Numerical Properties of the LLL Algorithm

Franklin T. Luk[a] and Sanzheng Qiao[b]

[a]Department of Mathematics, Hong Kong Baptist University, Kowloon Tong, Hong Kong
[b]Dept. of Computing and Software, McMaster Univ., Hamilton, Ontario L8S 4L7, Canada

ABSTRACT

The LLL algorithm is widely used to solve the integer least squares problems that arise in many engineering applications. As most practitioners did not understand how the LLL algorithm works, they avoided the issue by referring to the method as an integer Gram Schmidt approach (without explaining what they mean by this term). Luk and Tracy[1] were first to describe the behavior of the LLL algorithm, and they presented a new numerical implementation that should be more robust than the original LLL scheme. In this paper, we compare the numerical properties of the two different LLL implementations.

Keywords: LLL algorithm, unimodular transformation, QR decomposition, reduced basis, Gauss transformation, plane reflection, numerical overflow and underflow.

1. INTRODUCTION

The famous algorithm due to Lenstra, Lenstra and Lovasz[2] has many important applications; for example, wireless communication, cryptography, and GPS (see Hassibi and Vikalo[3] and references therein). In some of these applications, researchers use the LLL algorithm as a preconditioner in solving an integer least squares problem. Although the LLL algorithm is often referred to as an integer Gram-Schmidt procedure, no one has explained the workings of such a process. Luk and Tracy[1] achieved a breakthrough by showing how an LLL reduction can be implemented using orthogonal instead of Gauss transformations. The purpose of this paper is to compare the two different numerical implementations of the LLL method.

This paper is organized as follows. In Sections 2 and 3, we describe the original[2] and new[1] implementations of the LLL algorithm. In Section 4, we present the result[1] that the two different implementations give the same answers in exact arithmetic. Lastly, in Section 5, we conclude the paper by presenting examples to compare the numerical properties of the two implementations.

2. LLL ALGORITHM

Given a nonsingular matrix $B \in \mathbf{R}^{n\times n}$, an idea in Lenstra et al.[2] is to construct a unimodular matrix $M \in \mathbf{Z}^{n\times n}$ so that the columns of BM become almost orthogonal; a usual consequence is that the condition number of BM will become much smaller than that of B.

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.*

A key concept in the LLL paper[2] is that of a reduced basis. Consider the QR decomposition of B:

$$Q^{\mathrm{T}}B = DU, \tag{1}$$

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

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

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

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

Send correspondence to S. Qiao: qiao@mcmaster.ca

DEFINITION 2. *The columns of B form a reduced basis if*

$$|u_{i,j}| \le 0.5, \quad \textit{for } 1 \le i < j \le n, \tag{2}$$

and

$$d_i^2 \ge (\omega - u_{i-1,j}^2) d_{i-1}^2, \quad \textit{for } 2 \le i \le n, \tag{3}$$

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

Condition (2) states that the absolute value of any strictly upper triangular element of U is at most 0.5. Condition (3) states that the diagonal elements of U must be *ordered* in a certain manner.

LEMMA 2. *Since the value of the quantity inside the parentheses in (3) is always less than one, an upper triangular matrix $B \in \mathbf{R}^{n\times n}$ with a constant diagonal satisfies condition (3).*

EXAMPLE 1. *The columns of this triangular matrix $\widehat{B} \in \mathbf{R}^{n\times n}$ form a reduced basis:*

$$\widehat{B} = \begin{bmatrix} 1 & -0.5 & -0.5 & \cdots & \cdots & -0.5 \\ & 1 & -0.5 & \cdots & \cdots & -0.5 \\ & & 1 & \ddots & \cdots & -0.5 \\ & & & \ddots & \ddots & \vdots \\ & & & & 1 & -0.5 \\ & & & & & 1 \end{bmatrix}. \tag{4}$$

The matrix is very ill-conditioned, for Luk and Tracy[1] *show that its condition number increases like $(1.5)^{n-2}/2$.*

Let us describe the actions of the LLL algorithm by showing how conditions (2) and (3) are enforced. Condition (2) is easy to impose on $U \equiv (u_{i,j})$, an upper triangular matrix with a unit diagonal. We begin by defining elementary unimodular transformation. Let $i < j$, and let $\mathbf{e}_i \in \mathbf{Z}^n$ and $\mathbf{e}_j \in \mathbf{Z}^n$ denote the unit coordinate vectors in the i-th and j-th directions, respectively. Define $M_{ij} \in \mathbf{Z}^{n\times n}$ by

$$M_{ij} \equiv I - \gamma \mathbf{e}_i \mathbf{e}_j^{\mathrm{T}}, \tag{5}$$

where γ is an integer.

LEMMA 3. *The matrix M_{ij} defined in (5) is an integer unimodular transformation.*

We use M_{ij} to ensure that the (i,j)-th element of U is sufficiently small. Suppose that (2) is not satisfied for some i and j; that is, $|u_{i,j}| > 0.5$. Calculate γ as the integer closest to $u_{i,j}$:

$$\gamma = \lceil u_{i,j} \rfloor. \tag{6}$$

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

$$B \leftarrow BM_{ij} \quad \text{and} \quad U \leftarrow UM_{ij}. \tag{7}$$

The (i,j)-th element of the new U satisfies (2). We summarize the actions to enforce (2) in the next procedure.

PROCEDURE DECREASE(i,j) *Given B and U, calculate M_{ij} and γ using (5) and (6), respectively. Apply M_{ij} to B and U:*

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

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

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

$$X_i \equiv \begin{bmatrix} I_{i-2} & & & \\ & \mu & 1-\xi\mu & \\ & 1 & -\xi & \\ & & & I_{n-i} \end{bmatrix}. \tag{8}$$

For condition (3), we use the two numerical transformations defined in the two notations. Note that

$$\det(X_i) = -1, \tag{9}$$

and that X_i^{-1} is given by

$$X_i^{-1} = \begin{bmatrix} I_{i-2} & & & \\ & \xi & 1-\xi\mu & \\ & 1 & -\mu & \\ & & & I_{n-i} \end{bmatrix}. \tag{10}$$

The matrix X_i^{-1} is made up of a product of two Gauss transformations; 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:

$$\begin{bmatrix} \xi & 1-\xi\mu \\ 1 & -\mu \end{bmatrix} \begin{bmatrix} 1 & \mu \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} = \begin{bmatrix} 1 & \xi \\ 0 & 1 \end{bmatrix}. \tag{11}$$

In words, equation (11) says that the matrix X_i^{-1} restores the triangularity of a permuted triangular matrix. Note that both upper triangular matrices in (11) have ones on their diagonals.

Suppose that the (3) 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 those of U:

$$B \leftarrow B\Pi_i \quad \text{and} \quad U \leftarrow U\Pi_i. \tag{12}$$

We then use the transformation X_i^{-1} of (10) to restore U to triangular form:

$$U \leftarrow X_i^{-1} U. \tag{13}$$

Lenstra et al.[2] give the formulas that are used to update the squares of the diagonal elements d_{i-1} and d_i of D. Specifically,

$$\hat{d}_{i-1}^2 = d_i^2 + \mu^2 d_{i-1}^2 \quad \text{and} \quad \hat{d}_i^2 = (d_i^2 d_{i-1}^2)/\hat{d}_{i-1}^2, \tag{14}$$

where $\hat{d}_{i-1}$ and $\hat{d}_i$ are the new diagonal elements. The paper[2] also gives the values of ξ and μ in (8). As is obvious from (11), μ is given by

$$\mu = u_{i-1,i}. \tag{15}$$

In addition, ξ is given by

$$\xi = \mu\, d_{i-1}^2/(d_i^2 + \mu^2 d_{i-1}^2). \tag{16}$$

The actions to enforce (3) are written out in the next procedure.

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

$$D^2 \leftarrow D_{new}^2, \quad B \leftarrow B\Pi_i, \quad \textit{and} \quad U \leftarrow X_i^{-1} U \Pi_i. \tag{17}$$

The matrix D_{new}^2 is obtained by (14) and X_i^{-1} is computed by the equations (10), (15), and (16).

Luk and Tracy[1] use the two procedures, Decrease and Swap, to construct an algorithmic description of the LLL algorithm. The original LLL paper[2] contains a proof of convergence, but not an algorithmic description, of the method. It is fair to say that the algorithmic description inspired Luk and Tracy[1] to derive their new implementation. Although the LLL algorithm has shown to be an effective tool[3] to reduce the condition number of most ill-conditioned matrices that occur in practice, it does not modify the ill-conditioned matrix $\widehat{B}$ of (4) because its columns already form a reduced basis.

ALGORITHM LLL *Given B, transform its columns so that they will form a reduced basis.*

compute the QR decomposition of B to get D^2 and U;
set $k \leftarrow 2$;
while $k \leq n$
  if $|u_{k-1,k}| > 0.5$ then DECREASE$(k-1,k)$;
  if $d_k^2 < (\omega - u_{k-1,k}^2) d_{k-1}^2$ then
    SWAP(k);
    $k \leftarrow \max(k-1, 2)$;
  else
    for $i = k-2$ down to 1
      if $|u_{i,k}| > 0.5$ then DECREASE(i,k);
    $k \leftarrow k+1$.

3. A NEW IMPLEMENTATION

Luk and Tracy[1] extend the idea of a reduced basis formed by column vectors to that of a *reduced* triangular matrix. Let $B \in \mathbf{R}^{n\times n}$ be nonsingular. Consider its QR decomposition:

$$Q^{\mathrm{T}} B = R, \tag{18}$$

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

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

This extension[1] leads to a new algorithm to transform a given matrix B to a reduced triangular matrix R.

DEFINITION 3. *The columns of B form a reduced basis if*

$$r_{i,i} \geq 2|r_{i,j}|, \quad \textit{for } 1 \leq i < j \leq n, \tag{19}$$

and

$$r_{i,i}^2 \geq [\omega - (r_{i-1,i}/r_{i-1,i-1})^2] r_{i-1,i-1}^2, \quad \textit{for } 2 \leq i \leq n, \tag{20}$$

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

DEFINITION 4. *An upper triangular matrix R is reduced if its elements satisfy the conditions (19) and (20).*

PROPOSITION 1. *Given $B \in \mathbf{R}^{n\times n}$, the 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:*

$$Q^{\mathrm{T}} B M = R, \tag{21}$$

so that R is reduced. The columns of BM form a reduced basis as defined in the LLL paper.

The new approach[1] enforces conditions (19) and (20). While condition (19) states that any diagonal element of R is at least twice as large as any other element of R along the same row, condition (20) states that the diagonal elements of R must be *ordered* in a certain way. We use M_{ij} of (5) to ensure that the (i,j)-th element of R is sufficiently small relative to $r_{i,i}$. Suppose that (19) is not satisfied for some i and j; that is,

$$r_{i,i} < 2|r_{i,j}|.$$

Calculate γ as the integer closest to $r_{i,j}/r_{i,i}$:

$$\gamma = \lceil r_{i,j}/r_{i,i} \rfloor. \tag{22}$$

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

$$R \leftarrow R M_{ij}, \tag{23}$$

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 (23) satisfies (19). For condition (20) we need to use a plane reflection[4] (a basic numerical tool that is closely related to the more familiar plane rotation).

PROCEDURE NEWDECREASE(i,j) *Given R and M, calculate M_{ij} and γ using (5) and (22), respectively, and apply M_{ij} to both R and M:*

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

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

$$J_i \equiv \begin{bmatrix} I_{i-2} & & & \\ & c & s & \\ & s & -c & \\ & & & I_{n-i} \end{bmatrix}, \tag{24}$$

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

Note that

$$\det(J_i) = -1, \tag{25}$$

just as $\det(X_i) = -1$ in (9). Luk and Tracy[1] use plane reflections instead of plane rotations because the X_i's are closely related to plane reflections, as will be seen in the next section. Suppose that (20) is not satisfied for some i:

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

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

$$R \leftarrow R\Pi_i, \tag{26}$$

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

$$R \leftarrow J_i R. \tag{27}$$

We accumulate the transformations in M and Q:

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

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

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

$$R \leftarrow J_i R\Pi_i, \quad M \leftarrow M\Pi_i \quad \textit{and} \quad Q \leftarrow QJ_i. \tag{28}$$

ALGORITHM NEW

compute $B = QR$;
set $M \leftarrow I$ and $k \leftarrow 2$;
while $k \le n$
    if $r_{k-1,k-1} < 2|r_{k-1,k}|$ then NEWDECREASE$(k-1,k)$;
    if $r_{k,k}^2 < [\omega - (r_{k-1,k}/r_{k-1,k-1})^2] r_{k-1,k-1}^2$ then
        NEWSWAP(k);
        $k \leftarrow \max(k-1, 2)$;
    else
        for $i = k-2$ down to 1
            if $r_{i,i} < 2|r_{i,k}|$ then NEWDECREASE(i,k);
        $k \leftarrow k+1$.

4. EQUIVALENCE RESULT

There are many similarities between Algorithms New and LLL. Both algorithms aim to reduce the given matrix B to a triangular form. A major difference lies in the transformations used. Algorithm New applies plane reflections J_i of (24) directly to R, while Algorithm LLL applies special transformations X_i^{-1} of (10) to U and D^2 separately. A significant result[1] is that the two transformations are related via

$$J_i = D_1 X_i^{-1} D_2, \tag{29}$$

where D_1 and D_2 are $n \times n$ diagonal matrices. Thus, we may view X_i^{-1} as a scaled plane reflection. Luk and Tracy[1] show that in exact arithmetic, the two algorithms produce identical numerical results.

Representing the effect of transformations (26) and (27) by

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

we write out the key 2-by-2 transformations as follows:

$$\begin{bmatrix} \hat{\alpha} & \hat{\gamma} \\ 0 & \hat{\beta} \end{bmatrix} = \begin{bmatrix} c & s \\ s & -c \end{bmatrix} \begin{bmatrix} \alpha & \gamma \\ 0 & \beta \end{bmatrix} \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}. \tag{30}$$

Define a new transformation Y by

$$Y \equiv \begin{bmatrix} 1/\hat{\alpha} & 0 \\ 0 & 1/\hat{\beta} \end{bmatrix} \begin{bmatrix} c & s \\ s & -c \end{bmatrix} \begin{bmatrix} \alpha & 0 \\ 0 & \beta \end{bmatrix}. \tag{31}$$

If we choose

$$\xi = \hat{\gamma}/\hat{\alpha} \quad \text{and} \quad \mu = \gamma/\alpha, \tag{32}$$

then we get[1]

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

and

$$\begin{bmatrix} 1 & \xi \\ 0 & 1 \end{bmatrix} = Y \begin{bmatrix} 1 & \mu \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}. \tag{33}$$

Note that (33) is exactly equation (11) for the LLL method. Also, we can easily prove that the μ and ξ as defined in (32) have the same values as the μ and ξ as defined in (15) and (16). Thus, the transformation Y of (31) is exactly the 2×2 part of the workhorse X_i^{-1} of the LLL algorithm. Let

$$D \equiv \begin{bmatrix} E_1 & & & \\ & \alpha & 0 & \\ & 0 & \beta & \\ & & & E_2 \end{bmatrix}, \tag{34}$$

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

$$D_1 \equiv \begin{bmatrix} E_1 & & & \\ & \hat{\alpha} & 0 & \\ & 0 & \hat{\beta} & \\ & & & E_2 \end{bmatrix} \quad \text{and} \quad D_2 \equiv \begin{bmatrix} E_1^{-1} & & & \\ & 1/\alpha & 0 & \\ & 0 & 1/\beta & \\ & & & E_2^{-1} \end{bmatrix}. \tag{35}$$

Then

$$J_i = D_1 X_i^{-1} D_2. \tag{36}$$

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 being transformed 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 is applicable to Algorithm New. The former algorithm is numerically more 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 efficiecy is a possible loss in numerical accuracy, as will be shown in the next section.

5. NUMERICAL PROPERTIES

As pointed out in the last section, a significant difference between Algorithms New and LLL is that New works directly on R while LLL works on U and D^2 individually. Put it simply, New computes $r_{i,i}$ while LLL calculates d_i^2. Consequently, Algorithm LLL is susceptible to underflow (respectively overflow) exceptions when the diagonal elements d_i's are small (respectively large). For our discussion, we assume standard IEEE floating-point arithmetic. In single precision, we would have minimum exponent value $e_{\min} = -126$ and maximum exponent value $e_{\max} = 127$. Due to the presence of denormals, a number x underflows if $|x| < 2^{-126-23} = 2^{-149}$, whereas the number x overflows if $|x| \geq 2^{128}$.

Even if the quantities d_i^2's are not small or large enough to cause exceptions, a straightforward implementation of the LLL algorithm could still result in errors. Let $\omega = 0.75$, and consider the following 2-by-2 upper triangular matrix

$$R = \begin{bmatrix} \alpha & \mu\,\alpha \\ 0 & \sqrt{0.5}\alpha \end{bmatrix}.$$

The condition (20) is not satisfied when $|\mu| < 0.5$. Recall the updating formula (14):

$$\hat{d}_i^2 = (d_i^2 d_{i-1}^2)/\hat{d}_{i-1}^2.$$

The numerator $(0.5\,\alpha^4)$ may readily underflow or overflow; for example, in single precision, an underflow would occur if

$$2^{-1}\,\alpha^4 < 2^{-149} \qquad \text{or} \qquad \alpha < 2^{-37} \approx 8 \times 10^{-12},$$

and an overflow would occur if

$$2^{-1}\,\alpha^4 \geq 2^{128} \qquad \text{or} \qquad \alpha \geq 2^{32.25} \approx 5 \times 10^{9},$$

Although it may be possible to avoid an exception in (14) by doing the division before the multiplication, we cannot apply the same technique to prevent a possible underflow in the calculation of the numerator in (16):

$$\xi = (\mu\, d_{i-1}^2)/(d_i^2 + \mu^2 d_{i-1}^2),$$

where small values of $|\mu|$ and α could cause the product $(\mu\,\alpha^2)$ to underflow.

As experiments, we programmed Algorithms LLL and New in Matlab, which supports IEEE double precision. In double precision, we would have minimum exponent value $e_{\min} = -1022$ and maximum exponent value $e_{\max} = 1023$. After both programs were run hundreds of times with identical random data input, we observed neither underflows nor overflows and the output results were numerically indistinguishable.

ACKNOWLEDGMENTS

This work is partially supported by Natural Sciences and Engineering Research Council of Canada.

REFERENCES

1. F. T. Luk and D. M. Tracy, "An improved LLL algorithm," *Linear Algebra and Its Applications* , pp. x–x, to appear in 2007.
2. A. Lenstra, H. Lenstra, and L. Lovasz, "Factoring polynomials with rational coefficients," *Mathematicsche Annalen* **261**, pp. 515–534, 1982.
3. B. Hassibi and H. Vikalo, "On the sphere-decoding algorithm i: Expected complexity," *IEEE Transactions on Signal Processing* **53**, pp. 2806–2818, 2005.
4. G. Golub and C. V. Loan, *Matrix Computations, 3rd Ed.*, The Johns Hopkins University Press, Baltimore, MD, 1996.