Available online at www.sciencedirect.com
ScienceDirect
Linear Algebra and its Applications xxx (2007)xxx–xxx
LINEAR ALGEBRA AND ITS APPLICATIONS
www.elsevier.com/locate/laa

An improved LLL algorithm

Franklin T. Luk*, Daniel M. Tracy [1]
Department of Computer Science, Rensselaer Polytechnic Institute, Troy, NY 12180, USA

Received 18 January 2007; accepted 27 February 2007
Submitted by M.K. Ng

Abstract

The LLL algorithm has received a lot of attention as an effective numerical tool for preconditioning an integer least squares problem. However, the workings of the algorithm are not well understood. In this paper, we present a new way to look at the LLL reduction, which leads to a new implementation method that performs better than the original LLL scheme.

Keywords: LLL algorithm; Integer least squares; Unimodular transformation; Reduced basis; Gauss transformation; QR decomposition; Plane reflection; Condition number

1. Introduction

The famous algorithm due to Lenstra, Lenstra and Lovasz (LLL [4]) has many important applications; for example, wireless communication, cryptography, and GPS (see [3] and references therein). In some of these applications, researchers use the LLL algorithm as a preconditioner in the solution of an integer least squares problem. Although the LLL algorithm is often referred to as an integer Gram–Schmidt procedure, no one has fully analyzed its numerical behavior. In this paper, we present a new way to examine the LLL reduction. Our idea leads to a new, generalized LLL technique that uses orthogonal instead of Gauss transformations in the reduction process.

Our paper is organized as follows. In Section 2, we describe the problem of integer least squares. We present the idea of a reduced basis and the LLL algorithm in Sections 3 and 4, respectively.

* Corresponding author.
E-mail address: luk@cs.rpi.edu (F.T. Luk).
[1] This work was supported in part by DARPA/DSO under the Geo* program.

0024-3795/$ - see front matter © 2007 Published by Elsevier Inc.
doi:10.1016/j.laa.2007.02.029

In Section 5, we extend the idea of a reduced basis to that of a reduced triangular matrix, and in Section 6, we present our new algorithm based on this extension. In Section 7, we show that the two methods will give the same results in exact arithmetic. We conclude the paper by presenting examples in Section 8 to illustrate how the numerical results produced by our new method can be significantly better than those produced by the original LLL method.

2. Integer least squares

Consider a linear least squares problem:

$$\min_{\mathbf{s}} \|B\mathbf{s} - \mathbf{y}\|_2, \tag{1}$$

where $B \in \mathbf{R}^{n\times n}$, $\mathbf{y} \in \mathbf{R}^n$, $\mathbf{s} \in \mathbf{Z}^n$, and B is nonsingular. One important application is wireless communication: in a multiple-input-multiple-output (MIMO) model with a finite impulse response (FIR), the matrix B could possess a block Toeplitz form. The problem (1) is NP hard; that is, the known solution algorithms all have exponential complexity. Indeed, most procedures are based on the Sphere Decoding Algorithm (SDA) of Pohst [5], which examines lattice points that lie inside a hypersphere. A two-step algorithm is given in Hassibi and Vikalo [3]:

1. Find the exact solution $B^{-1}\mathbf{y}$ and round each element of the vector to the closest integer:
$$\tilde{\mathbf{s}} = \lceil B^{-1}\mathbf{y} \rfloor.$$
The estimate $\tilde{\mathbf{s}}$ is called a Babai point.
2. Use $\tilde{\mathbf{s}}$ to determine the radius α of a sphere, and apply the SDA [5] to search over all points inside the sphere.

Note that the first step requires $O(n^3)$ flops and the second $O(\alpha^n)$ flops. If the matrix B has a special structure, such as a Toeplitz form, we could apply a fast QR decomposition technique and reduce the cost of step 1 to $O(n^2)$ flops. However, in light of the exponential cost in step 2, the saving is likely to be insignificant. To accelerate the convergence of SDA, Fincke and Pohst [1] suggested the use of the LLL algorithm.

To aid in the solution of (1), we use integer unimodular transformations:

Definition 1. A nonsingular matrix M is unimodular if $\det(M) = \pm 1$.

Lemma 1. *A nonsingular integer matrix M is unimodular if and only if M^{-1} is an integer matrix.*

Given B, the idea in Lenstra et al. [4] is to construct a unimodular matrix $M \in \mathbf{Z}^{n\times n}$ so that the columns of BM are almost orthogonal. The total work is $O(n^4)$. The integer least squares problem (1) then becomes

$$\min_{\mathbf{s}} \|(BM)(M^{-1}\mathbf{s}) - \mathbf{y}\|_2. \tag{2}$$

Using LLL as a preconditioner to reduce the condition number of BM in (2) is quite common in many applications; see, e.g., [3]. The preconditioning step is followed by a QR decomposition of BM to solve the least squares problem. In Section 5, we will present a simpler and better approach that combines the two sequential steps into one single step by computing both M and the QR decomposition of BM at the same time.

Please cite this article in press as: F.T. Luk, D.M. Tracy, An improved LLL algorithm, Linear Algebra Appl. (2007), doi:10.1016/j.laa.2007.02.029

3. Reduced basis

Let $B \in \mathbf{R}^{n\times n}$ be nonsingular. Consider its QR decomposition:

$$Q^{\mathrm{T}}B = DU, \tag{3}$$

where $Q \in \mathbf{R}^{n\times n}$ is orthogonal, $D \equiv \mathrm{diag}(d_i) \in \mathbf{R}^{n\times n}$ is diagonal with

$$d_i > 0 \quad \text{for } i = 1, 2, \ldots, n$$

and $U \equiv (u_{ij}) \in \mathbf{R}^{n\times n}$ is upper triangular with ones on its diagonal:

$$u_{ii} = 1 \quad \text{for } i = 1, 2, \ldots, n.$$

Note that instead of u_{ij}, the parameter μ_{ij} is used in [4]. Fortunately, the two parameters are related via

$$u_{ij} = \mu_{ji} \quad \text{forall } i \text{ and } j.$$

A key concept in the LLL algorithm is that of a reduced basis.

Definition 2 [4]. The columns of B form a reduced basis if

$$|u_{ij}| \leqslant 0.5 \quad \text{for } 1 \leqslant i < j \leqslant n \tag{4}$$

and

$$d_i^2 \geqslant (\omega - u_{i-1,i}^2)d_{i-1}^2 \quad \text{for } 2 \leqslant i \leqslant n, \tag{5}$$

where $0.25 < \omega < 1$ is a parameter that controls the rate of convergence.

Condition (4) states that the absolute value of any strictly upper triangular element of U is at most 0.5. Condition (5) states that the diagonal elements of D must be *ordered* in a certain manner. Let $\omega = 0.75$, a usual choice in [4]. Then (5) can be rewritten as

$$d_i^2 \geqslant (0.75 - u_{i-1,i}^2)d_{i-1}^2 \geqslant (0.75 - 0.5^2)d_{i-1}^2 = 0.5d_{i-1}^2. \tag{6}$$

Eq. (6) says that d_i^2 must be at least half as large as d_{i-1}^2.

Lemma 2. *Since the value of the quantity inside the parentheses in* (5) *is always less than one, an upper triangular matrix* $B \in \mathbf{R}^{n\times n}$ *with a constant diagonal satisfies condition* (5).

Example 1. For $1 \leqslant i < j \leqslant n$, let u_{ij} denote any number so that $|u_{ij}| \leqslant 0.5$. The columns of this triangular matrix $B_u \in \mathbf{R}^{n\times n}$ form a reduced basis:

$$B_u \equiv \begin{bmatrix} 1 & u_{12} & u_{13} & \cdots & \cdots & u_{1n} \\ & 1 & u_{23} & \cdots & \cdots & u_{2n} \\ & & 1 & \ddots & \cdots & u_{3n} \\ & & & \ddots & \ddots & \vdots \\ & & & & 1 & u_{n-1,n} \\ & & & & & 1 \end{bmatrix}. \tag{7}$$

Please cite this article in press as: F.T. Luk, D.M. Tracy, An improved LLL algorithm, Linear Algebra Appl. (2007), doi:10.1016/j.laa.2007.02.029

4. LLL reduction algorithm

In this section, we describe the actions of the LLL algorithm by showing how conditions (4) and (5) are enforced.

Condition (4) is easy to impose on $U \equiv (u_{ij})$, an upper triangular matrix with a unit diagonal. We begin by defining an elementary unimodular transformation. Let $i < j$, and let $\mathbf{e}_i \in \mathbf{Z}^n$ and $\mathbf{e}_j \in \mathbf{Z}^n$ denote unit coordinate vectors in the ith and jth directions, respectively. Define $M_{ij} \in \mathbf{Z}^{n\times n}$ by

$$M_{ij} \equiv I - \gamma \mathbf{e}_i \mathbf{e}_j^{\mathrm{T}}, \tag{8}$$

where γ is an integer.

Lemma 3. *The matrix M_{ij} defined in (8) is an integer unimodular transformation.*

We use M_{ij} to ensure that the (i, j)th element of U is sufficiently small. Suppose that (4) is not satisfied for some i and j; that is

$$|u_{ij}| > 0.5.$$

Calculate γ as the integer closest to u_{ij}:

$$\gamma = \lceil u_{ij} \rfloor. \tag{9}$$

Construct the unimodular matrix M_{ij} with its (i, j)th element equal to $-\gamma$. Apply M_{ij} to B and to U:

$$B \leftarrow BM_{ij} \quad \text{and} \quad U \leftarrow UM_{ij}. \tag{10}$$

It is straightforward to check that the (i, j)th element of the new U satisfies (4).

PROCEDURE DECREASE(i, j) *Given B and U, calculate M_{ij} and γ using (8) and (9), respectively. Apply M_{ij} to B and to U:*

$$B \leftarrow BM_{ij} \quad \textit{and} \quad U \leftarrow UM_{ij}.$$

For condition (5), we need to define two numerical transformations.

Notation 1. The matrix $\Pi_i \in \mathbf{Z}^{n\times n}$ denotes a permutation in the $(i-1, i)$ plane, where $2 \leqslant i \leqslant n$.

Notation 2. The matrix $X_i \in \mathbf{R}^{n\times n}$ denotes a transformation in the $(i-1, i)$ plane, where $2 \leqslant i \leqslant n$. It has the form:

$$X_i \equiv \begin{bmatrix} I_{i-2} & & & \\ & \mu & 1-\xi\mu & \\ & 1 & -\xi & \\ & & & I_{n-i} \end{bmatrix}. \tag{11}$$

Note that

$$\det(X_i) = -1 \tag{12}$$

Please cite this article in press as: F.T. Luk, D.M. Tracy, An improved LLL algorithm, Linear Algebra Appl. (2007), doi:10.1016/j.laa.2007.02.029

and that X_i^{-1} is given by

$$X_i^{-1} = \begin{bmatrix} I_{i-2} & & & \\ & \xi & 1-\xi\mu & \\ & 1 & -\mu & \\ & & & I_{n-i} \end{bmatrix}. \tag{13}$$

It is shown in [4] that the matrix X_i^{-1} is made up of a product of two Gauss transformations [2]. Indeed, here is a quick illustration:

$$\begin{bmatrix} \xi & 1-\xi\mu \\ 1 & -\mu \end{bmatrix} = \begin{bmatrix} 1 & \xi \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 0 & 1 \\ 1 & -\mu \end{bmatrix}.$$

This matrix X_i^{-1} is a workhorse in the LLL algorithm, and the following relation is key:

$$\begin{bmatrix} \xi & 1-\xi\mu \\ 1 & -\mu \end{bmatrix} \begin{bmatrix} 1 & \mu \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} = \begin{bmatrix} 1 & \xi \\ 0 & 1 \end{bmatrix}. \tag{14}$$

In words, Eq. (14) says that the matrix X_i^{-1} restore the triangularity of a permuted triangular matrix. Note that both triangular matrices in (14) have ones on their diagonals.

Suppose that (5) is not satisfied for some i:

$$d_i^2 < [\omega - u_{i-1,i}^2] d_{i-1}^2.$$

We interchange columns i and $i-1$ of B and of U:

$$B \leftarrow B\Pi_i \quad \text{and} \quad U \leftarrow U\Pi_i. \tag{15}$$

We then use the transformation X_i^{-1} of (13) to restore U to triangular form:

$$U \leftarrow X_i^{-1} U. \tag{16}$$

The LLL paper [4] gives the formulas on updating the squares of the diagonal elements d_{i-1} and d_i of D. We skip the details and summarize the transformation in D^2 by

$$D^2 \leftarrow D^2_{\text{new}}. \tag{17}$$

The paper [4] also gives the values of ξ and μ in (11). As is obvious from (14), μ is given by

$$\mu = u_{i-1,i}. \tag{18}$$

In addition, ξ is given by

$$\xi = \mu \cdot d_{i-1}^2 / (d_i^2 + \mu^2 d_{i-1}^2). \tag{19}$$

PROCEDURE SWAP(i) *Given D^2, B and U, update D^2, swap columns $i-1$ and i of B and of U, and use the transformation X_i^{-1} to transform the permuted U back to triangular form*:

$$D^2 \leftarrow D^2_{\text{new}}, \quad B \leftarrow B\Pi_i, \quad \text{and} \quad U \leftarrow X_i^{-1} U\, \Pi_i. \tag{20}$$

The matrix X_i^{-1} is computed using Eqs. (11), (18), *and* (19).

We now present the LLL algorithm. A proof of convergence is given in [4].

ALGORITHM LLL Given B, transform its columns so that they will form a reduced basis.
compute QR decomposition of B to get D^2 and U;
set $k \leftarrow 2$;

Please cite this article in press as: F.T. Luk, D.M. Tracy, An improved LLL algorithm, Linear Algebra Appl. (2007), doi:10.1016/j.laa.2007.02.029

```
while k ⩽ n
    if |u_{k−1,k}| > 0.5 then DECREASE (k − 1, k);
    if d_k^2 < [ω − u_{k−1,k}^2]d_{k−1}^2 then
      SWAP(k);
      k ← max(k − 1, 2);
    else
      for i = k − 2 down to 1
        if |u_{ik}| > 0.5 then DECREASE(i, k);
      k ← k + 1.
```

It is well known (see [3] and references therein) that the LLL algorithm is an effective tool in reducing the condition number of a given matrix. However, LLL sometimes fails to decrease the condition number of an ill-conditioned matrix. We present one such example here.

Example 2. The LLL algorithm does not modify the matrix B_u of (7) because its columns already form a reduced basis. Choose $u_{ij} = -0.5$ for all i and j, and we get

$$\widehat{B} \equiv \begin{bmatrix} 1 & -0.5 & -0.5 & \cdots & \cdots & -0.5 \\ & 1 & -0.5 & \cdots & \cdots & -0.5 \\ & & 1 & \ddots & \cdots & -0.5 \\ & & & \ddots & \ddots & \vdots \\ & & & & 1 & -0.5 \\ & & & & & 1 \end{bmatrix}. \tag{21}$$

The matrix $\widehat{B}$ is very ill-conditioned. Consider the matrix equation:

$$\widehat{B}\mathbf{x} = \mathbf{e}_n,$$

where $n \geqslant 2$. The first element of the solution vector $\mathbf{x}$ equals $(1.5)^{n-2}/2$. Thus, the smallest singular value of $\widehat{B}$ decreases like $2(1.5)^{-n+2}$ as n becomes large.

5. A new idea

We extend the idea of a reduced basis formed by the columns vectors to that of a *reduced* triangular matrix. Let $B \in \mathbf{R}^{n\times n}$ be nonsingular. Consider its QR decomposition:

$$Q^{\mathrm{T}}B = R, \tag{22}$$

where $Q \in \mathbf{R}^{n\times n}$ is orthogonal and $R \equiv (r_{ij}) \in \mathbf{R}^{n\times n}$ is upper triangular with a positive diagonal:

$$r_{ii} > 0 \quad \text{for } i = 1, 2, \ldots, p.$$

The concept in [4] can be rewritten as follows.

Definition 3. The columns of B form a reduced basis if

$$r_{ii} \geqslant 2|r_{ij}| \quad \text{for } 1 \leqslant i < j \leqslant n \tag{23}$$

and

$$r_{ii}^2 \geqslant [\omega - (r_{i-1,i}/r_{i,i})^2]r_{i-1,i-1}^2 \quad \text{for } 2 \leqslant i \leqslant n, \tag{24}$$

where $0.25 < \omega < 1$ is a parameter that controls the rate of convergence.

Please cite this article in press as: F.T. Luk, D.M. Tracy, An improved LLL algorithm, Linear Algebra Appl. (2007), doi:10.1016/j.laa.2007.02.029

Definition 4. A triangular matrix R is reduced if its elements satisfy conditions (23) and (24).

Our extension will lead to a new algorithm to transform a given matrix B to a reduced triangular matrix R.

Proposition 1. *Given $B \in \mathbf{R}^{n\times n}$, our new algorithm generates an orthogonal matrix $Q \in \mathbf{R}^{n\times n}$ and a unimodular matrix $M \in \mathbf{Z}^{n\times n}$ to transform B into a triangular matrix R :*

$$Q^{\mathrm{T}}BM = R, \tag{25}$$

so that R is reduced. The columns of BM form a reduced basis as defined in [4].

Our proposition will be proved by construction in the next section. We note that our new decomposition (25) is ideal for solving the integer least squares problem (2).

6. A new algorithm

In this section, we present our new algorithm and show how it enforces conditions (23) and (24). While condition (23) states that any diagonal element of R is at least twice as large as any other element of R along the same row, condition (24) states that the diagonal elements of R must be *ordered* in a certain way. Let $\omega = 0.75$, a usual choice in [4]. Then (24) can be rewritten as

$$r_{ii}^2 \geqslant [0.75 - (r_{i-1,i}/r_{i,i})^2]r_{i-1,i-1}^2 \geqslant [0.75 - 0.5^2]r_{i-1,i-1}^2 = 0.5r_{i-1,i-1}^2. \tag{26}$$

Eq. (26) says that r_{ii}^2 must be at least half as large as $r_{i-1,i-1}^2$.
We use M_{ij} of (8) to ensure that the (i, j)th element of R is sufficiently small. Suppose that (23) is not satisfied for some i and j; that is

$$r_{ii} < 2|r_{ij}|.$$

Calculate γ as the integer closest to r_{ij}/r_{ii}:

$$\gamma = \lceil r_{ij}/r_{ii} \rfloor. \tag{27}$$

Construct the unimodular matrix M_{ij} with its (i, j)th element equal to $-\gamma$. Apply M_{ij} to R:

$$R \leftarrow RM_{ij} \tag{28}$$

and accumulate the transformations in M:

$$M \leftarrow MM_{ij}.$$

It is easy to check that the (i, j)th element of the new R in (28) satisfies (23).

Procedure NewDecrease(i, j) *Given R and M, calculate M_{ij} and γ using* (8) *and* (27), *respectively, and apply M_{ij} to both R and M*:

$$R \leftarrow RM_{ij} \quad \textit{and} \quad M \leftarrow MM_{ij}.$$

For condition (24) we need a basic numerical transformation [2].

Notation 3. The symmetric matrix $J_i \in \mathbf{R}^{n\times n}$ denotes a plane reflection in the $(i-1, i)$ plane, where $2 \leqslant i \leqslant n$. It has the form:

$$J_i \equiv \begin{bmatrix} I_{i-2} & & & \\ & c & s & \\ & s & -c & \\ & & & I_{n-i} \end{bmatrix}, \tag{29}$$

Please cite this article in press as: F.T. Luk, D.M. Tracy, An improved LLL algorithm, Linear Algebra Appl. (2007), doi:10.1016/j.laa.2007.02.029

where $c^2 + s^2 = 1$.

Note that

$$\det(J_i) = -1, \tag{30}$$

just like $\det(X_i) = -1$ in (12). In this paper, we use plane reflections instead of plane rotations because the X_i's are closely related to plane reflections, as we will show in the next section. Suppose that (24) is not satisfied for some i:

$$r_{ii}^2 < [\omega - (r_{i-1,i}/r_{i,i})^2] r_{i-1,i-1}^2.$$

We interchange columns i and $i-1$ of R:

$$R \leftarrow R\,\Pi_i \tag{31}$$

and use a plane reflection J_i to restore R to triangular form:

$$R \leftarrow J_i R. \tag{32}$$

We accumulate the transformations in M and in Q:

$$M \leftarrow M\Pi_i \quad \text{and} \quad Q \leftarrow QJ_i.$$

PROCEDURE NEWSWAP(i) *Given R, M, and Q, swap columns $i-1$ and i of R and M, use a plane reflection J_i to transform the permuted R back to triangular form, and update Q*:

$$R \leftarrow J_i R\,\Pi_i, \quad M \leftarrow M\Pi_i \quad \textit{and} \quad Q \leftarrow QJ_i. \tag{33}$$

Now, we have all the tools to present our new algorithm as a matrix decomposition technique.

ALGORITHM NEW Given B, compute M, Q, and R, so that $BM = QR$ and R is reduced.

```
compute B = QR;
set M ← I and k ← 2;
while k ⩽ n
    if r_{k-1,k-1} < 2|r_{k-1,k}| then NEWDECREASE(k − 1, k);
    if r_{kk}^2 < [ω − (r_{k-1,k}/r_{kk})^2] r_{k-1,k-1}^2 then
      NEWSWAP(k);
      k ← max(k − 1, 2);
    else
      for i = k − 2 down to 1
        if r_{ii} < 2|r_{ik}| then NEWDECREASE(i, k);
      k ← k + 1.
```

7. Comparing the two algorithms

There are many similarities between Algorithms LLL and New. Both algorithms aim to reduce the given matrix B to a triangular form, and the overall structures are identical. The only difference lies in the transformations used: Algorithm New applies plane reflections J_i of (29) directly to R, while Algorithm LLL applies special transformations X_i^{-1} of (13) to U and updates D^2 separately. In this section, we will derive two $n \times n$ diagonal matrices D_1 and D_2 such that

$$J_i = D_1 X_i^{-1} D_2. \tag{34}$$

Please cite this article in press as: F.T. Luk, D.M. Tracy, An improved LLL algorithm, Linear Algebra Appl. (2007), doi:10.1016/j.laa.2007.02.029

Thus, we may view X_i^{-1} as a scaled plane reflection. We will also show that in exact arithmetic, the two algorithms will produce identical numerical results.

Representing the effect of transformations (31) and (32) by

$$R_{\text{new}} = J_i R \Pi_i,$$

we write out the key 2×2 transformations as follows:

$$\begin{bmatrix} \hat{\alpha} & \hat{\gamma} \\ 0 & \hat{\beta} \end{bmatrix} = \begin{bmatrix} c & s \\ s & -c \end{bmatrix} \begin{bmatrix} \alpha & \gamma \\ 0 & \beta \end{bmatrix} \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \tag{35}$$

from which we get

$$\begin{bmatrix} \hat{\alpha} & \hat{\gamma} \\ 0 & \hat{\beta} \end{bmatrix} = \begin{bmatrix} c & s \\ s & -c \end{bmatrix} \begin{bmatrix} \gamma & \alpha \\ \beta & 0 \end{bmatrix} = \begin{bmatrix} c\gamma + s\beta & c\alpha \\ s\gamma - c\beta & s\alpha \end{bmatrix}. \tag{36}$$

Eq. (35) can be transformed into

$$\begin{bmatrix} 1 & \hat{\gamma}/\hat{\alpha} \\ 0 & 1 \end{bmatrix} = \begin{bmatrix} 1/\hat{\alpha} & 0 \\ 0 & 1/\hat{\beta} \end{bmatrix} \begin{bmatrix} c & s \\ s & -c \end{bmatrix} \begin{bmatrix} \alpha & 0 \\ 0 & \beta \end{bmatrix} \begin{bmatrix} 1 & \gamma/\alpha \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}.$$

Define a new transformation Y by

$$Y \equiv \begin{bmatrix} 1/\hat{\alpha} & 0 \\ 0 & 1/\hat{\beta} \end{bmatrix} \begin{bmatrix} c & s \\ s & -c \end{bmatrix} \begin{bmatrix} \alpha & 0 \\ 0 & \beta \end{bmatrix}. \tag{37}$$

Then

$$\begin{bmatrix} 1 & \hat{\gamma}/\hat{\alpha} \\ 0 & 1 \end{bmatrix} = Y \begin{bmatrix} 1 & \gamma/\alpha \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$$

and

$$Y = \begin{bmatrix} c\alpha/\hat{\alpha} & s\beta/\hat{\alpha} \\ s\alpha/\hat{\beta} & -c\beta/\hat{\beta} \end{bmatrix} = \begin{bmatrix} \hat{\gamma}/\hat{\alpha} & (\hat{\alpha} - c\gamma)/\hat{\alpha} \\ \hat{\beta}/\hat{\beta} & -s\gamma/(s\alpha) \end{bmatrix} = \begin{bmatrix} \hat{\gamma}/\hat{\alpha} & 1 - \hat{\gamma}\gamma/(\hat{\alpha}\alpha) \\ 1 & -\gamma/\alpha \end{bmatrix}$$

by using the equalities in (36). If we choose

$$\xi = \hat{\gamma}/\hat{\alpha} \quad \text{and} \quad \mu = \gamma/\alpha, \tag{38}$$

then we get

$$Y = \begin{bmatrix} \xi & 1 - \xi\mu \\ 1 & -\mu \end{bmatrix}$$

and

$$\begin{bmatrix} 1 & \xi \\ 0 & 1 \end{bmatrix} = Y \begin{bmatrix} 1 & \mu \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}. \tag{39}$$

Note that (39) is exactly Eq. (14) for the LLL method. Also, we can easily prove that the μ and ξ as defined in (38) have the same values as the μ and ξ as defined in (18) and (19). Thus, the transformation Y is exactly the 2×2 part of the workhorse X_i^{-1} of the LLL algorithm.

From (37), we get

$$\begin{bmatrix} c & s \\ s & -c \end{bmatrix} = \begin{bmatrix} \hat{\alpha} & 0 \\ 0 & \hat{\beta} \end{bmatrix} Y \begin{bmatrix} 1/\alpha & 0 \\ 0 & 1/\beta \end{bmatrix}.$$

Please cite this article in press as: F.T. Luk, D.M. Tracy, An improved LLL algorithm, Linear Algebra Appl. (2007), doi:10.1016/j.laa.2007.02.029

Let

$$D \equiv \begin{bmatrix} E_1 & & & \\ & \alpha & 0 & \\ & 0 & \beta & \\ & & & E_2 \end{bmatrix}, \tag{40}$$

where $E_1 \in \mathbf{R}^{(i-2)\times(i-2)}$ and $E_2 \in \mathbf{R}^{(n-i)\times(n-i)}$ are positive diagonal matrices. Define

$$D_1 \equiv \begin{bmatrix} E_1 & & & \\ & \hat{\alpha} & 0 & \\ & 0 & \hat{\beta} & \\ & & & E_2 \end{bmatrix} \quad \text{and} \quad D_2 \equiv \begin{bmatrix} E_1^{-1} & & & \\ & 1/\alpha & 0 & \\ & 0 & 1/\beta & \\ & & & E_2^{-1} \end{bmatrix}. \tag{41}$$

Then

$$J_i = D_1 X_i^{-1} D_2. \tag{42}$$

Consider

$$J_i R = D_1 X_i^{-1} D_2 R.$$

We see that D_2 reduces R to a unit-diagonal triangular matrix (namely U), and that D_1 gives the new diagonal of $D_2 R$ after the transformation by X_i^{-1}. Therefore, we conclude that Algorithms LLL and New produce the same numerical results in exact arithmetic. It also follows that the convergence result for Algorithm LLL in [4] is applicable to Algorithm New.

The LLL algorithm [4] is numerically efficient in that it avoids the computation of square roots, which is one reason why it updates D^2 instead of D. Thus, we may view the transformations in the LLL method as square-root-free plane reflections. The potential cost for this efficiency is a possible loss in numerical accuracy when the given matrix is ill-conditioned, as we shall show in the next section.

8. Numerical experiments

In this section, we present numerical examples to compare our new method against the original LLL algorithm. The initial matrix $B \in \mathbf{R}^{n\times n}$ is upper triangular with each nonzero element as a random number in $(-1, 1)$. We use the symbol κ to represent the condition number of a matrix. Thus,

$$\kappa(B) \equiv \mathrm{cond}(B).$$

For well-conditioned test matrices, the two different schemes produce essentially identical results. Hence we show mostly ill-conditioned examples in Table 1. However, to avoid matrices that are numerically singular, we place an upper limit on the condition number of B:

$$\kappa(B) \leqslant 10^{15};$$

that is, we would keep on generating test matrices until we get one matrix that has a sufficiently small condition number. To see which method is better, we compare the two resultant triangular matrices DU and R, and the condition numbers of the two resultant BM's. In exact arithmetic, DU should equal R. We therefore calculate the Frobenius norm of the difference:

$$\|DU - R\|_{\mathrm{F}},$$

and normalize the result by dividing by the quantity $\alpha(n)$, given by

$$\alpha(n) \equiv \sqrt{n(n+1)/2}.$$

Please cite this article in press as: F.T. Luk, D.M. Tracy, An improved LLL algorithm, Linear Algebra Appl. (2007), doi:10.1016/j.laa.2007.02.029

Table 1
LLL method against our new LLL method on ill-conditioned matrices

n	$\kappa(B)$	$d_{\max}$	$r_{\max}$	$\|DU - R\|_F/\alpha(n)$	$\kappa_{\mathrm{LLL}}(BM)$	$\kappa_{\mathrm{New}}(BM)$
5	3.74×10^{2}	0.41	0.41	1.30×10^{-15}	2.84×10^{0}	2.84×10^{0}
10	2.13×10^{3}	0.57	0.57	7.60×10^{-15}	4.58×10^{0}	4.58×10^{0}
15	3.95×10^{4}	0.61	0.61	9.69×10^{-14}	1.03×10^{1}	1.03×10^{1}
20	8.32×10^{4}	0.78	0.78	7.79×10^{-13}	9.29×10^{0}	9.29×10^{0}
25	2.24×10^{8}	0.78	0.78	3.43×10^{-11}	1.78×10^{2}	1.78×10^{2}
30	4.96×10^{6}	0.92	0.92	4.95×10^{-9}	2.19×10^{1}	2.19×10^{1}
35	3.96×10^{9}	0.93	0.93	7.85×10^{-4}	4.04×10^{1}	4.04×10^{1}
40	1.07×10^{9}	0.94	0.94	1.83×10^{-5}	3.57×10^{1}	3.57×10^{1}
45	2.94×10^{13}	1.96	0.99	1.26×10^{-1}	5.05×10^{2}	5.16×10^{1}
50	1.32×10^{14}	4.72	0.85	2.04×10^{-1}	4.17×10^{3}	6.81×10^{1}
55	4.67×10^{13}	1.39	0.93	1.06×10^{-1}	2.13×10^{3}	8.06×10^{1}
60	1.52×10^{13}	12.58	0.87	4.49×10^{-1}	6.73×10^{3}	9.54×10^{1}
65	2.80×10^{14}	6.03	0.74	2.34×10^{-1}	1.28×10^{5}	1.72×10^{2}
70	1.84×10^{14}	3.81	0.75	2.53×10^{-1}	3.14×10^{6}	2.06×10^{2}
75	3.75×10^{14}	11.44	0.90	4.60×10^{-1}	2.77×10^{6}	3.26×10^{2}
80	1.50×10^{14}	5.57	0.93	2.15×10^{-1}	4.65×10^{5}	3.31×10^{2}
85	9.45×10^{14}	8.60	0.75	2.83×10^{-1}	2.73×10^{7}	3.62×10^{2}
90	1.46×10^{13}	14.81	0.98	6.02×10^{-1}	1.16×10^{6}	3.43×10^{2}
95	1.61×10^{14}	13.18	0.79	4.54×10^{-1}	9.61×10^{5}	4.11×10^{2}
100	5.97×10^{14}	32.78	0.84	8.42×10^{-1}	1.53×10^{9}	6.47×10^{2}

Of course, even when $DU \neq R$, we cannot conclude which method is better. Thus, we also compare two quantities:

$$d_{\max} \equiv \max_{1 \leqslant i \leqslant n} d_i \quad \text{and} \quad r_{\max} \equiv \max_{1 \leqslant i \leqslant n} r_{ii}.$$

Since both triangular matrices U and R are reduced, we see that $d_{\max}$ and $r_{\max}$ represent the maximal elements of DU and R, respectively. As all elements of the initial matrix B are smaller than 1 in magnitude, we expect that a good numerical method should keep the maximal element of the resultant matrix small than 1. In Table 1, we see that

$$d_{\max} > 1 \quad \text{for } n \geqslant 45 \qquad \text{and} \qquad r_{\max} < 1 \quad \text{for } n \leqslant 100.$$

Our proposal that our method works better than the original LLL method is also supported by the values of $\kappa(BM)$. In Table 1, we observe that

$$\kappa_{\mathrm{LLL}}(BM) > 10^{3} \quad \text{for } n \geqslant 50 \qquad \text{and} \qquad \kappa_{\mathrm{New}}(BM) < 10^{3} \quad \text{for } n \leqslant 100.$$

Indeed, when $n = 100$, we get

$$d_{\max} = 32.78 \quad \text{and} \quad r_{\max} = 0.84$$

and

$$\kappa_{\mathrm{LLL}}(BM) = 1.53 \times 10^{9} \quad \text{and} \quad \kappa_{\mathrm{New}}(BM) = 6.47 \times 10^{2}.$$

Thus, our experimental results confirm our theory that our method should produce more accurate results than the LLL method because orthogonal transformations are more stable than Gauss transformations.

Please cite this article in press as: F.T. Luk, D.M. Tracy, An improved LLL algorithm, Linear Algebra Appl. (2007), doi:10.1016/j.laa.2007.02.029

Acknowledgements

The authors acknowledge helpful discussions with Professors Richard Brent and Joohwan Chun. They are grateful to Professor Brent for many valuable suggestions to improve the paper.

References

[1] U. Fincke, M. Pohst, Improved methods for calculating vectors of short length in a lattice, including a complexity analysis, Math. Comp. 44 (1985) 463–471.
[2] G.H. Golub, C.F. Van Loan, Matrix Computations, third ed., The Johns Hopkins University Press, Baltimore, MD, 1996.
[3] B. Hassibi, H. Vikalo, On the sphere-decoding algorithm I. Expected complexity, IEEE Trans. Signal Process. 53 (2005) 2806–2818.
[4] A.K. Lenstra, H.W. Lenstra, L. Lovasz, Factoring polynomials with rational coefficients, Mathematische Annalen 261 (1982) 515–534.
[5] M. Pohst, On the computation of lattice vectors of minimal length, successive minima and reduced bases with applications, ACM SIGSAM Bull. 15 (1981) 37–44.

Please cite this article in press as: F.T. Luk, D.M. Tracy, An improved LLL algorithm, Linear Algebra Appl. (2007), doi:10.1016/j.laa.2007.02.029