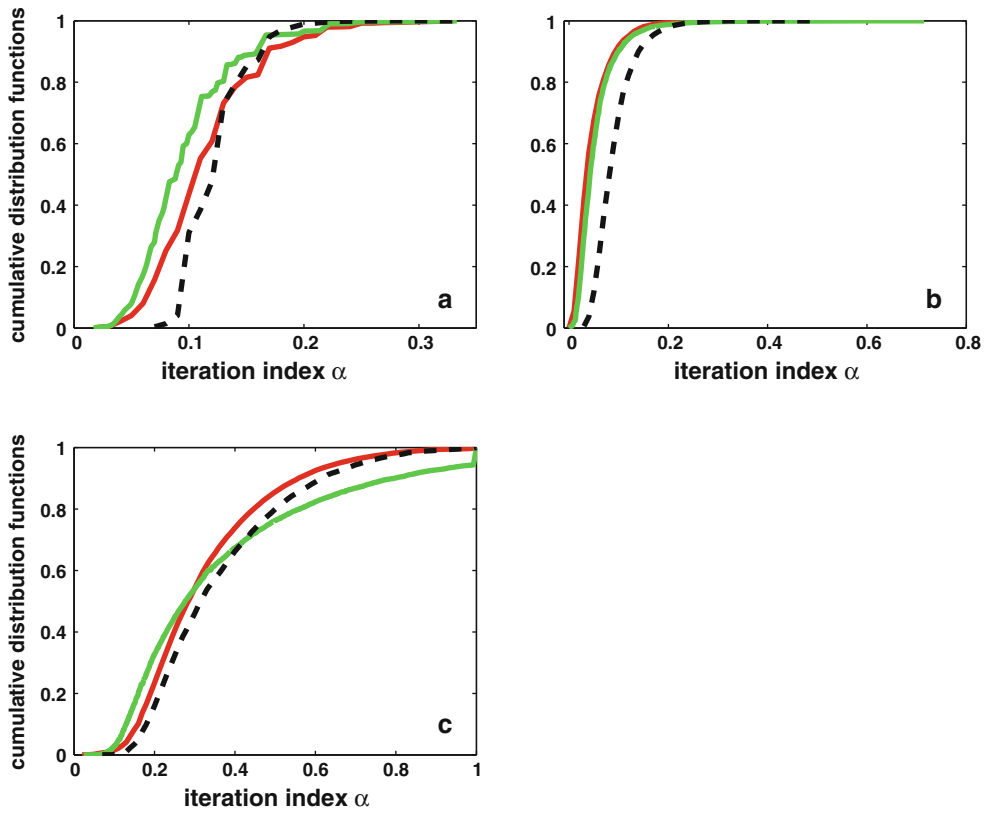

Fig. 2 The cumulative distribution functions of the iteration indices α_{ACE}, α_{SQR} and α_{PRT} computed from the 10,000 random examples for each of the three experiments using the sorting strategies *ASCE*, *SEQR* and *PERT*, respectively. panel **a**—the first experiment; panel **b**—the second experiment; panel **c**—the third experiment; *red solid line*—the sorting strategy *ASCE*; *green solid line*—the sorting strategy *SEQR*; and *black dashed line*—the sorting strategy *PERT*.

Table 2 Performance statistics of comparing the Cholesky-based reduction with one of the sorting strategies *ASCE, SEQR* and *PERT* with the LLL algorithm

Experiment	Methods	ASCE_LLL	SEQR_LLL	PERT_LLL
1	BExamples	54.21	48.27	54.46
	WExamples	25.98	31.69	25.55
2	BExamples	72.46	39.49	72.45
	WExamples	27.54	60.51	27.55
3	BExamples	98.38	87.44	98.43
	WExamples	1.62	12.56	1.57

All the statistical indices and notations shown in this table are the same as in Table 1

Now we will briefly compare Algorithm 3 with one of the three sorting techniques with the LLL algorithm. Exactly in the same manner as in Table 1, we list the comparative performance statistics in Table 2. Generally speaking, when compared with the LLL algorithm, Algorithm 3 with the sorting strategy *ASCE* or *PERT* performs much better than that with the sorting strategy *SEQR*, which is consistent with the above comparison among the three sorting strategies themselves. It is also clear from columns ASCE_LLL and PERT_LLL of Table 2 that they produce significantly better results than the LLL algorithm in all the three experiments. The LLL algorithm only wins both *ASCE* and *PERT* for about a quarter of examples in the first two experiments. The reduction with the sorting strategy *SEQR* is satisfactorily better than the LLL algorithm in the first experiment and overwhelmingly better in the third experiment. However, it is surprisingly much worse than the LLL algorithm in the second experiment, with 60.51% of the problems ending up with a bigger condition number.

4.3 Performance analysis of the parallel Cholesky-based reduction algorithm

Let $\mathbf{W}_f$, $\mathbf{W}_L$ and $\mathbf{W}_N$ be the original ILS positive definite matrix of (4) and the reduced positive definite matrices after applying the LLL algorithm and the parallel Cholesky-based reduction method with the sorting techniques *ASCE* and *PERT*, respectively. Their corresponding condition numbers are denoted by κ_B, κ_L and κ_N, respectively. With the 10,000 random examples in hand for each of the three experiments,

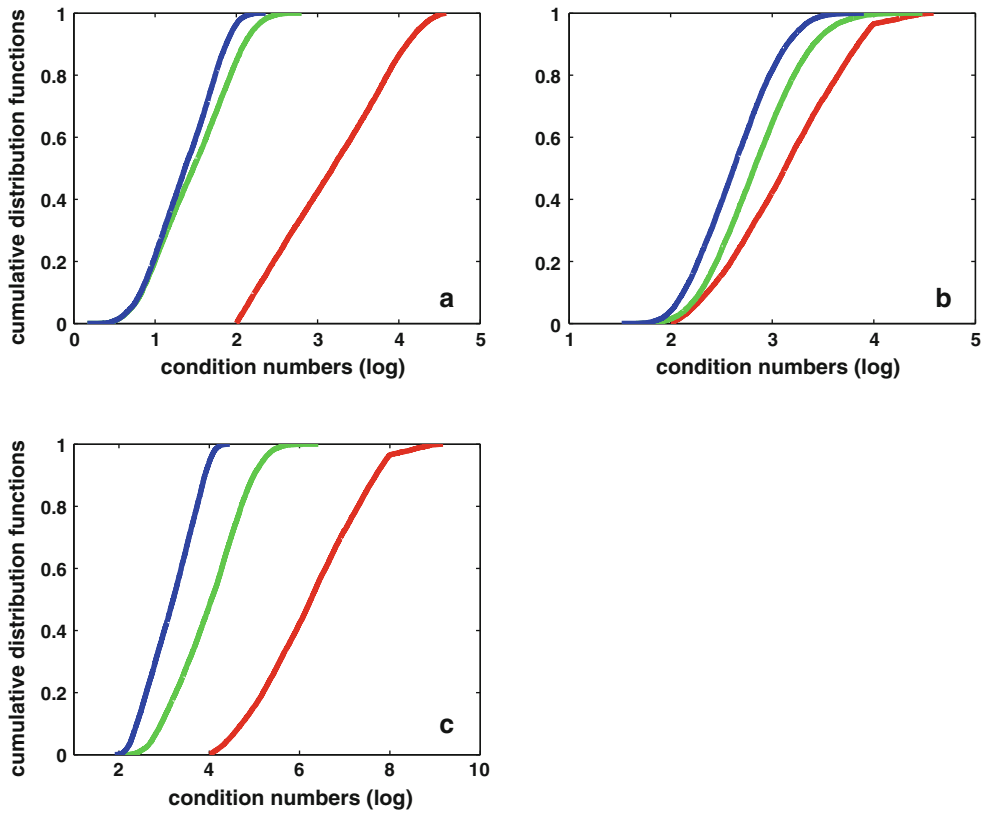

Fig. 3 The cumulative distribution functions of the condition numbers κ_N and κ_L after reduction, together with those of the problems κ_B, each cdf curve computed from the 10,000 random examples. The condition numbers are all shown in logarithm. Panels **a**, **b** and **c** correspond to the first, second and third experiments, respectively, with the *blue solid line*—κ_N after the Cholesky-based reduction, the *green line* —κ_L after the LLL reduction, and the *red line*—κ_B for the original ILS problems

we can accordingly obtain $10{,}000\kappa_B$ of $\mathbf{W}_f$, $10{,}000\kappa_L$ of $\mathbf{W}_L$ and $10{,}000\kappa_N$ of $\mathbf{W}_N$, respectively, namely,

$$\boldsymbol{\kappa}_B = (\kappa_B^1, \kappa_B^2, \ldots, \kappa_B^{10{,}000}), \tag{24a}$$

$$\boldsymbol{\kappa}_L = (\kappa_L^1, \kappa_L^2, \ldots, \kappa_L^{10{,}000}), \tag{24b}$$

and

$$\boldsymbol{\kappa}_N = (\kappa_N^1, \kappa_N^2, \ldots, \kappa_N^{10{,}000}), \tag{24c}$$

where the superscript i of each element of the vectors stands for the ith random example.

The cdf curves of κ_B, κ_L and κ_N are shown in Fig. 3. It is obvious from this figure that both the parallel reduction and the LLL algorithm are very successful in reducing the condition numbers of the examples in all the three experiments. The parallel reduction method has performed significantly better than the LLL algorithm. It enables to reduce the condition numbers of the original examples below 10^2 with a probability of 0.954 in the first experiment, below 2×10^3 with a probability of 0.955 in the second experiment, and below 10^4 with a probability of 0.941 in the third experiment, respectively. In the case of the LLL algorithm, the corresponding probabilities reduce to 0.846 in the first experiment, 0.849 in the second experiment, and 0.479 (about half of 0.941) in the third experiment, respectively.

Although the cdf curves of Fig. 3 are very informative to show the performance of the parallel reduction method and the LLL algorithm, they do not provide the information on direct comparison of the simulated random examples. In order to analyze the performance of the parallel Cholesky-based reduction algorithm against $\mathbf{W}_f$ in detail, we compute the differences of condition numbers between $\boldsymbol{\kappa}_N$ and $\boldsymbol{\kappa}_B$, which is denoted by $\delta\boldsymbol{\kappa}_{\mathrm{NW}}$ and given as follows:

$$\delta\boldsymbol{\kappa}_{\mathrm{NW}} = \log \boldsymbol{\kappa}_N - \log \boldsymbol{\kappa}_B, \tag{25}$$

where log stands for the operation of logarithm to base 10. We then divide all these 10,000 elements of $\delta\boldsymbol{\kappa}_{\mathrm{NW}}$ into two groups, depending on whether $\delta\kappa_{\mathrm{NW}}^i < 0$ or $\delta\kappa_{\mathrm{NW}}^i > 0$. The elements of the two groups are collected into the (sub)vectors $\delta\boldsymbol{\kappa}_{\mathrm{NW}}^1$ and $\delta\boldsymbol{\kappa}_{\mathrm{NW}}^2$, respectively. In other words, the parallel Cholesky-based reduction method is successful to reduce the condition numbers for all the examples in group 1 but worsens all the examples in group 2 by outputting even bigger condition numbers. Assuming that the numbers of elements in each of the two groups are equal to n_1 and n_2, we can compute

Table 3 Performance statistics of the parallel Cholesky-based reduction method and the LLL reduction algorithm from the first 10,000 random simulated examples in the first experiment with a low dimension up to 20

Methods	Chol_$\mathbf{W}_f$	LLL_$\mathbf{W}_f$	Chol_LLL
BExamples	99.94	99.46	59.49
WExamples	0.06	0.54	20.59
MeanImprove	−1.801	−1.719	−0.181
MaxImprove	−4.148	−4.148	−0.865
MeanWorsen	0.064	0.089	0.081
MaxWorsen	0.139	0.260	0.493

BExamples—number of examples with an improved (smaller) condition number (in %); WExamples—number of examples with a deteriorated (larger) condition number (in %); MeanImprove—mean improvement of condition numbers for the examples in group 1 (in logarithm); MaxImprove—maximum improvement of condition numbers for the examples in group 1 (in logarithm); MeanWorsen—mean deterioration of condition numbers for the examples in group 2 (in logarithm); MaxWorsen—maximum deterioration of condition numbers for the examples in group 2 (in logarithm)

the performance statistics: (i) $(n_1/10{,}000 \times 100)$ to show the percentage of examples with which the parallel Cholesky-based reduction method produces better results; and (ii) $(n_2/10{,}000 \times 100)$ to show the percentage of examples with which the parallel Cholesky-based reduction method worsens the condition numbers of the original problems, as in the case of Table 1. In addition, we further compute four more statistics: (iii) the mean value of $\delta\boldsymbol{\kappa}^1_{NW}$ for all the elements in group 1; (iv) the minimum value of $\delta\boldsymbol{\kappa}^1_{\mathrm{NW}}$ to show the maximum improvement in condition numbers; (v) the mean value of $\delta\boldsymbol{\kappa}^2_{\mathrm{NW}}$ to show the average deterioration of condition numbers after applying the parallel Cholesky-based reduction method; and finally, (vi) the maximum value of $\delta\boldsymbol{\kappa}^2_{\mathrm{NW}}$ to show the maximum deterioration of condition numbers. All these six performance indices are listed in column Chol_$\mathbf{W}_f$ of Table 3, respectively. Following the same procedure, we compute the same performance statistics and list them in column LLL_$\mathbf{W}_f$ of Table 3 to compare the LLL algorithm against $\mathbf{W}_f$ and in column Chol_LLL to compare the parallel Cholesky-based reduction method with the LLL algorithm, respectively.

It is clear from Table 3 that both the parallel Cholesky-based reduction method and the LLL algorithm are highly successful in reducing the condition numbers of $\mathbf{W}_f$ for almost all the examples in the first experiment with a low dimension up to 20, with an average improving factor of $10^{1.801}(= 63.241)$ for the former method and $10^{1.719}$ $(= 52.360)$ for the latter, respectively. In the best case, both methods can reduce the condition number of $\mathbf{W}_f$ by a factor of 14050.766 (compare row MaxImprove in Table 3). However, both methods have worsened the condition numbers of 6 (or 0.06%) and 54 (or 0.54%) examples, respectively. Nevertheless, the extent of deterioration is quite small for both methods, as can be clearly seen from the small mean values of the increased condition numbers (compare row MeanWorsen of Table 3). To give the reader a clear impression on the significant improvements of both methods, we plot the estimated probability density functions (pdf) of $(\log\kappa_N - \log\kappa_B)$, $(\log\kappa_L - \log\kappa_B)$ and $(\log\kappa_N - \log\kappa_L)$ in Fig. 4. Both the horizontal axes of panels a and b basically take negative values, indicating that the condition numbers of the examples have been successfully reduced by the parallel reduction method and the LLL algorithm.

It is also clear from the last column, namely, column Chol_LLL of Table 3, that the parallel Cholesky-based reduction method generally performs significantly better than the LLL algorithm. The conclusion can also be immediately confirmed after a look at the pdf of $(\log\kappa_N - \log\kappa_L)$ in panel c of Fig. 4. Statistically, of these 10,000 examples in the first experiment with a low dimension up to 20, the parallel Cholesky-based reduction method performs better than the LLL algorithm with 59.49% of the examples, with a maximum improvement factor of 7.32 in condition numbers. However, the opposite is true only with 20.59% of the examples, about one third of the former. If we confine ourselves to the examples satisfying $\kappa_L \leq 0.5\kappa_N$, such a percentage of improvement drops to 0.25%.

By keeping the same experimental setting as in the first set of examples but increasing the dimensions of the problems to change uniformly between 21 and 50, we repeat the same experiment and obtain all the condition numbers κ_B, κ_L and κ_N for the second experiment of 10,000 random examples. The same performance indices as in the first experiment are computed and listed in Table 4 and the estimated probability density functions of $(\log\kappa_N - \log\kappa_B)$, $(\log\kappa_L - \log\kappa_B)$ and $(\log\kappa_N - \log\kappa_L)$ are shown in Fig. 5. Generally speaking, the parallel Cholesky-based reduction method and the LLL algorithm still perform quite satisfactorily for this second set of examples, as can be seen from Table 4 and panels a and b of Fig. 5. They have significantly reduced the condition numbers for 82.08 and 67.59% of the simulated examples, respectively. However, the mean and maximum improving factors in condition numbers are now only 4.22 and 368.98 for the parallel Cholesky-based reduction, and 3.44 and 206.06 for the LLL algorithm, respectively. These values are significantly smaller than those in the first experiment. With the increase of dimensions m, the number of examples whose condition numbers have been worsened has also increased significantly, up to 17.90% of the total number of examples in this experiment for the Cholesky-based method and 32.41% for the LLL algorithm (compare row WExamples of Table 4). On average, the condition numbers of the examples in group 2 have been deteriorated by 50.5 and 78.6% for both methods, respectively. The parallel Cholesky-based reduction method obviously performs significantly better than the LLL

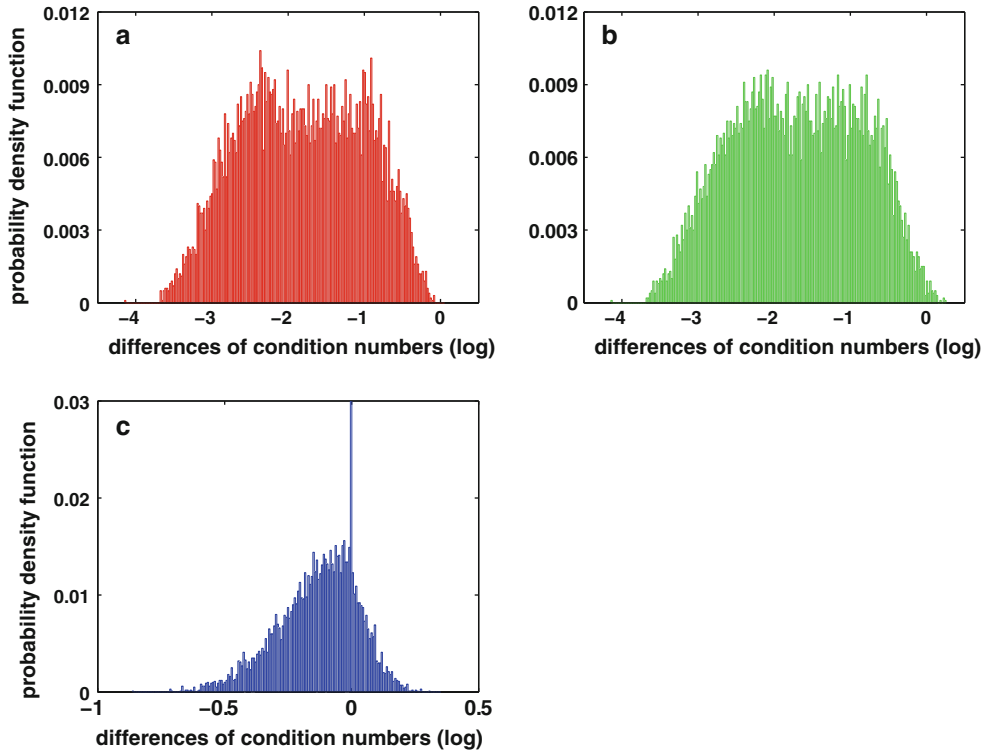

Fig. 4 The probability density functions of the differences of condition numbers for the first experiment, with a low dimension of **z** up to 20. panel **a**—$(\log \kappa_N - \log \kappa_B)$ to compare the parallel Cholesky-based reduction with the original problems, panel **b**—$(\log \kappa_L - \log \kappa_B)$ to compare the LLL reduction with the original problems, and panel **c**—$(\log \kappa_N - \log \kappa_L)$ to compare the parallel Cholesky-based reduction with the LLL algorithm

Table 4 Performance statistics of the parallel Cholesky-based reduction method and the LLL reduction algorithm with the second experiment of 10,000 random simulated examples

Methods	Chol_$\mathbf{W}_f$	LLL_$\mathbf{W}_f$	Chol_LLL
BExamples	82.08	67.59	84.44
WExamples	17.90	32.41	15.56
MeanImprove	−0.625	−0.536	−0.255
MaxImprove	−2.567	−2.314	−1.252
MeanWorsen	0.178	0.252	0.094
MaxWorsen	0.836	1.177	0.479

The dimensions of **z** vary between 21 and 50. All the statistical indices and notations shown in this table are the same as in Table 3

algorithm, as can also be seen from the statistics listed in the last column of Table 4, which is consistent with the results in the first experiment. The probability for $\kappa_L \leq 0.5\kappa_N$ is only 0.004, while the probability for $\kappa_N \leq 0.5\kappa_L$ is 0.294. Actually, the better performance of the parallel Cholesky-based reduction over the LLL algorithm has also been clearly shown by the pdf of $(\log \kappa_N - \log \kappa_L)$ in panel c of Fig. 5.

Finally, let us examine the third experiment of the 10,000 randomly simulated examples with a large condition number. The experiment setting has been explained in Sect. 4.1 and will not be repeated here. As in the first two experiments, we compute all the performance indices from these 10,000 examples and show them in Table 5. The estimated probability density functions of $(\log \kappa_N - \log \kappa_B)$, $(\log \kappa_L - \log \kappa_B)$ and $(\log \kappa_N - \log \kappa_L)$ for this experiment are plotted in Fig. 6. Although the numbers of integer unknowns of the problems are also uniformly (but independently) distributed over 21 and 50, as in the second experiment, the condition numbers have been significantly increased by a maximum factor of (4×10^4). In this experiment, both the parallel Cholesky-based reduction method and the LLL algorithm have performed extremely well, which can also be immediately observed from the performance statistics in Table 5 and panels a and b of Fig. 6. The maximum improving factors are all in the order of magnitude six to seven for both methods. It is rather safe to say that the ill-conditioned nature of these problems has been successfully removed. After a careful examination of the remaining small fraction of examples with a worsened condition number, we found that all these examples are with a dimension higher than 40 and with a condition number of $\mathbf{W}_f$ in the order of 10^4. This phenomenon, however, is well consistent with what we have observed in the second experiment. It is interesting to see from column

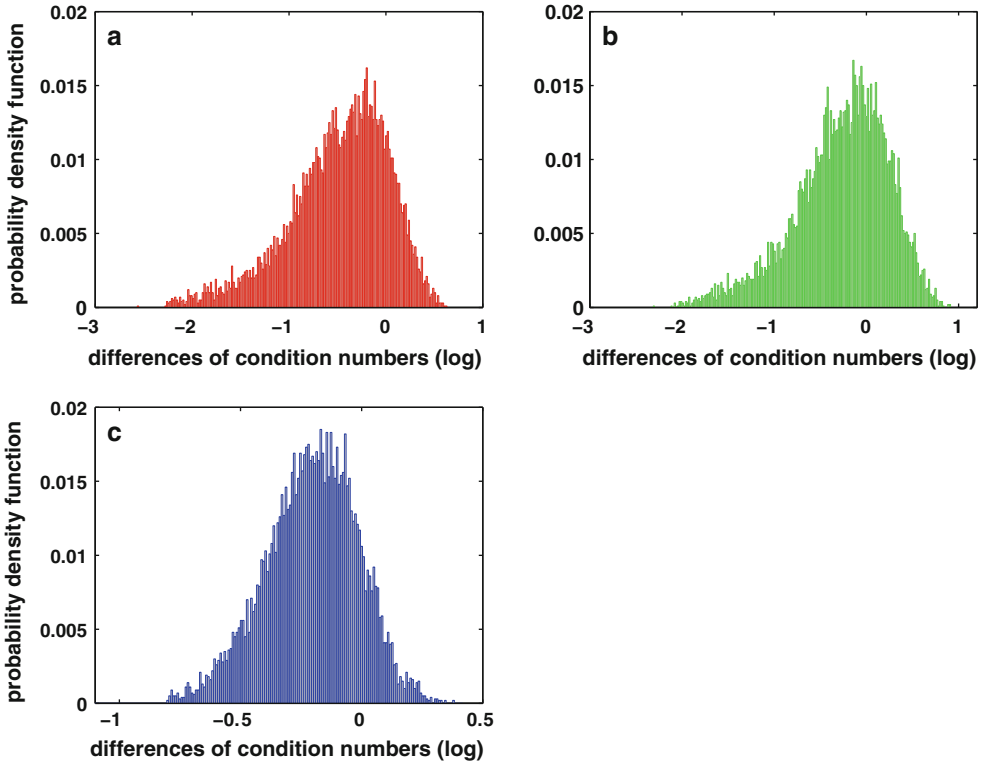

Fig. 5 The probability density functions of the differences of condition numbers for the second experiment, with a relative large dimension of $\mathbf{z}$ between 21 and 50. panel **a**—$(\log \kappa_N - \log \kappa_B)$ to compare the parallel Cholesky-based reduction with the original problems, panel **b**—$(\log \kappa_L - \log \kappa_B)$ to compare the LLL reduction with the original problems, and panel **c**—$(\log \kappa_N - \log \kappa_L)$ to compare the Cholesky-based reduction with the LLL algorithm

Table 5 Performance statistics of the parallel Cholesky-based reduction method and the LLL reduction algorithm with the third set of 10,000 random simulated examples

Methods	Chol_$\mathbf{W}_f$	LLL_$\mathbf{W}_f$	Chol_LLL
BExamples	99.99	98.77	99.55
WExamples	0.01	1.23	0.45
MeanImprove	−3.062	−2.264	−0.832
MaxImprove	−6.981	−6.405	−2.623
MeanWorsen	0.108	0.245	0.088
MaxWorsen	0.108	0.875	0.329

The condition numbers of these problems are between 10^4 and (1.5×10^9). All the statistical indices and notations shown in this table are the same as in Table 3

Chol_LLL of Table 5 that the parallel Cholesky-based reduction method performs overwhelmingly better than the LLL algorithm for almost all the examples in this experiment (also compare panel c of Fig. 6). This may indicate that the former is much more suitable for solving the ill-conditioned nature of problems than the LLL algorithm.

Before closing the analysis and comparison of the simulated results, we would like to briefly discuss the effect of a different maximum number of iterations on the performance of the parallel Cholesky-based reduction algorithm. More specifically, we set a new maximum number of iterations to m for the parallel reduction algorithm and repeat the second and third experiments. The first experiment is not repeated, since all of its 10,000 examples terminate naturally in less than m iterations, as has been seen in panel a of Fig. 2. To start the performance comparison with the different maximum numbers of iterations m and $3m$, let us denote the 10,000 condition numbers with the maximum number of iterations m by $\boldsymbol{\kappa}_N(m)$. After comparing $\boldsymbol{\kappa}_N(m)$ with $\boldsymbol{\kappa}_N$ of (24c) in the second experiment, we found that $\boldsymbol{\kappa}_N(m)$ and $\boldsymbol{\kappa}_N$ are essentially the same, except for only three examples. The differences of condition numbers of these three examples are also very small. The maximum difference is −0.101 in logarithm. As for the third experiment, we plot the cdf of $\{\log \kappa_N - \log \kappa_N(m)\}$ in Fig. 7. We may conclude from Fig. 7 that: (i) the results from using two different numbers of iterations for termination are not significantly different. In this experiment, 56.3% of the examples have exactly the same results. Nearly 90% of the 10,000 examples produce the condition numbers of κ_N and $\kappa_N(m)$ within a range of difference by 50%; and (ii) the increase of iterations tends to further improve the condition numbers $\kappa_N(m)$. The improvement seems to be not significant, because the probability of

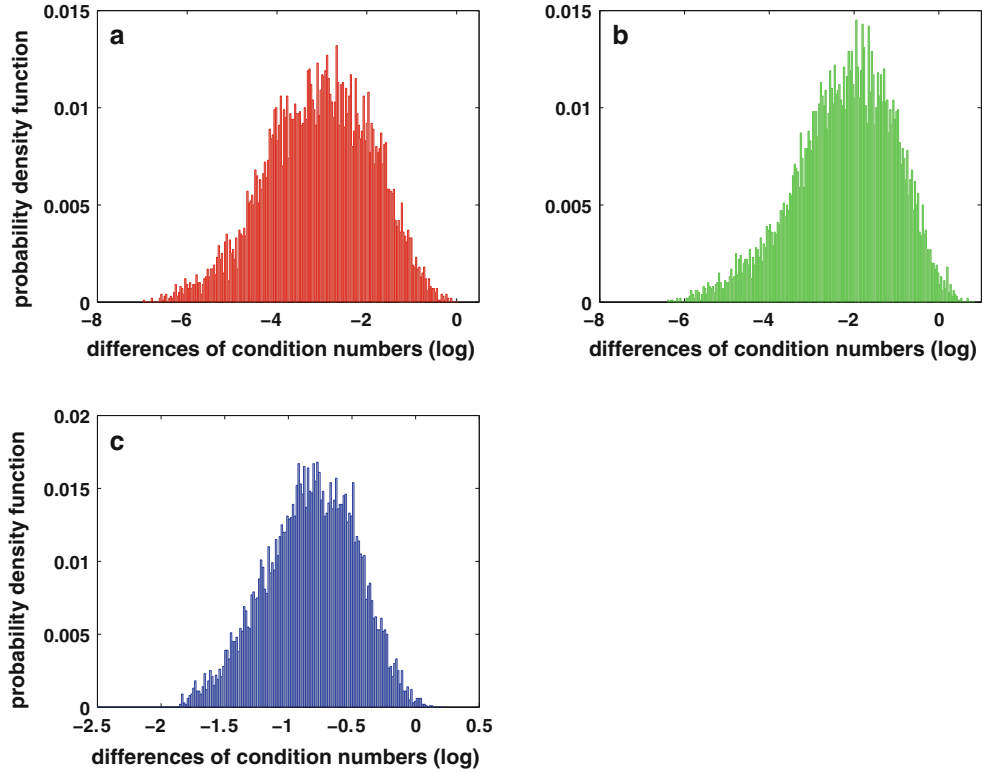

Fig. 6 The probability density functions of the differences of condition numbers for the third experiment with the 10,000 ill-conditioned problems. panel **a**—$(\log \kappa_N - \log \kappa_B)$ to compare the parallel Cholesky-based reduction with the original problems, panel **b**—$(\log \kappa_L - \log \kappa_B)$ to compare the LLL reduction with the original problems, and panel **c**—$(\log \kappa_N - \log \kappa_L)$ to compare the Cholesky-based reduction with the LLL algorithm

improvement is only 0.076 for $\kappa_N \leq 0.5\kappa_N(m)$. However, the increase of iterations does not always result in a smaller condition number, as clearly shown by the upper-right part of the cdf curve in Fig. 7.

5 Conclusion

The integer linear model (1) is highly interdisciplinary (see, e.g. Agrell et al. 2002; Artés et al. 2003; Banihashemi and Khandani 1998; Brunetti and Daurat 2003; Grafarend 2000; Regev 2009; Joux and Stern 1998; Gardner et al. 1999; Teunissen 1993; Xu et al. 1995; Xu 2006). As a statistical model, (1) is fundamentally different from a standard (real-valued) linear model well documented in the statistical literature in that the unknown parameters in (1) are integers but not real-valued. Actually (1) has received almost no attention in statistics. As a pure mathematical and/or engineering problem, mathematicians and engineers have either focused on the geometry of numbers or numerically estimating the integers **z** from the noise-contaminated measurements **y**. The error probability and its lower bound of correctly estimating integers were given by Shannon (1959). Tighter upper and lower probabilistic bounds were recently derived by Xu

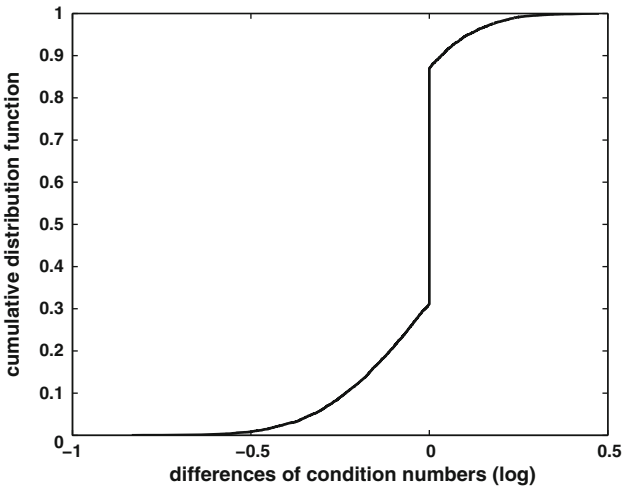

Fig. 7 The cumulative distribution function of the differences of condition numbers $\{\log \kappa_N - \log \kappa_N(m)\}$ from the 10,000 random examples of the third experiment. The condition numbers κ_N and $\kappa_N(m)$ are computed using the parallel Cholesky-based reduction with the maximum numbers of iterations set to two different values, namely, $3m$ and m, respectively

(2006) on the basis of best fitting of the Voronoi cell. However, statistical inference such as integer hypothesis testing and quality control on the integer linear model (1) and/or

the mixed integer linear model (2) has not yet been paid due attention (see, e.g. Xu 2006). When the least squares and/or ML principles are applied to (1), estimating the integers $\mathbf{z}$ from $\mathbf{y}$ is equivalent to numerically solving the weighted closest point or ILS problem (3). In this aspect, reduction of lattice vectors and positive definite quadratic forms has been widely used to aid finding the suboptimal or optimal integer solution of $\mathbf{z}$ in the integer linear model (1).

We have proposed a parallel reduction method of positive definite quadratic forms for solving the ILS problem (3), which consists of two basic components: Cholesky decomposition and reduction of the $\mathbf{L}$ matrix. Actually, the parallel Cholesky-based reduction method implements two versions of decomposition: (i) by simply arranging the diagonal elements of $\mathbf{W}_f$ in ascending order; and (ii) by perturbing the sorting strategy (i) with the ascending ordering of the diagonal elements of $\mathbf{D}$ in the beginning of reduction. Reduction with either of the sorting strategies runs independently in parallel. Our reduction method directly works on the positive definite matrix in association with (3).

We have shown that the parallel Cholesky-based reduction method satisfies part of the inequalities required by Minkowski's reduction of quadratic forms. It is of fixed complexity of $O(m^4)$ arithmetic operations by limiting the maximum number of iterations up to $3m$. The simulations of 30,000 randomly generated examples, with varying dimensions and varying condition numbers up to (1.5×10^9), have clearly shown that: (i) the parallel reduction method and the LLL algorithm can work very well practically to reduce the condition number of $\mathbf{W}_f$; and (ii) the parallel Cholesky-based reduction performs significantly better than the LLL algorithm in terms of producing a smaller condition number. Our reduction method has also been shown to be extremely powerful in removing the ill-conditioned nature of problems. Finally, we would like to note that although the sorted QR ordering is very powerful in constructing suboptimal solutions to (3), it is least effective for reduction when compared with the other two sorting strategies, as clearly shown by the simulations. A suboptimal solution is sensitive to the sequential ordering of integer parameters $\mathbf{z}$ according to the conditional variances of the real-valued solution $\mathbf{z}_f$, while reduction is more sensitive to simultaneously reducing both the diagonal and off-diagonal elements of $\mathbf{W}_f$.

Acknowledgments The author thanks three reviewers and the editors, Prof. J. Kusche and Prof. R. Klees, for their constructive comments which help improve the paper substantially.

References

Afflerbach L, Grothe H (1985) Calculation of Minkowski-reduced lattice bases. Computing 35:269–276

Agrell E, Eriksson T, Vardy A, Zeger K (2002) Closest point search in lattices. IEEE Trans Inf Theory 48:2201–2214

Akhavi A (2002) Random lattices, threshold phenomena and efficient reduction algorithms. Theor Comput Sci 287:359–385

Artés H, Seethaler D, Hlawatsch F (2003) Efficient detection algorithms for MIMO channels: a geometrical approach to approximate ML detection. IEEE Trans Signal Proc 51(11):2808–2820

Babai L (1986) On Lovász' lattice reduction and the nearest lattice point problem. Combinatorica 6:1–13

Banihashemi AH, Khandani AK (1998) On the complexity of decoding lattices using the KorkinZolotarev reduced basis. IEEE Trans Inf Theory 44:162–171

Brunetti S, Daurat A (2003) An algorithm reconstructing convex lattice sets. Theor Comput Sci 304:35–57

Conway JH, Sloane NJA (1999) Sphere packings, lattices and groups, 3rd edn. Springer, Berlin

Damen MO, Gamal HE, Caire G (2003) On maximum-likelihood detection and the search for the closest lattice point. IEEE Trans Inf Theory 49:2389–2402

Daudé H, Vallée B (1994) An upper bound on the average number of iterations of the LLL algorithm. Theor Comput Sci 123:95–115

Dickson TJ (1968) A sufficient condition for an extreme covering of n-space by spheres. J Aust Math Soc 8:56–62

Dickson TJ (1972) On Voronoi reduction of positive definite quadratic forms. J Numer Theory 4:330–341

Fincke U, Pohst M (1985) Improved methods for calculating vectors of short length in a lattice, including a complexity analysis. Math Comput 44:463–471

Gardner RJ, Gritzmann P, Prangenberg D (1999) On the computational complexity of reconstructing lattice sets from their X-rays. Discret Math 202:45–71

Grafarend EW (2000) Mixed integer-real valued adjustment (IRA) problems: GPS initial cycle ambiguity resolution by means of the LLL algorithm. GPS Solut 4:31–44

Grötschel M, Lovász L, Schrijver A (1988) Geometric algorithms and combinatorial optimization. Springer, Berlin

Gruber PM, Lekkerkerker CG (1987) Geometry of numbers. North-Holland, Amsterdam

Helfrich B (1985) Algorithms to construct Minkowski reduced and Hermite reduced lattice bases. Theor Comput Sci 41:125–139

Hofmann-Wellenhof B, Lichtenegger H, Collins J (1992) GPS—theory and practice. Springer, Berlin

Jaldén J, Seethaler D, Matz G (2008) Worst- and average-case complexity of LLL lattice reduction in MIMO wireless systems. In: Proceedings of IEEE international conference on acoustics, speech, and signal processing (ICASSP), vol 4, pp 2685–2688. Las Vegas, March 30–April 4

Joux A, Stern J (1998) Lattice reduction: a toolbox for the cryptanalyst. J Cryptol 11:161–185

Kannan R (1987) Algorithmic geometry of numbers. Ann Rev Comput Sci 2:231–267

Koch KR (1999) Parameter estimation and hypothesis testing in linear models, 2nd edn. Springer, Berlin

LaMacchia BA (1991) Basis reduction algorithms and subset sum problems. Master thesis, MIT

Lenstra AK, Lenstra HW, Lovász L (1982) Factoring polynomials with rational coefficients. Math Ann 261:515–534

Li D, Sun X (2006) Nonlinear integer programming. Springer, New York

Ling C, Mow WH (2009) A unified view of sorting in lattice reduction: from V-BLAST to LLL and beyond. In: Proceedings of 2009 IEEE information theory workshop, pp 529–533

Luk FT, Tracy DM (2008) An improved LLL algorithm. Linear Algebra Appl 428:441–452

Mahler K (1938) On Minkowski's theory of reduction of positive definite quadratic forms. Q J Math 9:259–262

Nemhauser G, Wolsey L (1988) Integer and combinatorial optimization. Wiley, New York

Nguyen PQ, Stehle D (2004) Low-dimensional lattice basis reduction revisited. In: ANTS 2004, LNCS, vol 3076. Springer, Berlin, pp 338–357

Nguyen PQ, Stehlé D (2009) An LLL algorithm with quadratic complexity. SIAM J Comput 39:874–903

Pohst M (1981) On the computation of lattice vectors of minimal length, successive minima and reduced bases with applications. ACM SIGGAM Bull 15:37–44

Regev O (2009) On Lattices, learning with errors, random linear codes, and cryptography. J ACM 56:34:1–34:40

Ryshkov SS (1976) The theory of Hermite–Minkowski reduction of positive definite quadratic forms. J Math Sci 6:651–671

Schnorr CP (1987) A hierarchy of polynomial time lattice basis reduction algorithm. Theor Comput Sci 53:201–224

Schnorr CP (2006) Fast LLL-type lattice reduction. Inf Comput 204: 1–25

Schnorr CP, Euchner M (1994) Lattice basis reduction: improved practical algorithms and solving subset sum problems. Math Prog 66:181–199

Seysen M (1993) Simultaneous reduction of a lattice basis and its reciprocal basis. Combinatorica 13:363–376

Shannon CE (1959) Probability of error for optimal codes in a Gaussian channel. Bell Syst Tech J 38:611–656

Taha H (1975) Integer programming—theory, applications, and computations. Academic Press, New York

Tammela PP (1976) The Hermite–Minkowski domain of reduction of positive definite quadratic forms in six variables. J Math Sci 6:677–688

Tammela PP (1979) Reduction theory of positive quadratic forms. J Math Sci 11:197–277

Tammela PP (1985) Venkov's reduction theory of positive-quadratic forms. J Math Sci 29:1306–1312

Teunissen PJG (1993) Least-squares estimation of the integer GPS ambiguities. In: LGR-Series No. 6, Delft Geodetic Computing Centre. Delft University of Technology, pp 59–74

Vallée B (1991) Gauss' algorithm revisited. J Algorithm 12:556–572

Vetter H, Ponnampalam V, Sandell M, Hoeher PA (2009) Fixed complexity LLL algorithm. IEEE Trans Signal Proc 57:1634–1637

Waters DW, Barry JR (2005a) A reduced-complexity lattice-aided decision-feedback detector. In: Proceedings of 2005 international conference wireless networks, communications and mobile computing, pp 845–850

Waters DW, Barry JR (2005b) Noise-predictive decision-feedback detection for multiple-input multiple-output channels. IEEE Trans Signal Proc 53:1852–1859

Wübben D, Böhnke R, Rinas J, Kühn V, Kammeyer KD (2001) Efficient algorithm for decoding layered space-time codes. Electron Lett 37:1348–1350

Wübben D, Böhnke R, Kühn V, Kammeyer KD (2003) MMSE extension of V-BLAST based on sorted QR decomposition. In: Proceedings of IEEE 58th vehicular technology conference, VTC 2003-Fall, vol 5, pp 508–512

Wübben D, Böhnke R, Kühn V, Kammeyer KD (2004) Near-maximum-likelihood detection of MIMO systems using MMSE-based lattice-reduction. In: Proceedings of IEEE International Conference Communications, Paris, vol 2, pp 798–802

Xu PL (1998) Mixed integer geodetic observation models and integer programming with applications to GPS ambiguity resolution. J Geod Soc Japan 44:169–187

Xu PL (2001) Random simulation and GPS decorrelation. J Geod 75:408–423

Xu PL (2006) Voronoi cells, probabilistic bounds and hypothesis testing in mixed integer linear models. IEEE Trans Inf Theory 52:3122–3138

Xu PL (2010) Mixed integer linear models, Chapter 38. In: Freeden W, Nashed Z, Sonar T (eds) Handbook of geomathematics. Springer, Berlin, pp 1129–1157

Xu PL, Cannon E, Lachapelle G (1995) Mixed integer programming for the resolution of GPS carrier phase ambiguities, presented at IUGG95 Assembly, Boulder, July 2–14. arXiv:1010.1052v1[cs.IT]

Xu PL, Cannon E, Lachapelle G (1999) Stabilizing ill-conditioned linear complementarity problems. J Geod 73:204–213