Fast LLL-Type Lattice Reduction.

Claus Peter Schnorr

Fachbereiche Mathematik/Biologie-Informatik, Universität Frankfurt, PSF 111932, D-60054 Frankfurt am Main, Germany. schnorr@cs.uni-frankfurt.de

25. October, 2004

Abstract. We modify the concept of LLL-reduction of lattice bases in the sense of Lenstra, Lenstra, Lovász [LLL82] towards a faster reduction algorithm. We organize LLL-reduction in segments of the basis.

Our SLLL-bases approximate the successive minima of the lattice in nearly the same way as LLL-bases. For integer lattices of dimension n given by a basis of length $2^{O(n)}$, SLLL-reduction runs in $O(n^{5+\varepsilon})$ bit operations for every $\varepsilon > 0$, compared to $O(n^{7+\varepsilon})$ for the original LLL and to $O(n^{6+\varepsilon})$ for the LLL-algorithms of Schnorr (1988) and Storjohann (1996). We present an even faster algorithm for SLLL-reduction via iterated subsegments running in $O(n^3 \log n)$ arithmetic steps.

Keywords. LLL-reduction, SLLL-reduction, length defect, segments, local LLL-reduction, Householder reflection, floating point errors, error bounds.

Abbreviated Title. Fast LLL-Lattice Reduction.

1 Introduction.

The set of all linear combinations with integer coefficients of a set of linearly independent vectors $\boldsymbol{b}_1, ..., \boldsymbol{b}_n \in \mathbb{R}^d$ is a *lattice* of *dimension* n with *basis* $\boldsymbol{b}_1, ..., \boldsymbol{b}_n$. The problem of finding a shortest, nonzero lattice vector is a landmark problem in complexity theory. This problem is polynomial time for fixed dimension n due to [Le83, LLL82] and is NP-hard for varying n [E81, Aj98, Mi98]. The famous LLL-algorithm of Lenstra, Lenstra, Lovász [LLL82] for lattice basis reduction is a ground breaking technique for solving important problems in algorithmic number theory, integer optimization, diophantine approximation and cryptography, for a few recent applications see [BN00, Bo00, Co97,Co01,NS00,BM03,Ma03] and [Lo86,MG02,S04] for background. We refer to integer lattices of dimension n contained in $\mathbb{Z}^d$, $d = O(n)$, given by a lattice basis of vectord of Euclidean length M_0. Throughout the introduction we assume that $M_0 = 2^{O(n)}$. Lattice reduction decreases the length of such input bases by at most a factor $2^{O(n)}$.

Performance of the original LLL-algorithm [LLL82]. The **LLL** performs $O(n^5)$ arithmetic steps using $O(n^2)$-bit integers. Approximating the shortest lattice vector to within *length defect* c means to find a nonzero lattice vector with at most c-times the minimal possible length. The LLL achieves for arbitrary $\varepsilon > 0$ length defect $(\frac{4}{3} + \varepsilon)^{n/2}$. It repeatedly constructs short bases in two-dimensional lattices, the two-dimensional problem was already solved by Gauss [Ga801].

Finding very short lattice vectors. Finding very short lattice vectors requires additional search beyond LLL-type reduction. The algorithm of KANNAN [K83] finds the shortest lattice vector in $n^{O(n)}$ steps. The improved algorithm of HELFRICH [He85] runs in $n^{\frac{n}{2}+o(n)}$ steps. The recent probabilistic sieve algorithm of [AKS01] runs in $2^{O(n)}$ average time and space, but is impractical as the exponent $O(n)$ is about $30\,n$. SCHNORR [S87] has generalized the LLL-algorithm by repeated construction of short lattice bases of dimension $2k \geq 2$. $2k$-reduction [S87] runs in $O(n^3 k^{k+o(k)} + n^4)$ arithmetic steps achieving *length defect* $(2k)^{n/k}$ The stronger BKZ-reduction [S87, SE91] is quite efficient for $k \leq 20$ but lacks a proven time bound. LLL-reduction is the case $k = 1$ of $2k$-reduction. Recently, AJTAI [Aj03] proves a complexity lower bound for $2k$-reduction that matches the proven time bound of [S87] up to a constant factor in the exponent. By random sampling of short lattice vectors SCHNORR [S03] achieves under heuristic assumptions in $O(n^3 k^k + n^4)$ steps length defect $(k/6)^{n/8k}$, the 8-th root of the length defect achievable in that time by $2k$-reduction [S87].

Floating point arithmetic. The **LLL** uses under exact integer arithmetic intermediate integers of bit length $O(n^2)$. This bit length can be reduced to $O(n)$ using floating point arithmetic (*fpa*, for short). The algorithm $\mathbf{LLL}_H$ of Section 3 compute intermediate vectors by a sequence of Householder reflections. This method is both practical and fully proven. It outperforms in practice the method of [SE91] and matches the proven time bound of the theoretic method of [S88]. $\mathbf{LLL}_H$ runs under *fpa* in $O(n^{6+\varepsilon})$ bit operations saving a factor n compared to the original **LLL**. We will combine this saving with another one from Segment LLL-reduction. Our time bounds assume fast multiplication of n-bit integers within $O(n^{1+\varepsilon})$ bit operations for every $\varepsilon > 0$.

Segment LLL-reduction in fpa. Segment LLL-reduction adapts LLL-reduction to a better use of local LLL-reduction. It improves the LLL-time bound and approximates the successive minima in nearly the same way as the **LLL**. Following Schönhage [Sc84] we partition a basis $\boldsymbol{b}_1, \ldots, \boldsymbol{b}_n$ of dimension $n = k\,m$ into m segments of k consecutive basis vectors. LLL-swaps are done using local coordinates of dimension $2k$ of two adjacent segments. Local LLL-swaps cost merely $O(k^2)$ arithmetic steps, local size-reduction included — compared to $O(n^2)$ steps for a global LLL-swap. We design Segment LLL-reduction as to minimize the number of local LLL-reductions. In Section 4 we present our basic $\mathbf{SLLL}_0$-algorithm that runs in $O(n^4)$ arithmetic steps, compared to $O(n^5)$ steps of the original **LLL**. It uses integers and *fpa* numbers of bit length $O(n^2)$. The refined algorithm **SLLL** of Section 5 decreases this bit length to $O(n)$ performing $O(n^4 \log n)$ arithmetic steps. **SLLL** runs under *fpa* in $O(n^{5+\varepsilon})$ bit operations, compared to $O(n^{7+\varepsilon})$ for the original **LLL** and $O(n^{6+\varepsilon})$ bit operations for $\mathbf{LLL}_H$, the LLL-algorithms of [S88] and [St96], and the semi-reduction of [Sc84]. In Section 6 we speed up SLLL-reduction by extending LLL-steps iteratively to larger and larger segments. The algorithm $\mathbf{SLLL}^+$ runs in $O(n^3 \log n)$ arithmetic steps.

Space efficiency. **SLLL** runs in linear space $O(nd \log_2 M_0)$ and input bases of length M_0 fit into space $nd \log_2 M_0$. The LLL-algorithms of [S88] and $\mathbf{LLL}_H$ of section 3 are also linear in space, while the original **LLL** of [LLL82] and the algorithms of [Sc84], [St96] expand the space of the input by a factor $O(n)$. The recent Hermite reduction algorithm of [Mi01] is also in linear space but is much slower than **SLLL** requiring $O(n^5(\log_2 M_0)^2)$ arithmetic steps.

Related work. Schönhage's [Sc84] concept of semi-reduction achieves length defect 2^n and runs in $O(n^4)$ arithmetic steps using $O(n^2)$-bit integers. STORJOHANN [St96] proposes an LLL-algorithm that replaces size-reduction by modular reduction, the Gram-Schmidt coefficients are reduced modulo a squared determinant of order M_0^n. This *modular* LLL uses matrix multiplication as a core subroutine. If multiplication of $n \times n$-matrices runs in $O(n^\beta)$ arithmetic steps it requires $O(n^{\beta+1})$ arithmetic steps using $O(n \log_2 M_0)$-bit integers. The drawback is the bit length $O(n \log_2 M_0)$ of integers. We are not aware of an LLL-code that uses long integers as proposed in [LLL82, St96] and performs for moderately large n and M_0. [St96, Thm 24] accelerates semi-reduction of [Sc84] by modular reduction via fast matrix multiplication to run in $O(n^{5+\frac{1}{5-\beta}+\varepsilon})$ bit operations. **SLLL** beats this time bound even for the unlikely value $\beta = 2$ and achieves the smaller length defect $(\frac{4}{3}+\varepsilon)^{n/2}$ for every $\varepsilon > 0$. MEHROTRA AND LI [ML01] combine our previous segment LLL-reduction [KS01a] with modular reduction to run in $O(n^{3.5})$ arithmetic steps using $O(n \log_2 M_0)$ bit integers, and thus running in $O(n^{5.5+\varepsilon})$ bit operations.

DAUDÉ AND VALLÉE [DV94] and AKHAVI [Ak02] study random input bases consisting of real vectors $\boldsymbol{b}_1, ..., \boldsymbol{b}_n$ that are independently drawn from the unit ball in $\mathbb{R}^n$. LLL-reduction performs for such random input bases on average $O(n^4 \log_2 n)$ arithmetic steps using real numbers [DV94]. Size-reduction of such a random basis achieves length defect $(\frac{4}{3})^{(n-1)/2}$ with high probability [Ak02]. The present paper continues and revises the reports [KS01a, KS01b, KS02].

2 LLL Reduction of Lattice Bases.

Notation. Let $\mathbb{R}^d$ be the real vector space of dimension d with standard *inner product* $\langle \boldsymbol{x}, \boldsymbol{y} \rangle = \boldsymbol{x}^t \boldsymbol{y}$. A vector $\boldsymbol{b} \in \mathbb{R}^d$ has *length* $\|\boldsymbol{b}\| = \langle \boldsymbol{b}, \boldsymbol{b} \rangle^{\frac{1}{2}}$. An ordered set of linearly independent vectors $\boldsymbol{b}_1, ..., \boldsymbol{b}_n \in \mathbb{R}^d$ is a *basis* of the *lattice* $\mathcal{L} = \sum_{i=1}^n \boldsymbol{b}_i \mathbb{Z} \subset \mathbb{R}^d$ of *dimension* $\dim \mathcal{L} = n$, consisting of all integer linear combinations of $\boldsymbol{b}_1, ..., \boldsymbol{b}_n$. We identify the basis with the matrix $B = [\boldsymbol{b}_1, ..., \boldsymbol{b}_n] \in \mathbb{R}^{d \times n}$, we write $\mathcal{L} = \mathcal{L}(B) = \mathcal{L}(\boldsymbol{b}_1, ..., \boldsymbol{b}_n)$. All vectors will be column vectors. Let $\boldsymbol{q}_i$ denote the component of $\boldsymbol{b}_i$ that is orthogonal to $\boldsymbol{b}_1, ..., \boldsymbol{b}_{i-1}$, $\boldsymbol{q}_1 = \boldsymbol{b}_1$. The *orthogonal vectors* $\boldsymbol{q}_1, ..., \boldsymbol{q}_n \in \mathbb{R}^d$ and the *Gram-Schmidt coefficients* $\mu_{j,i}$, $1 \le i, j \le n$ of the basis $\boldsymbol{b}_1, ..., \boldsymbol{b}_n$ satisfy for $j = 1, ..., n$:

$$\boldsymbol{b}_j = \textstyle\sum_{i=1}^j \mu_{j,i} \boldsymbol{q}_i, \qquad \mu_{j,j} = 1, \qquad \mu_{j,i} = 0 \text{ for } i > j.$$

$$\mu_{j,i} = \langle \boldsymbol{b}_j, \boldsymbol{q}_i \rangle / \langle \boldsymbol{q}_i, \boldsymbol{q}_i \rangle, \qquad \langle \boldsymbol{q}_j, \boldsymbol{q}_i \rangle = 0 \text{ for } j \neq i.$$

The geometric normal form (GNF) *of a basis.* The basis $B \in \mathbb{R}^{d \times n}$ has a unique decomposition $B = QR$, where $Q \in \mathbb{R}^{d \times n}$ has pairwise orthogonal columns of length 1, and $R = [r_{i,j}] \in \mathbb{R}^{n \times n}$ is upper-triangular with positive diagonal entries, $r_{i,j} = 0$ for $i > j$ and $r_{1,1}, ..., r_{n,n} > 0$. Hence $Q = [\, \boldsymbol{q}_1/\|\boldsymbol{q}_1\|, ..., \boldsymbol{q}_n/\|\boldsymbol{q}_n\| \,]$, $\mu_{j,i} = r_{i,j}/r_{i,i}$, and $\|\boldsymbol{q}_i\| = r_{i,i}$. Two bases $B = QR$, $\bar{B} = \bar{Q}\bar{R}$ are isometric iff

$R = \bar{R}$, or equivalently iff $B^t B = \bar{B}^t \bar{B}$. We call R the *geometric normal form* (GNF) of the basis, $\mathrm{GNF}(B) := R$.

The lattice $\mathcal{L} = \mathcal{L}(B)$ has *determinant* $\det \mathcal{L} = \det(B^t B)^{\frac{1}{2}} = \prod_{i=1}^n \|\boldsymbol{q}_i\|$, where B^t is the *transpose* and $B^t B$ is the *Gram matrix* of B. Let $\lceil r \rfloor = \lceil r - \frac{1}{2} \rceil$ denote the nearest integer to $r \in \mathbb{R}$. Let $\mathrm{col}(j, B)$ $(\mathrm{row}(j, B))$ denote the j-th column (j-th row) vector of the matrix B.

Duality. The *dual* of lattice $\mathcal{L} = \mathcal{L}(B)$ with basis $B \in \mathbb{R}^{d \times n}$ is the lattice

$$\mathcal{L}^* =_{\mathrm{def}} \{\boldsymbol{x} \in \mathrm{span}(\mathcal{L}) \mid \langle \boldsymbol{x}, \boldsymbol{y} \rangle \in \mathbb{Z} \text{ for all } \boldsymbol{y} \in \mathcal{L}\}$$

having determinant $\det \mathcal{L}^* = (\det \mathcal{L})^{-1}$. $\mathcal{L}^*$ has a basis $\bar{B} \in \mathbb{R}^{d \times n}$ satisfying $\bar{B}^t B = I_n$, where I_n is the $n \times n$ identity matrix. Inverting the order of the columns of $\bar{B} = [\bar{\boldsymbol{b}}_1, ..., \bar{\boldsymbol{b}}_n]$ yields the *dual basis* $B^* = [\boldsymbol{b}_1^*, ..., \boldsymbol{b}_n^*] = [\bar{\boldsymbol{b}}_n, ..., \bar{\boldsymbol{b}}_1]^{-1}$ of B satisfying $\langle \boldsymbol{b}_{n-i+1}^*, \boldsymbol{b}_j \rangle = \delta_{n-i+1,j}$ and $\|\boldsymbol{q}_i\| = \|\boldsymbol{q}_{n-i+1}^*\|^{-1}$ for $i = 1, ..., n$.

The successive minima. The j-th successive minimum λ_j of a lattice $\mathcal{L}$, $1 \leq j \leq \dim \mathcal{L}$, is the minimal real number r for which there exist j linearly independent lattice vectors of length bounded by r. λ_1 is the length of the shortest nonzero lattice vector. $\|\boldsymbol{b}_1\|/\lambda_1$ is the *length defect* of the basis.

Definition 1. *A basis* $\boldsymbol{b}_1, \ldots, \boldsymbol{b}_n \in \mathbb{Z}^d$ *with orthogonal vectors* $\boldsymbol{q}_1, ..., \boldsymbol{q}_n \in \mathbb{R}^d$ *is an* LLL-basis (or LLL-reduced) *for given* δ, $\frac{1}{4} < \delta \leq 1$, *if*

1. $|\mu_{j,i}| \leq \frac{1}{2}$ *for* $1 \leq i < j \leq n$,
2. $\delta \, \|\boldsymbol{q}_i\|^2 \leq \mu_{i+1,i}^2 \, \|\boldsymbol{q}_i\|^2 + \|\boldsymbol{q}_{i+1}\|^2$ *for* $i = 1, \ldots, n-1$.

A basis satisfying 1. is called *size-reduced.* For the rest of the paper LLL-reduction refers to given δ, $\alpha := 1/(\delta - 1/4)$. A.K. Lenstra, H.W. Lenstra, Jr. and L. Lovász [LLL82] introduced LLL-bases focusing on $\delta = 3/4$ and $\alpha = 2$.

Theorem 1 (LLL82). *Every LLL-basis* $\boldsymbol{b}_1, \ldots, \boldsymbol{b}_n \in \mathbb{Z}^d$ *with orthogonal vectors* $\boldsymbol{q}_1, ..., \boldsymbol{q}_n \in \mathbb{R}^d$ *of lattice* $\mathcal{L}$ *satisfies*

1. $\|\boldsymbol{q}_i\|^2 \leq \alpha^{j-i} \, \|\boldsymbol{q}_j\|^2$ *and* $\|\boldsymbol{b}_i\|^2 \leq \alpha^{j-1} \, \|\boldsymbol{q}_j\|^2$ *for* $1 \leq i \leq j \leq n$,
2. $\|\boldsymbol{b}_1\| \leq \alpha^{\frac{n-1}{4}} (\det \mathcal{L})^{\frac{1}{n}}$,
3. $\alpha^{-j+1} \leq \|\boldsymbol{q}_j\|^2 \lambda_j^{-2} \leq \|\boldsymbol{b}_j\|^2 \, \lambda_j^{-2} \leq \alpha^{n-1}$ *for* $j = 1, ..., n$.

The inequalities (1), (3) of Theorem 1 follow by the argument of Theorem 6.

Size measures. We call $M_0 =_{\mathrm{def}} \max(\|\boldsymbol{b}_1\|, ..., \|\boldsymbol{b}_n\|)$ the *length* of the basis $B = [\boldsymbol{b}_1, ..., \boldsymbol{b}_n] \in \mathbb{Z}^{d \times n}$ and $M =_{\mathrm{def}} \max(d_1, ..., d_n, 2^n)$ the *volume* of the basis, where $d_i := \det(\mathcal{L}(\boldsymbol{b}_1, ..., \boldsymbol{b}_i))^2 = \|\boldsymbol{q}_1\|^2 \cdots \|\boldsymbol{q}_i\|^2$. We use a novel measure for bounding the *length defect* of a basis: $M_1 =_{\mathrm{def}} \max_{1 \leq i \leq j \leq n} \|\boldsymbol{q}_i\|/\|\boldsymbol{q}_j\|$. The argument of Theorem 6 shows that every size-reduced basis satisfies

$$\tfrac{4}{j+3}/M_1^2 \leq \|\boldsymbol{b}_j\|^2/\lambda_j^2 \leq M_1^2 \, \tfrac{j+3}{4} \quad \text{for } j = 1, ..., n.$$

By Theorem 1, $M_1^2 \leq \alpha^{n-1}$ holds for LLL-bases. A basis B and its dual B^* have the same M_1-value. Lattice reduction aims at a lattice basis with small M_1-value. Clearly, $d_i \leq M_0^{2i}$, $M \leq M_0^{2n}$ and $M^{-1} \leq \|\boldsymbol{q}_i\|^2 \leq M$, and thus $M_1 \leq M$ follows from $\|\boldsymbol{q}_i\|^2 = d_i/d_{i-1}$. We let M_0 refer to the input basis of an algorithm. M and M_1 do not increase during LLL-reduction. $M_1, M = 2^{O(n^2)}$ holds for every basis of length $2^{O(n)}$. We present the main steps of the LLL-algorithm, see [LLL82] for more details.

LLL

`INPUT` $\boldsymbol{b}_1, \dots, \boldsymbol{b}_n \in \mathbb{Z}^d$ (a basis with M_0, M), δ, $\frac{1}{4} < \delta < 1$

`OUTPUT` $\boldsymbol{b}_1, \dots, \boldsymbol{b}_n$ LLL-basis

1. $l := 1$ # $\boldsymbol{b}_1, \dots, \boldsymbol{b}_{l'}$ *is always an LLL-basis for* $l' := \max(l-1, 1)$.
2. `WHILE` $l \le n$ `DO`

   compute the rational numbers $\mu_{l,1}, ..., \mu_{l,l-1}$ and $\|\boldsymbol{q}_l\|^2$

   #*size-reduce* $\boldsymbol{b}_l$ *against* $\boldsymbol{b}_{l-1}, ..., \boldsymbol{b}_1$:

   `FOR` $i = l-1, ..., 1$ `DO` $\boldsymbol{b}_l := \boldsymbol{b}_l - \lceil \mu_{l,i} \rfloor \boldsymbol{b}_i$, update $\mu_{l,i}, ..., \mu_{l,1}$

   `IF` $l \ne 1$ `and` $\delta \|\boldsymbol{q}_{l-1}\|^2 > \mu_{l,l-1}^2 \|\boldsymbol{q}_{l-1}\|^2 + \|\boldsymbol{q}_l\|^2$

   `THEN` swap $\boldsymbol{b}_{l-1}, \boldsymbol{b}_l$, $l := l-1$ `ELSE` $l := l+1$.

LLL-time bound. One round of the `WHILE`-loop, i.e., one LLL-swap of $\boldsymbol{b}_{l-1}, \boldsymbol{b}_l$ requires $O(nd)$ arithmetic steps, size-reduction of $\boldsymbol{b}_l$ and computation of the rationals $\mu_{l,1}, ..., \mu_{l,l-1}, \|\boldsymbol{q}_l\|^2$ included. Given an integer basis in $\mathbb{Z}^d$ of length $M_0 \ge 2$ and volume M, **LLL** performs $O(n \log_{1/\delta} M) = O(n^2 \log_{1/\delta} M_0)$ LLL-swaps for $\delta < 1$, and runs in $O(n^2 d \log_{1/\delta} M)$ arithmetic steps using $O(\log_2(M_0 M))$-bit integers. Given a basis of length $2^{O(n)}$ and $d = O(n)$ this requires $O(n^{7+\varepsilon})$ bit operations for every $\varepsilon > 0$ because $\log_2(M_0 M) = O(n^2)$.

3 LLL Algorithm via Householder Reflections.

In this section we present the $\mathbf{LLL}_H$ variant of **LLL** which computes the $\mu_{l,i}$, $\|\boldsymbol{q}_l\|$ by a sequence of Householder reflections. We first analyse $\mathbf{LLL}_H$ in ideal real arithmetic, thereafter under floating point arithmetic. $\mathbf{LLL}_H$ under *fpa* saves a factor n in the number of bit operations compared to **LLL**. While the intermediate data $\mu_{l,i}$, $\|\boldsymbol{q}_l\|$ are computed in *fpa*, the basis vectors are in exact arithmetic. All subsequent reduction algorithm are based on $\mathbf{LLL}_H$.

Computing the GNF of $B = [\boldsymbol{b}_1, ..., \boldsymbol{b}_n]$. There is an extensive literature on numerical algorithms for computing the GNF R of the decomposition $B = QR$ of a basis B, see [LH95]. Householder algorithms and modified Gram-Schmidt orthogonalization are in our experience practically equivalent for the LLL. We use Householder reflection matrices because of the published *fpa*-error bounds.

We compute an orthogonal matrix $Q' \in \mathbb{R}^{d \times d}$ that extends Q and a matrix $R' \in \mathbb{R}^{d \times n}$ that extends R by zero-rows and allows that $\mathrm{col}(i, R') = \pm\, \mathrm{col}(i, R)$. In ideal arithmetic we get R' by a sequence of Householder transformations

$$R'_0 := B, \qquad R'_j := Q_j R'_{j-1} \text{ for } j = 1, ..., n,$$
$$R' := R'_n, \qquad Q' := Q_1 \cdots Q_n = Q_1^t \cdots Q_n^t,$$

where $Q_j := I_d - 2\|\boldsymbol{h}_j\|^{-2}\boldsymbol{h}_j\boldsymbol{h}_j^t \in \mathbb{R}^{d\times d}$ is orthogonal and symmetric, $\boldsymbol{h}_j \in \mathbb{R}^d$.

The transform $R'_j := Q_j R'_{j-1}$ zeroes the entries in positions $j+1$ through d of $\mathrm{col}(j, R'_{j-1})$, it *triangulates* $\mathrm{col}(j, R'_{j-1})$ so that $R'_j \in \mathbb{R}^{d\times n}$ is upper-triangular for the first j columns. The transform $\boldsymbol{x} \mapsto Q_j\boldsymbol{x}$ reflects $\boldsymbol{x}$ at the hyperplane that is orthogonal to the *Householder vector* $\boldsymbol{h}_j \in \mathbb{R}^d$ so that

$$Q_j\boldsymbol{h}_j = -\boldsymbol{h}_j, \qquad Q_j\boldsymbol{x} = \boldsymbol{x} \text{ for } \langle \boldsymbol{h}_j, \boldsymbol{x}\rangle = 0.$$

Setting $\boldsymbol{r} := (r_1, ..., r_d)^t := \mathrm{col}(j, R'_{j-1})$ and $z := \mathrm{sign}(r_j)(\sum_{i=j}^d r_i^2)^{\frac{1}{2}}$
we get $\boldsymbol{h}_j := (0, ..., 0, r_j + z, r_{j+1}, ..., r_d)^t/\sqrt{2zr_l + 2z^2}$.

Correctness. We have that $2\boldsymbol{h}_j\langle \boldsymbol{h}_j, \boldsymbol{r}\rangle\|\boldsymbol{h}_j\|^{-2} = 2\boldsymbol{h}_j \frac{zr_j+z^2}{2zr_j+2z^2} = \boldsymbol{h}_j$, and thus

$$Q_j\boldsymbol{r} = \boldsymbol{r} - \boldsymbol{h}_j = (r_1, ..., r_{j-1}, -z, 0, ..., 0)^t \in \mathbb{R}^d.$$

This shows that $Q_j\boldsymbol{r}$ is correctly triangulated and $\boldsymbol{h}_j$ is well chosen.

The sign of z is chosen as to maximize the denominator $2zr_j + 2z^2 = \|\boldsymbol{h}_j\|^2$ in Q_j. Clearly, $Q_j \cdots Q_1\boldsymbol{b}_j = \mathrm{col}(j, R') = -\mathrm{sign}(r_j)\mathrm{col}(j, R)$ because $\langle \boldsymbol{h}_j, \mathrm{col}(i, R')\rangle = 0$ for $i < j$. We abbreviate $\boldsymbol{r}_l := \mathrm{col}(l, R)$ for $R = \mathrm{GNF}(B)$. Since `TriCol` computes $\boldsymbol{r}_l$ from $\mathrm{col}(l, R')$ this extends $\boldsymbol{r}_l$ by $d - n$ zeroes.

`TriCol`$(\boldsymbol{b}_1, ..., \boldsymbol{b}_l, \boldsymbol{h}_1, ..., \boldsymbol{h}_{l-1}, \boldsymbol{r}_1, ..., \boldsymbol{r}_{l-1})$ (`TriCol`$_l$ for short)

\# `TriCol`$_l$ *computes* $\boldsymbol{h}_l$ *and* $\boldsymbol{r}_l := \mathrm{col}(l, R)$ *and size-reduces* $\boldsymbol{b}_l, \boldsymbol{r}_l$.

1. $\boldsymbol{r} = (r_1, ..., r_d)^t := \boldsymbol{b}_l/\|\boldsymbol{b}_l\|$
   \# *we normalize* $\|\boldsymbol{r}\|$ *and* $\|\boldsymbol{h_l}\|$ *to* 1.
2. `FOR` $j = 1, ..., l-1$ `DO` $\boldsymbol{r} := \boldsymbol{r} - 2\langle \boldsymbol{h}_j, \boldsymbol{r}\rangle\, \boldsymbol{h}_j$
3. $z := \mathrm{sign}(r_l)(\sum_{i=l}^d r_i^2)^{\frac{1}{2}}$, $\boldsymbol{h}_l := (0, ..., 0, r_l + z, r_{l+1}, ..., r_d)^t/\sqrt{2zr_l + 2z^2}$
4. $\boldsymbol{r}_l := -\mathrm{sign}(r_l)\, \|\boldsymbol{b}_l\|\, (r_1, ..., r_{l-1}, -z, 0, ..., 0)^t \in \mathbb{R}^d$
5. \# *size-reduce* $\boldsymbol{b}_l$ *against* $\boldsymbol{b}_{l-1}, ..., \boldsymbol{b}_1$ *and update* $\boldsymbol{r}_l$:
   `FOR` $i = l-1, ..., 1$ `DO` $\boldsymbol{b}_l := \boldsymbol{b}_l - \lceil r_{i,l}/r_{i,i}\rfloor \boldsymbol{b}_i$, $\boldsymbol{r}_l := \boldsymbol{r}_l - \lceil r_{i,l}/r_{i,i}\rfloor \boldsymbol{r}_i$.

The normalization simplifies the *fpa*-error analysis, but it is not essential. In step 4 we have $\mathrm{sign}(r_l)z > 0$, and thus upon termination we have that $r_{l,l} > 0$.

Step bound. `TriCol`$_l$ runs in $O(dl)$ arithmetic steps and one sqrt.

The LLL-algorithm in terms of $R = \mathrm{GNF}(B)$. Consider the diagonal submatrix $R_{l-1,1} = \begin{bmatrix} r_{l-1,l-1} & r_{l-1,l} \\ 0 & r_{l,l} \end{bmatrix} \subset R$ shown in Fig. 1. (We let $R' \subset R$ denote that R' is a submatrix of R, i.e., $r'_{i,j} = r_{i+k,j+m}$ for all i, j and some k, m.) $\mathbf{LLL}_H$ performs simultaneous column operations on R and B that shorten the first column of some $R_{l-1,1}$. It swaps columns $\boldsymbol{r}_{l-1}, \boldsymbol{r}_l$ and $\boldsymbol{b}_{l-1}, \boldsymbol{b}_l$ if this shortens the square length of the first column of $R_{l-1,l}$ by the factor δ. To enable a swap the entry $r_{l-1,l}$ is first reduced to $|r_{l-1,l}| \le \frac{1}{2}\,|r_{l-1,l-1}|$ by transforming $\boldsymbol{r}_l := \boldsymbol{r}_l - \lceil r_{l-1,l}/r_{l-1,l-1}\rfloor \boldsymbol{r}_{l-1}$. The ideal $\mathbf{LLL}_H$ algorithm reads

Fig. 1. The submatrix $R_{l-1,1} \subset R$

LLL$_H$

`INPUT` $\boldsymbol{b}_1, \ldots, \boldsymbol{b}_n \in \mathbb{Z}^d$ (a basis with M_0, M_1, M), δ, $\frac{1}{4} < \delta < 1$

`OUTPUT` $\boldsymbol{b}_1, \ldots, \boldsymbol{b}_n$ LLL-basis for δ

1. $l := 1$, # $\mathbf{b}_1, ..., \mathbf{b}_{\max(l-1,1)}$ *is always an LLL-basis*
2. `WHILE` $l \leq n$ `DO`

   `TriCol`$(\boldsymbol{b}_1, ..., \boldsymbol{b}_l, \boldsymbol{h}_1, ..., \boldsymbol{h}_{l-1}, \boldsymbol{r}_1, ..., \boldsymbol{r}_{l-1})$

   `IF` $l \neq 1$ `and` $\delta\, r^2_{l-1,l-1} > r^2_{l-1,l} + r^2_{l,l}$

   `THEN` swap $\boldsymbol{b}_{l-1}, \boldsymbol{b}_l$, $l := l - 1$ `ELSE` $l := l + 1$.

Correctness. At stage l we get $\boldsymbol{r}_l = \text{col}(l, R)$ of $R = \text{GNF}(B)$, and we have $\boldsymbol{r}_{l-1}$ from a previous stage. Using the coefficients $r_{l-1,l-1}, r_{l-1,l}, r_{l,l}$ **LLL**$_H$ correctly simulates **LLL** since $r^2_{i,j} = \mu^2_{j,i}\|\boldsymbol{q}_i\|^2$. The GNF $[\boldsymbol{r}_1, ...\boldsymbol{r}_l]$ of $[\boldsymbol{b}_1, ...\boldsymbol{b}_l]$ is preserved during simultaneous size-reduction of $\boldsymbol{r}_l$ and $\boldsymbol{b}_l$ in $\texttt{TriCol}_l$.

LLL$_H$ **using floating point arithmetic.** We use the *fpa* model of Wilkinson [Wi63]. There is no assumption by this model. We merely want to use proven *fpa*-error bounds. A *fpa* number with $t = 2t' + 1$ *precision bits* is of the form $\pm 2^e \sum_{i=-t'}^{t'} b_i 2^i$, where $b_i \in \{0, 1\}$ and $e \in \mathbb{Z}$. It has bit length $t+s+2$ for $|e| < 2^s$, two signs included. We denote the set of these numbers by $\mathbb{FL}_t$. Standard double length *fpa* has $t = 53$ precision bits, $t + s + 2 = 64$. Let $fl : \mathbb{R} \supset [-2^{2^s}, 2^{2^s}] \ni r \mapsto \mathbb{FL}_t$ approximate real numbers by *fpa* numbers. A step $c := a \circ b$ for $a, b, c \in \mathbb{R}$ and a binary operation $\circ \in \{+, -, \cdot, /\}$ translates under *fpa* into $\bar{a} := fl(a)$, $\bar{b} := fl(b)$, $\bar{c} := fl(\bar{a} \circ \bar{b})$, resp. into $\bar{a} := fl(\circ(\bar{a}))$ for unary operations $\circ \in \{\lceil\ \rfloor, \sqrt{\ }\}$. Each *fpa* operation induces a normalized relative error bounded in magnitude by 2^{-t}: $|fl(\bar{a} \circ \bar{b}) - \bar{a} \circ \bar{b}|/|\bar{a} \circ \bar{b}| \leq 2^{-t}$. If $|\bar{a} \circ \bar{b}| > 2^{2^s}$ or $|\bar{a} \circ \bar{b}| < 2^{-2^s}$ then $fl(\bar{a} \circ \bar{b})$ is undefined due to an *overflow*, resp. *underflow*.

It is common to require that $2^s \leq t^2$ and thus $s \leq 2 \log_2 t$, for brevity we identify the bit length of *fpa*-numbers with t, neglecting the minor $(s + 2)$-part.

Under *fpa* we let $\mathbf{LLL}_H$ use approximate vectors $\bar{\boldsymbol{h}}_l,\ \bar{\boldsymbol{r}}_l \in \mathbb{FL}_t^d$ and exact basis vectors in $\mathbb{Z}^d$.

`TriCol`$_l$ *under fpa.* A detailed discussion and analysis of steps 1-4 of `TriCol`$_l$ under *fpa* is in [LH95, chapter 15]. In order to keep *fpa*-errors small during the iteration of `TriCol`$_l$ within $\mathbf{LLL}_H$ we replace under *fpa* for the rest of the paper `TriCol`$_l$ by the following iterative

fpa-version of `TriCol`$_l$. Let $\varepsilon > 0$ be given as input.

Zero $\lceil \bar{r}_{i,l}/\bar{r}_{i,i} \rfloor$ in step 5 if $|\bar{r}_{i,l}/\bar{r}_{i,i}| < \frac{1}{2} + \varepsilon/2$ holds. Repeat steps 1-5 of the above `TriCol`$_l$-procedure in a loop until step 5 leaves $\boldsymbol{b}_l$ unchanged, i.e., $|\bar{r}_{i,l}/\bar{r}_{i,i}| < \frac{1}{2} + \varepsilon/2$ holds for $i = l-1, ..., 1$.

Zeroing of $\lceil \bar{r}_{i,l}/\bar{r}_{i,i} \rfloor$ cancels a size-reduction step and prevents cycling through steps 1-5. In `TriCol`$_l$'s last round size-reduction is void and the value of $\boldsymbol{r}_l$ in step 4 and its *fpa*-error remain unchanged.

The proof of Theorem 2 shows under *fpa* that `TriCol`$_l$ with $t = 5n + 2\log_2 M_0$ precision bits performs two rounds through steps 1-5, the first correctly size-reduces $\boldsymbol{b}_l$, the second decreases *fpa*-errors given that $\|\boldsymbol{b}_l\|$ is already small.

Given $\varepsilon > 0$ we set $\delta_- := \delta - \varepsilon,\ \delta_+ := \delta + \varepsilon \le 1 - \varepsilon$ and $\alpha_- := 1/(\delta - \varepsilon - 1/4)$.

Theorem 2. *Given a basis of length* M_0*,* $0 < \varepsilon < 0.02$ *and* $\delta \ge 0.96$*,* $\mathbf{LLL}_H$ *using* fpa *of* $t = 5n + 2\log_2 M_0$ *precision bits computes for* $n \ge n_0(\varepsilon)$ *an approximate LLL-basis for* δ_- *with* $\mu_{j,i}$ *and orthogonal vectors* $\boldsymbol{q}_1, ..., \boldsymbol{q}_n$ *satisfying*

1. $|\mu_{j,i}| < \frac{1}{2} + \varepsilon$ *for* $1 \le i < j \le n$,
2. $\delta_- \|\boldsymbol{q}_i\|^2 \ \le\ \mu_{i+1,i}^2\ \|\boldsymbol{q}_i\|^2 + \|\boldsymbol{q}_{i+1}\|^2$ *for* $i = 1, \ldots, n-1$.

$\mathbf{LLL}_H$ *runs under* fpa *in* $O(n^2 d \log_{1/\delta} M)$ *arithmetic steps using* $2n + 2\log_2 M_0$ *bit integers and* fpa *numbers of bit length* $3n + 2\log_2 M_0$.

In particular $\mathbf{LLL}_H$ runs for $M_0 = 2^{O(n)}$ in $O(n^4 d)$ arithmetic steps, i.e. for $d = O(n)$, in $O(n^{6+\varepsilon'})$ bit operations for every $\varepsilon' > 0$. If 1. of Theorem 2 holds we call the basis *size-reduced* under *fpa*.

Proof. The proof uses an *fpa*-version of Theorem 1 which will later be proved in Theorem 6. In particular, clauses 1 and 2 of Theorem 2 imply the inequalities

$$\alpha_-^{-j+1} \ \le\ \|\boldsymbol{b}_j\|^2\, \lambda_j^{-2} \ \le\ \alpha_-^{n-1} \quad \text{for } j = 1, ..., n,\ \ \alpha_- := 1/(\delta - \varepsilon - 1/4).$$

For all size bounds of intermediate data we neglect the effect of ε on α_-. For simplicity we assume that $\alpha_- = \alpha \le \sqrt{2}$ since $1/(0.96 - \frac{1}{4}) < \sqrt{2}$. We also neglect that the $|\mu_{j,i}|$ for $i < j$ can be larger than $\frac{1}{2}$ but less than $\frac{1}{2} + \varepsilon$.

Length of intermediate bases. We show in ideal arithmetic that all intermediate basis vectors have length $\le 2^n M_0$. We show that the claim holds during size-reduction within $\mathbf{LLL}_H$. A *size-reduction step* $\boldsymbol{b}_l := \boldsymbol{b}_l - \lceil \mu_{l,j} \rfloor \boldsymbol{b}_j$ for $j < l$ induces $\mu_{l,i} := \mu_{l,i} - \lceil \mu_{l,j} \rfloor \mu_{j,i}$ for $i = 1, ..., j$, where $\boldsymbol{b}_1, ..., \boldsymbol{b}_j$ is an LLL-basis. As $|\mu_{j,i}| \le \frac{1}{2}$ for $i < j$ this increases $\max_{i<l} |\mu_{l,i}|$ by at most a factor $\frac{3}{2}$ (the rounding to $\lceil \mu_{l,j} \rfloor$ can be neglected).

Consider the initial values $\boldsymbol{b}_l, \mu_{l,i}$ and the final values $\boldsymbol{b}'_l, \mu'_{l,i}$ after h size-reduction steps. We have $\mu'_{l,l} = 1,\ |\mu'_{l,i}| \le \frac{1}{2}$ for $l - h \le i < l$. For $i < l - h$ there exists by the above argument j, $h-l \le j < l$ such that

$$|\mu'_{l,i}|\,\|\boldsymbol{q}_i\| \le (\tfrac{3}{2})^h |\mu_{l,j}|\,\|\boldsymbol{q}_i\| \le (\tfrac{3}{2})^h \alpha^{\frac{j-i}{2}} |\mu_{l,j}|\,\|\boldsymbol{q}_j\|,$$

because $\|\boldsymbol{q}_i\| \le \alpha^{\frac{j-i}{2}} \|\boldsymbol{q}_j\|$ as $\boldsymbol{b}_1, ..., \boldsymbol{b}_{l-1}$ is LLL-reduced. From $\alpha \le \sqrt{2}$ we get

$$\|\boldsymbol{b}'_l\|^2 = \textstyle\sum_{i=1}^l |\mu'_{l,i}|^2 \|\boldsymbol{q}_i\|^2 \le l(\tfrac{3}{2})^{2h} 2^{l/2} \|\boldsymbol{b}_l\|^2 \le l \cdot 3.15^l \|\boldsymbol{b}_l\|^2.$$

Therefore, in ideal arithmetic all intermediate vectors $\boldsymbol{b}'_l$ have length $\le 2^l \|\boldsymbol{b}_l\|$ for $l \ge 10$. This also holds under *fpa* due to the following *fpa*-error analysis.

Correctness under fpa. We study $\texttt{TriCol}_l$ within the algorithms $\mathbf{LLL}_H$, $\mathbf{SLLL}_0$, **SLLL**. It is crucial that the Householder reflection matrices Q_i preserve the inner product, $\langle \boldsymbol{x}, \boldsymbol{y} \rangle = \boldsymbol{x}^t \boldsymbol{y} = \boldsymbol{x}^t Q_i^t Q_i \boldsymbol{y} = \langle Q_i \boldsymbol{x}, Q_i \boldsymbol{y} \rangle$, and thus Q_i preserves in ideal arithmetic the length of *fpa*-error vectors. $A_{n+1} := Q_n \cdots Q_1 B$ is computed under *fpa* recursively as $\bar{A}_1 := B$, $\bar{A}_{i+1} := fl(\bar{Q}_i \bar{A}_i)$ for $i = 1, ..., n$.

Proposition 1. $\|\bar{\boldsymbol{h}}_l - \boldsymbol{h}_l\| = O(d\, l\, 7^l 2^{-t}), \quad \|\bar{\boldsymbol{r}}_l - \boldsymbol{r}_l\| = O(d\, l\, 7^l 2^{-t} \|\boldsymbol{b}_l\|).$ (1)

Proof. We proceed by induction on l. We extend the error analysis of [LH95, pp.85,86]. Let $\boldsymbol{r}_l = \mathrm{col}(l, R)$, $\boldsymbol{b}_1 = (r_1, ..., r_d)^t$ and $z = \mathrm{sign}(r_1)(\sum_{i=1}^d r_i^2)^{\frac{1}{2}}$ then the errors of $\boldsymbol{r}_1 = (z, 0, ..., 0)^t$, $\boldsymbol{h}_1 = (r_1 + z, r_2, ..., r_d)^t$ are bounded as

$$\|\bar{\boldsymbol{r}}_1 - \boldsymbol{r}_1\| = O(d\,\|\boldsymbol{b}_1\|\,(2^{-t} + 2^{-2t})), \;\; \|\bar{\boldsymbol{h}}_1 - \boldsymbol{h}_1\| = O(d\,(2^{-t} + 2^{-2t})).$$

We will neglect all 2^{-2t}-terms. The first bound is obvious and implies the second, see (15.22), (15.23)[LH95]. $\texttt{TriCol}_l$ computes $\boldsymbol{r}_l$, $\boldsymbol{h}_l$ via

$$\boldsymbol{r}'_{l-1} := \textstyle\prod_{i=1,...,l-2} (1 - 2\boldsymbol{h}_i \boldsymbol{h}_i^t)\,(\boldsymbol{b}_l / \|\boldsymbol{b}_l\|) \text{ and } \boldsymbol{r}'_l := (1 - 2\boldsymbol{h}_{l-1} \boldsymbol{h}_{l-1}^t) \boldsymbol{r}'_{l-1}.$$

The induction hypothesis for $l-1$ yields $\|\bar{\boldsymbol{r}}'_{l-1} - \boldsymbol{r}'_{l-1}\| = O(d\,(l-1) 7^{l-1} 2^{-t})$. If $\bar{\boldsymbol{r}}'_l := (1 - 2\bar{\boldsymbol{h}}_{l-1} \bar{\boldsymbol{h}}_{l-1}^t) \bar{\boldsymbol{r}}'_{l-1}$ is computed in ideal arithmetic we have

$$\begin{aligned} \|\bar{\boldsymbol{r}}'_l - \boldsymbol{r}'_l\| &\le \|\bar{\boldsymbol{r}}'_{l-1} - \boldsymbol{r}'_{l-1}\| + 2 \cdot 2\, \|\bar{\boldsymbol{h}}_{l-1} - \boldsymbol{h}_{l-1}\| + 2\, \|\bar{\boldsymbol{r}}'_{l-1} - \boldsymbol{r}'_{l-1}\| \\ &= O(d\,(l-1) 7 \cdot 7^{l-1} 2^{-t}), \end{aligned}$$

where 2^{-2t}-terms are omitted. We used that $\|\bar{\boldsymbol{r}}'_{l-1}\|, \|\bar{\boldsymbol{h}}_{l-1}\| = 1 + o(1)$ for $t \ge 3n$. One factor 2 in $2 \cdot 2$ comes from the two occurences of $\bar{\boldsymbol{h}}_{l-1}$ in $\bar{\boldsymbol{r}}'_l$.

The computation of $\bar{\boldsymbol{r}}'_l$ from $\boldsymbol{r}'_{l-1}$ under *fpa* adds $O(d\, 2^{-t})$ to the error obtained in ideal arithmetic. Step 3 of $\texttt{TriCol}_l$ computes $\boldsymbol{h}_l$ from $\bar{\boldsymbol{r}}'_l$ so that

$$\|\bar{\boldsymbol{h}}_l - \boldsymbol{h}_l\| = \|\bar{\boldsymbol{r}}'_l - \boldsymbol{r}'_l\| + O(d 2^{-t}) = O(d\, l\, 7^l 2^{-t}).$$

This proves the first induction claim.

The computation of $\boldsymbol{r}_l$ from $\boldsymbol{r}'_l$ and $\boldsymbol{h}_l$ in Step 4 of $\texttt{TriCol}_l$ multiplies errors by $\|\boldsymbol{b}_l\|$ so we get the second claim $\|\bar{\boldsymbol{r}}_l - \boldsymbol{r}_l\| = O(d\, l\, 7^l 2^{-t} \|\boldsymbol{b}_l\|)$. □

Referring to the GNF $[\boldsymbol{r}_1, ..., \boldsymbol{r}_l]$ of $\texttt{TriCol}_l$'s input basis $\boldsymbol{b}_1, ..., \boldsymbol{b}_l$ we denote

$$\bar{M}_0 := M_0 / r_{1,1} \qquad \bar{M}_1 := \max_{i<l} r_{1,1} / r_{i,i}. \qquad (2)$$

We let M_0, M_1 refer to the LLL-input basis. We always have $\bar{M}_1 \le M_1$, where $\bar{M}_1 \le \alpha^{\frac{l-1}{2}}$ holds within $\mathbf{LLL}_H$ since $\boldsymbol{b}_1, ..., \boldsymbol{b}_{l-1}$ is LLL-reduced. We show that $\texttt{TriCol}_l$'s last round correctly computes $\bar{\mu}_{l,i} = \bar{r}_{i,l} / \bar{r}_{i,i}$ up to an $\varepsilon/2$-error. Using

(1), (2) and assuming the initial bound $\|\boldsymbol{r}_l\| \le M_0$ Step 4 of $\texttt{TriCol}_l$ yields

$$\|\bar{\boldsymbol{r}}_l - \boldsymbol{r}_l\|/r_{i,i} = O(d\,7^l \bar{M}_0 \bar{M}_1 2^{-t}) \quad \text{for } 1 \le i \le l-1. \tag{3}$$

We have $|\bar{\mu}_{l,i} - \mu_{l,i}| = |\bar{r}_{i,l}/\bar{r}_{i,i} - r_{i,l}/r_{i,i}| \le \|\bar{\boldsymbol{r}}_l - \boldsymbol{r}_l\|/r_{i,i} + \|\boldsymbol{r}_l\| |\bar{r}_{i,i}^{-1} - r_{i,i}^{-1}|$. We bound the dominating $\|\boldsymbol{r}_l\||\bar{r}_{i,i}^{-1} - r_{i,i}^{-1}|$-term, the minor $\|\bar{\boldsymbol{r}}_l - \boldsymbol{r}_l\|/r_{i,i}$-term is bounded by (3) and will be neglected. Consider the right-hand side factors of $\|\boldsymbol{r}_l\||\bar{r}_{i,i}^{-1} - r_{i,i}^{-1}| = (\|\boldsymbol{r}_l\|/r_{i,i})(|\bar{r}_{i,i} - r_{i,i}|/\bar{r}_{i,i})$. Applying (3) to a previous $\texttt{TriCol}_i$-execution we have $|\bar{r}_{i,i} - r_{i,i}|/r_{1,1} \le O(d\,7^i \bar{M}_0 2^{-t})$. Multiplication with $\|\boldsymbol{r}_l\|/r_{1,1} \le \frac{1}{2}\bar{M}_0$ and $r_{1,1}^2/r_{i,i}^2 \le \bar{M}_1^2$ shows that step 4 of $\texttt{TriCol}_l$ yields

$$|\bar{r}_{i,l}/\bar{r}_{i,i} - r_{i,l}/r_{i,i}| \le O(d\,7^l \bar{M}_0^2 \bar{M}_1^2 2^{-t}) \le \varepsilon/2, \tag{4}$$

where the last inequality holds for $t = \Omega(1) + \log_2(d\,7^l \bar{M}_0^2 \bar{M}_1^2/\varepsilon)$, e.g., for $\varepsilon = 0.02$, $d = n \ge 40$ and $t \ge 3.5n + 2\log_2(\bar{M}_0\bar{M}_1)$. In particular, (4) holds upon termination of $\texttt{TriCol}_l$ as the final size-reduction in step 5 is void. Within $\mathbf{LLL}_H$ we have that $\bar{M}_1 \le \alpha^{\frac{n-1}{2}} \le 2^{\frac{n-1}{4}}$. Hence, upon termination $\boldsymbol{b}_l$ is size-reduced for $t \ge 4n + 2\log_2 M_0$ and $n \ge n_0(\varepsilon)$, proving clause 1 of Theorem 2.

$\texttt{TriCol}_l$*'s first round.* We have shown that $\|\boldsymbol{b}_l\|$ increases during size-reduction to at most $2^l M_0$. Retracing this proof with a view on *fpa*-errors shows that $\|\bar{\boldsymbol{r}}_l - \boldsymbol{r}_l\|/r_{i,i}$ increases during size-reduction by at most a factor 2^l compared to (3). This is a straightforward exercise left to the reader. We offset the increased *fpa*-errors by n additional precision bits. Hence, using $t \ge 5n + 2\log_2 M_0$ precision bits $\texttt{TriCol}_l$'s first round correctly size-reduces $\boldsymbol{b}_l$ for $n \ge n_0(\varepsilon)$, and $\texttt{TriCol}_l$ terminates in the second round.

Correct swapping. We see from (3) that the *fpa*-error of $r_{l,l}$ is bounded by $O(r_{1,1} d\,7^l \bar{M}_0 \bar{M}_1 2^{-t})$. Due to $|r_{l-1,l}| \le \frac{1}{2}|r_{l-1,l-1}|$ the *fpa*-error of $r_{l-1,l}^2 + r_{l,l}^2 - \delta r_{l-1,l-1}^2$ is at most $O(r_{1,1}(r_{l-1,l-1} + r_{l,l}) d\,7^l \bar{M}_0 \bar{M}_1 2^{-t})$. If $r_{l,l} \le r_{l-1,l-1}$ that *fpa*-error is less than $\varepsilon r_{l-1,l-1}^2$ for $t \ge 5n + 2\log_2 M_0$ due to $\bar{M}_0 \le M_0$, $\bar{M}_1 \le 2^{\frac{n-1}{4}}$. Then a valid swap for δ_- under ideal arithmetic, will also be executed under *fpa* and each swap under *fpa* is a valid swap for δ_+. If $r_{l,l} > r_{l-1,l-1}$ the inequality $\delta_- r_{l-1,l-1}^2 < r_{l-1,l}^2 + r_{l,l}^2$ is preserved under *fpa*-errors. Hence swapping is always correct for δ_-.

Time bound. As $\delta \le 1 - 2\varepsilon$, $\delta_+ \le 1 - \varepsilon$ we have that $\delta \le \delta_+^2$. Hence $\mathbf{LLL}_H$ performs at most $\log_{1/\delta_+} M^n \le 2\,n \log_{1/\delta} M$ LLL-swaps under *fpa*, each swap requiring one $\texttt{TriCol}_l$-execution. We have shown that $\texttt{TriCol}_l$ performs 2 rounds and thus requires $O(nd)$ arithmetic steps and 2 sqrt's. We see that $\mathbf{LLL}_H$ runs in $O(n^2 d \log_{1/\delta} M)$ arithmetic steps.

Costs of the sqrt's. There are $O(n \log_{1/\delta} M)$ sqrt's to be computed with $t = 5n + 2\log_2 M_0$ precision bits, one sqrt per round of $\texttt{TriCol}_l$. Using Newton iteration this requires $O(n \log_{1/\delta} M \log(n + \log_2 M_0))$ arithmetic steps that are covered by the claimed step bound provided that $\log_2 \log_2 M_0 = O(n^2)$.

Newton's iteration $x_0 := 1$, $x_{k+1} := \frac{1}{2}(x_k + \frac{m}{x_k})$ converges quadratically to $\sqrt{m}$. Therefore $O(\log(n + \log_2 M_0))$ rounds of Newton iteration suffice to compute $\sqrt{m}$ for $m \le 2^n M_0$ up to an error less than $2^{-2n}/M_0^2$. □

$\mathbf{LLL}_H$ *in practice.* In practice $\mathbf{LLL}_H$ is correct up to dimension $n = 250$ under *fpa* with $t = 53$ precision bits for *arbitrary* M_0, and not just for $t \ge$

$5n + 2\log_2 M_0$ as shown in Theorem 2. In practice, the constant 7 of Prop. 1 can be replaced by a constant near 1.1 [KS01b]. This is because the orthogonal transforms Q_j preserve the length of error vectors. Moreover the error vector resulting from computing $fl(Q_j \boldsymbol{r})$ is, due to cancellations, on average much smaller than in worst-case. However, $\mathbf{LLL}_H$ is in practice incorrect for $t = 53$ and dimension 400, see [KS01b].

Scaled LLL-reduction. Scaling is a useful concept of numerical analysis for reducing *fpa*-errors. Scaled LLL-reduction of [KS01b] associates with a given lattice basis an associated scaled basis that generates a sublattice of the given lattice. The scaled basis has all values $\bar{M}_0, \bar{M}_1 \leq 2$, which makes the error bounds (3), (4) particularly good. Its coefficients $\mu_{j,i}$ can be correctly computed using only limited *fpa*-precision. Scaled LLL-reduction performs a weak size-reduction, reducing relative to an associated scaled basis. The weaker size-reduction does scarcly lessen the quality of the reduced basis and can be done using limited precision. This way it is possible to implement variants of $\mathbf{LLL}_H$ and **SLLL** that are correct for all practical cases, namely up to dimension 2^{15} using *fpa* with merely 53 precision bits and preserving the run times of this paper.

Comparison with [S88] and the modular LLL of [St96]. The time bound of Theorem 2 also holds for the theoretic, less practical method of [S88].

The modular LLL [St96] performs $O(nd \log_{1/\delta} M)$ arithmetic steps on integers of bit length $\log_2(M_0 M)$ using standard matrix multiplication. This yields the same bound for the number of bit operations for $\mathbf{LLL}_H$ and the modular LLL [St96] if $M_0 = 2^{\Omega(n)}$. If $M_0 = 2^{o(n)}$ the given basis is shorter than an LLL-basis and LLL-reduction is useless. The practicability of $\mathbf{LLL}_H$ rests on the use of small integers of bit length $5n + 2\log_2 M_0$ whereas [St96] uses long integers of bit length $\log_2(M_0 M) = O(n \log M_0)$.

4 Basic Segment LLL-Reduction.

This section introduces main concepts of segment LLL-reduction and a first algorithm $\mathbf{SLLL}_0$. The argument of Theorem 4 for bounding the number of local LLL-reductions within $\mathbf{SLLL}_0$ will be used throughout the paper. This is also true for Lemma 1 and Corollary 1 that bound the norm of, and the *fpa*-errors induced by, local LLL-transforms. The algorithm $\mathbf{SLLL}_0$ is faster by a factor n in the number of arithmetic steps compared to $\mathbf{LLL}_H$ but uses longer integers and *fpa* numbers of bit length $5n + \log_2(M_0^2 M_1^3)$. The algorithm **SLLL** of section 5 reduces this bit length to $7n + 2\log_2 M_0$.

Segments and local coordinates. Let the basis $B = [\boldsymbol{b}_1, \ldots, \boldsymbol{b}_n] \in \mathbb{Z}^{d \times n}$ have dimension $n = k\,m$ and GNF $R \in \mathbb{R}^{n \times n}$. We partition B into m *segments* $B_{l,k} = [\boldsymbol{b}_{lk-k+1}, \ldots, \boldsymbol{b}_{lk}]$ for $l = 1, ..., m$. Local LLL-reduction of two consecutive segments $B_{l,k}, B_{l+1,k}$ is done in local coordinates of the submatrix

$$R_{l,k} := [r_{lk+i,lk+j}]_{-k<i,j\leq k} \in \mathbb{R}^{2k \times 2k}$$

of R. Let $H = [\boldsymbol{h}_1, ..., \boldsymbol{h}_n] = [h_{i,j}] \in \mathbb{R}^{d \times n}$ be the lower triangular matrix of Householder vectors and $H_{l,k} = [h_{lk+i,lk+j}]_{-k<i,j\leq k} \subset H$ the submatrix for

$R_{l,k}$. We control the calls, and minimize the number, of local LLL-reductions of the $R_{l,k}$ by means of the *local squared determinant* of $B_{l,k}$

$$D_{l,k} =_{\text{def}} \|\boldsymbol{q}_{lk-k+1}\|^2 \cdots \|\boldsymbol{q}_{lk}\|^2.$$

We have that $d_{lk} = \|\boldsymbol{q}_1\|^2 \cdots \|\boldsymbol{q}_{lk}\|^2 = D_{1,k} \cdots D_{l,k}$. Moreover, we will use

$$\mathcal{D}^{(k)} =_{\text{def}} \prod_{l=1}^{m-1} d_{lk} = \prod_{l=1}^{m-1} D_{l,k}^{m-l},$$

$$M_{l,k} =_{\text{def}} \max_{lk-k<i\le j\le lk+k} \|\boldsymbol{q}_i\|/\|\boldsymbol{q}_j\|.$$

$M_{l,k}$ is the M_1-value of $R_{l,k}$ of `locLLL`$(R_{l,k})$, obviously $M_{l,k} \le M_1$.

Definition 2. *A basis $\boldsymbol{b}_1, \dots, \boldsymbol{b}_n \in \mathbb{Z}^d$, $n = km$, is an* SLLL$_0$-basis (or SLLL$_0$-reduced) *for given k, $\delta > \frac{1}{4}$, $\alpha = 1/(\delta - 1/4)$ if it is size-reduced and*

1. $\delta \|\boldsymbol{q}_i\|^2 \le \mu_{i+1,i}^2 \|\boldsymbol{q}_i\|^2 + \|\boldsymbol{q}_{i+1}\|^2$ *for* $i \in [1, n-1] \setminus k\mathbb{Z}$,
2. $D_{l,k} \le (\alpha/\delta)^{k^2} D_{l+1,k}$ *for* $l = 1, \dots, m-1$.

Size-reducedness under *fpa* means that $|\mu_{j,i}| < \frac{1}{2} + \varepsilon$ holds for $1 \le i < j \le n$. We neglect the role of ε in SLLL-reduction, ε plays the same role as for $\mathbf{LLL}_H$.

Segment $B_{l,k}$ of an SLLL$_0$-basis is LLL-reduced in the sense that the $k \times k$-submatrix $[r_{lk+i,lk+j}]_{-k<i,j\le 0} \subset R$ is LLL-reduced. Clause 1 does not bridge distinct segments since the $i \in k\mathbb{Z}$ are excepted. Clause 2 relaxes the inequality $D_{l,k} \le \alpha^{k^2} D_{l+1,k}$ of LLL-bases, and this allows to bound the number of local LLL-reductions, see Theorem 4.

We could have used two independent δ-values for the two clauses of Def.2. Theorem 3 shows that the first vector of an SLLL$_0$-basis of lattice $\mathcal{L}$ is almost as short relative to $(\det \mathcal{L})^{1/n}$ as for LLL-bases.

Theorem 3. *Every SLLL$_0$-basis $\boldsymbol{b}_1, \dots, \boldsymbol{b}_n$ satisfies* $\|\boldsymbol{b}_1\| \le (\alpha/\delta)^{\frac{n-1}{4}} (\det \mathcal{L})^{\frac{1}{n}}$.

Proof. Every SLLL$_0$-basis satisfies by clause 2 of Def.2

$$D_{1,k} \le (\alpha/\delta)^{k^2 (i-1)} D_{i,k} \qquad \text{for } i = 1, \dots, m.$$

We multiply the m inequalities and take the m-th root. As $D_{1,k} \cdots D_{m,k} = (\det \mathcal{L})^2$ and $1 + 2 + \cdots + (m-1) = m \cdot \frac{m-1}{2}$ this yields

$$D_{1,k} \le (\alpha/\delta)^{k^2 \frac{m-1}{2}} (\det \mathcal{L})^{\frac{2}{m}}.$$

Moreover $\|\boldsymbol{b}_1\|^2 \le \alpha^{\frac{k-1}{2}} D_{1,k}^{\frac{1}{k}}$ holds as the basis $\boldsymbol{b}_1, \dots, \boldsymbol{b}_k$ is LLL-reduced. Combining the two latter inequalities proves the claim

$$\|\boldsymbol{b}_1\|^2 \le \alpha^{\frac{k-1}{2}} (\alpha/\delta)^{k \frac{m-1}{2}} (\det \mathcal{L})^{\frac{2}{mk}} \le (\alpha/\delta)^{\frac{n-1}{2}} (\det \mathcal{L})^{\frac{2}{n}}. \qquad \square$$

The dual of Theorem 3. Clause 2 of Def.2 is preserved under duality. If it holds for a basis $\boldsymbol{b}_1, \dots, \boldsymbol{b}_n$ it also holds for the dual basis $\boldsymbol{b}_1^*, \dots, \boldsymbol{b}_n^*$ of the lattice $\mathcal{L}^*$. We have that $\|\boldsymbol{b}_1^*\| = \|\boldsymbol{q}_n\|^{-1}$ and $\det(\mathcal{L}^*) = (\det \mathcal{L})^{-1}$. Hence, Theorem 3 implies that every SLLL$_0$-basis satisfies $\|\boldsymbol{q}_n\| \ge (\delta/\alpha)^{\frac{n-1}{4}} (\det \mathcal{L})^{\frac{1}{n}}$.

Local LLL-reduction. The procedure `locLLL`$(R_{l,k})$ locally LLL-reduces $R_{l,k} \subset R$ given $H_{l,k} \subset H$. Initially it produces a copy $[\mathbf{b}'_1, \dots, \mathbf{b}'_{2k}]$ of $R_{l,k}$. It LLL-reduces the local basis $[\mathbf{b}'_1, ..., \mathbf{b}'_{2k}]$ consisting of *fpa*-vectors. It updates and stores the local transform $T_{l,k} \in \mathbb{Z}^{2k\times 2k}$ so that $[\mathbf{b}'_1, ..., \mathbf{b}'_{2k}] = R_{l,k}T_{l,k}$ always holds for the current local basis $[\mathbf{b}'_1, ..., \mathbf{b}'_{2k}]$ and the initial $R_{l,k}$. E.g., it does $\mathrm{col}(l', T_{l,k}) := \mathrm{col}(l', T_{l,k}) - \mu\, \mathrm{col}(i, T_{l,k})$ along with $\boldsymbol{b}'_{l'} := \boldsymbol{b}'_{l'} - \mu\, \boldsymbol{b}'_i$ within $\mathtt{TriCol}_l$. It freshly computes $\boldsymbol{b}'_{l'}$ from the updated $T_{l,k}$. Using a correct $T_{l,k}$ this correction of $\boldsymbol{b}'_{l'}$ limits *fpa*-errors of the local basis, see Cor.1.

Local LLL-reduction of $R_{l,k}$ is done in local coordinates of dimension $2k$. A local LLL-swap merely requires $O(k^2)$ arithmetic steps, update of $R_{l,k}$, local triangulation and size-reduction via $\mathtt{TriCol}_l$ included, compared to $O(nd)$ arithmetic steps for an LLL-swap in global coordinates.

$\mathtt{locLLL}(R_{l,k})$

1. produce copies $[\boldsymbol{b}'_1, ..., \boldsymbol{b}'_{2k}] = R'_{l,k}$ of $R_{l,k}$ and $[\boldsymbol{h}'_1, ..., \boldsymbol{h}'_{2k}]$ of $H_{l,k} \subset H$
   $T_{l,k} := I_{2k},\ \ l' := 1$
2. `WHILE` $l' \le 2k$ `DO`
   $\mathtt{TriCol}(\boldsymbol{b}'_1, ..., \boldsymbol{b}'_{l'}, \boldsymbol{h}'_1, ..., \boldsymbol{h}'_{l'-1}, \boldsymbol{r}'_1, ..., \boldsymbol{r}'_{l'-1})$
   update $T_{l,k}$, $\boldsymbol{b}'_{l'} := R_{l,k}\, \mathrm{col}(l', T_{l,k})$
   `IF` $l' \ne 1$ `and` $\delta\, r'^2_{l'-1,l'-1} > r'^2_{l'-1,l'} + r'^2_{l',l'}$
   `THEN` swap $\boldsymbol{b}'_{l'-1}, \boldsymbol{b}'_{l'}$, swap $\boldsymbol{r}'_{l'-1}, \boldsymbol{r}'_{l'}$, update $T_{l,k}$, $l' := l' - 1$
   `ELSE` $l' := l' + 1$.

SLLL$_0$*-algorithm.* **SLLL**$_0$ transforms a given basis into an SLLL$_0$-basis. It iterates $\mathtt{locLLL}(R_{l,k})$ for submatrices $R_{l,k} \subset R$, followed by a global update that *transports* $T_{l,k}$ to B and triangulates $B_{l,k}, B_{l,k+1}$ via $\mathtt{TriSeg}_{l,k}$. *Transporting* $T_{l,k}$ to $B, R, T_{1,n/2}$ and so on means to multiply the submatrix consisting of $2k$ columns of $B, R, T_{1,n/2}$ corresponding to $R_{l,k}$ from the right by $T_{l,k}$.

The procedure $\mathtt{TriSeg}_{l,k}$ *triangulates* and size-reduces two adjacent segments $B_{l,k}, B_{l+1,k}$. Given $B_{l,k}, B_{l+1,k}$ and $\boldsymbol{h}_1, ..., \boldsymbol{h}_{lk-k}$, it computes $[\boldsymbol{r}_{lk-k+1}, ..., \boldsymbol{r}_{lk+k}] \subset R$ and $[\boldsymbol{h}_{lk-k+1}, ..., \boldsymbol{h}_{lk+k}] \subset H$.

$\mathtt{TriSeg}_{l,k}$

1. `FOR` $l' = lk - k + 1, ..., lk + k$ `DO` $\mathtt{TriCol}_{l'}$ (including updates of $T_{l,k}$)
2. $D_{j,k} := \prod_{i=0}^{k-1} r^2_{kj-i,kj-i}$ for $j = l, l+1$.

SLLL$_0$

`INPUT` $\boldsymbol{b}_1, \ldots, \boldsymbol{b}_n \in \mathbb{Z}^d$ (a basis with M_0, M_1, M), k, m, δ

`OUTPUT` $\boldsymbol{b}_1, \ldots, \boldsymbol{b}_n$ SLLL$_0$-basis for k, δ

`WHILE` $\exists\, l, 1 \le l < m$ such that `either` $D_{l,k} > (\alpha/\delta)^{k^2} D_{l+1,k}$
   `or` $\mathtt{TriSeg}_{l,k}$ has not yet been executed
 `DO` for the minimal such l: $\mathtt{TriSeg}_{l,k}$, $\mathtt{locLLL}(R_{l,k})$
   # *global update*: $[B_{l,k}, B_{l+1,k}] := [B_{l,k}, B_{l+1,k}]\, T_{l,k}$, $\mathtt{TriSeg}_{l,k}$.

Correctness in ideal arithmetic. All inequalities $D_{l,k} \le (\alpha/\delta)^{k^2} D_{l+1,k}$ hold upon termination of $\mathbf{SLLL}_0$. All segments $B_{l,k}$ are locally LLL-reduced and globally size-reduced and thus the terminal basis is SLLL_0-reduced.

The number of `locLLL`*-executions.* Let $\#_k$ denote the number of $\texttt{loclll}(R_{l,k})$-executions due to $D_{l,k} > (\alpha/\delta)^{k^2} D_{l,k}$ for all l. The first $\texttt{loclll}(R_{l,k})$-executions for each l is possibly not counted in $\#_k$, this yields at most $n/k - 1$ additional executions. We bound $\#_k$ by the Lovász volume argument.

Theorem 4. $\#_k \le 2\, n\, k^{-3} \log_{1/\delta} M$.

Proof. We show that a $\texttt{locLLL}(R_{l,k})$-execution decreases $D_{l,k}$ by the factor $\delta^{k^2/2}$ if it is due to $D_{l,k} > (\alpha/\delta)^{k^2} D_{l+1,k}$. $\texttt{locLLL}(R_{l,k})$ changes $D_{l,k}$, $D_{l+1,k}$ into $D'_{l,k}$, $D'_{l+1,k}$ and preserves $D_{l',k}$ for $l' \ne l, l+1$. It also preserves the product $D_{l,k} D_{l+1,k}$. $\texttt{locLLL}(R_{l,k})$ results in $D'_{l,k} \le \alpha^{k^2} D'_{l+1,k}$ because upon termination the matrix $R_{l,k}$ is LLL-reduced with δ and thus the claim follows from $\|\boldsymbol{q}_{lk-2k+i}\|^2 \le \alpha^k \, \|\boldsymbol{q}_{lk-k+i}\|^2$ for $i = 1, ..., k$. Therefore

$$D'_{l,k} \le \alpha^{k^2} D'_{l+1,k} = \alpha^{k^2} D'_{l,k} D'_{l+1,k} / D'_{l,k}$$
$$= \alpha^{k^2} D_{l,k} D_{l+1,k} / D'_{l,k} < \delta^{k^2} D^2_{l,k} / D'_{l,k},$$

and thus $D'_{l,k} \le \delta^{k^2/2} D_{l,k}$. Hence $\texttt{locLLL}(R_{l,k})$ decreases

$$\mathcal{D}^{(k)} = \prod_{l=1}^{m-1} d_{lk} = \prod_{l=1}^{m-1} D^{m-l}_{l,k}$$

by the factor $\delta^{k^2/2}$. As $\mathcal{D}^{(k)}$ is a positive integer, $\mathcal{D}^{(k)} \le M^{m-1}$, this implies

$$\#_k \ \le \ \log_{1/\delta^{k^2/2}} M^{m-1} \ \le \ 2 \tfrac{m-1}{k^2} \log_{1/\delta} M. \qquad \square$$

All intermediate $M_{l,k}$-values within $\mathbf{SLLL}_0$ are bounded by the M_1-value of the input basis of $\mathbf{SLLL}_0$. Consider the local transform $T_{l,k} \in \mathbb{Z}^{2k \times 2k}$ within $\texttt{locLLL}(R_{l,k})$. Let $\|T_{l,k}\|_1$ denote the maximal $\| \; \|_1$-norm of the columns of $T_{l,k}$.

Lemma 1. *Within* $\texttt{locLLL}(R_{l,k})$ *we have that* $\|T_{l,k}\|_1 \le 6k(\frac{3}{2})^{2k} M_{l,k}$.

Proof. We rename the input basis $\boldsymbol{b}'_1, ..., \boldsymbol{b}'_{2k}$ of $\texttt{locLLL}(R_{l,k})$ into $\boldsymbol{b}_1, ..., \boldsymbol{b}_{2k}$ and we let $\boldsymbol{b}'_1, ..., \boldsymbol{b}'_{2k}$ denote the current local basis. The input basis has been size-reduced by the preceding $\texttt{TriSeg}_{l,k}$-execution, and thus $|\mu_{j,i}| \le \frac{1}{2}$ for $1 \le i < j \le 2k$. W.l.o.g. let $|\mu'_{l,i}| \le \frac{1}{2}$ for $1 \le i < l \le 2k$ hold for the current basis because $\|\mathrm{col}(l', T_{l',k})\|_1$ increases during size-reduction of $\boldsymbol{b}'_{l'}$. The equations

$$[\boldsymbol{b}'_1, ..., \boldsymbol{b}'_{2k}] = [\boldsymbol{q}'_1, ..., \boldsymbol{q}'_{2k}][\mu'_{j,i}]^t = [\boldsymbol{q}_1, ..., \boldsymbol{q}_{2k}][\mu_{j,i}]^t T_{l,k}.$$

yield $T_{l,k} = ([\mu_{j,i}]^t)^{-1} \; [\langle \boldsymbol{q}_j, \boldsymbol{q}'_i \rangle \|\boldsymbol{q}_j\|^{-2}] \; [\mu'_{j,i}]^t_{1 \le i,j \le 2k}$. The coefficients $\nu_{j,i}$ of the inverse matrix $[\nu_{j,i}] := ([\mu_{j,i}]^t)^{-1}$ satisfy $|\nu_{j,i}| \le (\frac{3}{2})^{|j-i|}$, and thus $\sum_{i=1}^{2k} |\nu_{j,i}| \le \sum_{i=1}^{2k} (\frac{3}{2})^{|j-i|} < 3(\frac{3}{2})^{2k}$. We get that

$$\|T_{l,k}\|_1 \le 6k(\tfrac{3}{2})^{2k} \max_{1 \le i,j \le 2k} |\langle \boldsymbol{q}_j, \boldsymbol{q}'_i \rangle| / \|\boldsymbol{q}_j\|^2.$$

To finish the proof we show that $\max_{1\leq i,j\leq 2k} |\langle \boldsymbol{q}_j, \boldsymbol{q}'_i\rangle|/\|\boldsymbol{q}_j\|^2 \leq M_{l,k}$.
If $\boldsymbol{b}_{l'-1}, \boldsymbol{b}_{l'}$ get swapped, the swapped vectors $\boldsymbol{b}'_{l'-1}, \boldsymbol{b}'_{l'}$ clearly satisfy

$$\|\boldsymbol{q}_{l'}\| \leq \|\boldsymbol{q}'_{l'-1}\|, \ \|\boldsymbol{q}'_{l'}\| \leq \|\boldsymbol{q}_{l'-1}\|,$$

and thus $|\langle \boldsymbol{q}_j, \boldsymbol{q}'_i\rangle|/\|\boldsymbol{q}_j\|^2 \leq \|\boldsymbol{q}'_i\|/\|\boldsymbol{q}_j\| \leq \|\boldsymbol{q}_{l'-1}\|/\|\boldsymbol{q}_{l'}\|$ holds for $l'-1 \leq i,j \leq l'$, i.e., for the i,j that are *linked* by the LLL-swap.

More generally, we say that i,j are *linked* by a sequence of LLL-swaps, swapping $\boldsymbol{b}_{h_\nu}, \boldsymbol{b}_{h_\nu+1}$ for $\nu = 1, ..., s$ if the edges $(h_\nu, h_{\nu+1})$ link i and j by an undirected path. By induction on the sequence of LLL-swaps we see that $\|\boldsymbol{q}'_i\|/\|\boldsymbol{q}_j\| \leq M_{l,k}$ holds for all i,j such that the terminal $\boldsymbol{b}'_i$ and the initial $\boldsymbol{b}_j$ are linked by a sequence of LLL-swaps. Otherwise, if $\boldsymbol{b}'_i$ and $\boldsymbol{b}_j$ are not linked, we have that $\langle \boldsymbol{q}_j, \boldsymbol{q}'_i\rangle/\|q_j\|^2 = \delta_{i,j}$ because $\boldsymbol{q}'_i$ is in the linear space generated by the $\boldsymbol{q}_j$ such that $\boldsymbol{b}'_i$ and $\boldsymbol{b}_j$ are linked, and thus $\langle \boldsymbol{q}_j, \boldsymbol{q}'_i\rangle = 0$ for $i \neq j$. In particular, the quotients $\|\boldsymbol{q}_i\|/\|\boldsymbol{q}_j\|$ for $i > j$, which are not bounded by $M_{l,k}$, are irrelevant, they do not induce LLL-swaps and do not affect $T_{l,k}$. □

Next we study $\texttt{locLLL}(R_{l,k})$ under *fpa*, where $\texttt{TriCol}_l$ performs the iterative *fpa*-version of $\texttt{TriCol}_l$ that depends on ε, $0 < \varepsilon < 0.2$.

Corollary 1. *1. Within* $\texttt{locLLL}(R_{l,k})$ *the current* $R'_{l,k} := R_{l,k}T_{l,k}$ *and its approximation* $\bar{R}'_{l,k}$ *satisfy* $\|\bar{R}'_{l,k} - R'_{l,k}\|_F \leq \|\bar{R}_{l,k} - R_{l,k}\|_F 2^{2k} M_{l,k} + 7n\|R_{l,k}\|_F 2^{-t}$.
2. Let $\texttt{TriSeg}_{l,k}$ *and* $\texttt{locLLL}$ *use* fpa *with* $t = 3n + \log_2(M_0^2 M_1^3) + 2k$ *precision bits. If* $\bar{R}_{l,k}$ *is computed by* $\texttt{TriSeg}_{l,k}$ *then* $\texttt{locLLL}(\bar{R}_{l,k})$ *computes for* $n \geq n_0$ *a correct* $T_{l,k}$ *so that* $R_{l,k}T_{l,k}$ *is LLL-reduced with* $\delta\_$.

Proof. 1. $\texttt{locLLL}(R_{l,k})$ updates the current $R'_{l,k} = [\boldsymbol{b}'_1, ..., \boldsymbol{b}'_{2k}]$ by transforming the initial $R_{l,k}$ into $R'_{l,k} := R_{l,k}T_{l,k}$. In ideal arithmetic this increases $\|\bar{R}_{l,k} - R_{l,k}\|_F$ by at most a factor $\|T_{l,k}\|_1\sqrt{2k} \leq 2^{2k} M_{l,k}$ holds for $k \geq 9$ by Lemma 1. The $7n\|R_{l,k}\|_F 2^{-t}$-term accounts for the *fpa*-errors of the calculation of $R_{l,k}T_{l,k}$, using e.g., (15.30)[LH95] for $d \geq 37$. This term can be neglected as it is covered by the upper bound of $\|\bar{R}_{l,k} - R_{l,k}\|_F$ that follows from (1).

2. The input $R_{l,k}$ of $\texttt{locLLL}(R_{l,k})$ satisfies the inequalities (3),(4) with $\bar{M}_1 \leq M_1$. Therefore $\texttt{TriSeg}_{l,k}$'s *fpa*-errors are by a factor $M_1^2/2^{\frac{n-1}{2}}$ larger than for $\texttt{TriCol}_l$-executions within $\mathbf{LLL}_H$, where the input $\boldsymbol{b}_1, ..., \boldsymbol{b}_{l-1}$ is LLL-reduced and $\bar{M}_1 \leq 2^{\frac{n-1}{4}}$. This is offset by $2\log_2 M_1 - n/2$ additional precision bits.

We compensate the loss of precision described by clause 1 by another $\log_2 M_1 + 2k$ additional precision bits. Thus we add to the precision t of Theorem 2 $\log_2(M_1^3) - \frac{n}{2} + 2k$ with $k \leq \frac{n}{4}$ to get $t = 5n + \log_2(M_0^2 M_1^3)$. With the increased precision the argument of Theorem 2 shows the correctness of $T_{l,k}$. □

Theorem 5. *Let* $k = \Theta(\sqrt{n})$*. Given a basis with* M_0, M_1, M*,* $\mathbf{SLLL}_0$ *computes under* fpa *with* $t = 5n + \log_2(M_0^2 M_1^3)$ *precision bits for* $n \geq n_0$ *an* $SLLL_0$*-basis for* $\delta\_$*. It runs in* $O(nd \log_{1/\delta} M)$ *arithmetic steps using* $5n + \log_2(M_0^2 M_1^3)$*-bit integers and* fpa *numbers.*

$\mathbf{SLLL}_0$ saves a factor n in the number of arithmetic steps compared to $\mathbf{LLL}_H$ but uses longer integers and *fpa* numbers. The choice $k, m = \Theta(\sqrt{n})$ equalizes for $d = O(n)$ the number of local and global arithmetic steps. $\mathbf{SLLL}_0$ runs for $M_0 = 2^{O(n)}$, and thus for $M = 2^{O(n^2)}$, in $O(n^3 d)$ arithmetic steps using $O(n^2)$ bit integers. The bit length $O(n^2)$ will be reduced to $O(n)$ by the algorithm **SLLL** see Theorem 7.

Proof. Time bound. We separately count the *local* (resp. *global*) arithmetic steps of $\texttt{locLLL}(R_{l,k})$ (resp., of $\texttt{TriSeg}_{l,k}$). Initially we have that $\mathcal{D}^{(1)} \leq M^n$. Each LLL-swap of $\boldsymbol{b}_{l-1}, \boldsymbol{b}_l$, due to the inequality $\delta r^2_{l-1,l-1} > r^2_{l-1,l} + r^2_{l,l}$, decreases $\mathcal{D}^{(1)}$ by a factor δ. As initially $\mathcal{D}^{(1)} \leq M^n$ and $\mathcal{D}^{(1)} \geq 1$ holds upon termination there are at most $n \log_{1/\delta} M$ LLL-swaps.

Each of the $n \log_{1/\delta} M$ LLL-swaps, done in local coordinates of dimension $2k$, requires $O(k^2)$ steps for a local $\texttt{TriCol}_l$-execution and for updating $T_{l,k}$. In total there are $O(nk^2 \log_{1/\delta} M)$ local arithmetic steps.

Each $\texttt{locLLL}(R_{l,k})$-execution requires $O(ndk)$ global arithmetic steps for $\texttt{TriSeg}_{l,k}$ and for updating $B_{l,k}, B_{l+1,k}$. Therefore, the $n/k + 2nk^{-3} \log_{1/\delta} M$ $\texttt{locLLL}(R_{l,k})$-executions require $O(n^2 d + m^2 d \log_{1/\delta} M)$ global arithmetic steps. This proves the claimed step bound using that $M \geq 2^n$ and $m^2 = \Theta(n)$.

Correctness under fpa. We see from Cor.1(2) that $\texttt{locLLL}(R_{l,k})$ correctly LLL-reduces $R_{l,k}$ with $\delta\_$, computing a correct $T_{l,k}$ for $n \geq n_0$, $n \geq 4k$. The *fpa*-errors within $\texttt{locLLL}(R_{l,k})$ get corrected by the subsequent global update "$[B_{l,k}, B_{l+1,k}] := [B_{l,k}, B_{l+1,k}]\, T_{l,k}$, $\texttt{TriSeg}_{l,k}$" which restores and even improves the initial error bounds.

Selecting the right $R_{l,k}$ for the next $\texttt{locLLL}(R_{l,k})$-call within $\mathbf{SLLL}_0$ rests on the decision whether $D_{l,k}, D_{l+1,k}$ differ by at least a factor $(\alpha/\delta)^{k^2}$, where $(\alpha/\delta)^{k^2} > (\frac{4}{3})^{k^2} > 2^{0.4n}$ for $k \geq \sqrt{n}$. This is always correctly decided because the $r_{i,i}$ and thus $D_{l,k}, D_{l+1,k}$ are computed with an arbitrary small relative error ε due inequality (1). W.l.o.g. we can assume that all except possibly one $r_{i,i}$ satisfy $r_{i,i} \geq 2^{-n}/M_0$.

Intermediate basis vectors have length $\leq 6k(\frac{3}{2})^{2k} M_0 M_1 2^n = 2^{n+o(n)} M_0 M_1$ because $\|T_{l,k}\|_1 \leq 6k(\frac{3}{2})^{2k} M_1$ holds by Lemma 1, and size-reduction increases the length of intermediate basis vectors by at most a factor 2^n. Hence all integers and *fpa* numbers within $\mathbf{SLLL}_0$ have bit length $5n + \log_2(M_0^2 M_1^3)$. □

5 Gradual SLLL Reduction Using Short Bases.

The algorithm **SLLL** of this section achieves the same length defect as **LLL**, uses intermediate bases of length $2^{n+o(n)} M_0$, and is correct under *fpa* with $t = 7n + 2\log_2 M_0$ precision bits. **SLLL** prepares local LLL-reductions through local reductions on subsegments that get reduced with smaller δ-values, all local transforms have norm $2^{n+o(n)}$. **SLLL** saves a factor $n/\log_2 n$ in the number of arithmetic steps compared to $\mathbf{LLL}_H$, using $7n + 2\log_2 M_0$-bit integers and *fpa* numbers. For input bases of length $2^{O(n)}$ and $d = O(n)$ **SLLL** performs $O(n^{5+\varepsilon})$ bit operations for every $\varepsilon > 0$ compared to $O(n^{6+\varepsilon})$ bit operations for $\mathbf{LLL}_H$,

$\mathbf{SLLL}_0$ and the LLL-algorithms of [S88], [St96]. The advantage of **SLLL** is the use of small integers of bit length $7n + 2\log M_0$ which is crucial in practice.

The use of small integers and short intermediate bases within **SLLL** rests on a gradual LLL-type reduction so that all local LLL-transforms $T_{l,2^\sigma}$ of $R_{l,2^\sigma}$ have norm $O(2^n)$. This requires to work with segments of all sizes 2^σ and to perform LLL-reduction on $R_{l,2^\sigma}$ with a measured strength, i.e., SLLL-reduction according to Definition 3. If the submatrices $R_{2l,2^{\sigma-1}}, R_{2l+1,2^{\sigma-1}} \subset R_{l,2^\sigma}$ are already SLLL-reduced then `locLLL`$(R_{l,k})$ performs a transform $T_{l,2^\sigma}$ bounded as $\|T_{l,2^\sigma}\|_F = O(2^n)$. This is the core of *fpa*-correctness of **SLLL**.

Comparison with semi-reduction of [Sc84, St96]. The semi-reduction algorithm of [Sc84] also uses segments but proceeds without adjusting LLL-reduction according to Def. 2 and without Theorem 4. This algorithm runs for input bases of length $2^{O(n)}$ in $O(n^{6+\varepsilon})$ bit operations, its combination with modular reduction [St96] runs in $O(n^{5.5+\varepsilon})$-bit operations. This time bound also holds for a combination of [S88] and [Sc84], see Theorem 9 [S88]. Assuming that $n \times n$ matrices can be multiplied using $O(n^\beta)$ arithmetic steps the semi-reduction of [St96, Thm 24] runs in $O(n^{5+\frac{1}{5-\beta}+\varepsilon})$ bit operations. **SLLL** beats the [St96] time bound even if $n \times n$-matrix multiplication can be done in $O(n^2)$ steps. **SLLL** achieves for every $\varepsilon > 0$ length defect $(\frac{4}{3} + \varepsilon)^{n/2}$ whereas semi-reduction achieves 2^n. Moreover, **SLLL** is practical even for small n since all our O-constants and n_0-values are small.

We let n be a power of 2, $\frac{1}{2} \le \delta < 1$, $\alpha = \frac{1}{\delta - \frac{1}{4}}$. We set $s := \lceil \frac{1}{2}\log_2 n \rceil$ so that $\sqrt{n} \le 2^s < 2\sqrt{n}$.

Definition 3. *A basis* $\boldsymbol{b}_1, ..., \boldsymbol{b}_n \in \mathbb{R}^d$ *is an* SLLL-basis (or SLLL-reduced) *for* $\delta \ge \frac{1}{2}$ *if it satisfies for* $\sigma = 0, ..., s = \lceil \frac{1}{2}\log_2 n \rceil$ *and all* l, $1 \le l < n/2^\sigma$:

$$D_{l,2^\sigma} \le \alpha^{4^\sigma} \delta^{-n} D_{l+1,2^\sigma}.$$

If the inequalities of Def.3 hold for a basis they also hold for the dual basis. Thus the dual of an SLLL-basis is again an SLLL-basis. To preserve SLLL-reducedness by duality we do not require SLLL-bases to be size-reduced.

The inequalities of Def.3 for $\sigma = 0$ mean that $\|\boldsymbol{q}_l\|^2 \le \alpha\delta^{-n}\|\boldsymbol{q}_{l+1}\|^2$ holds for all l. The inequalities of Def.3 are merely required for $2^\sigma \le 2\sqrt{n}$. Therefore, **SLLL** locally LLL-reduces $R_{l,2^\sigma}$ via `locLLL`$(R_{l,2^\sigma})$ merely for segment sizes $2^\sigma < 2\sqrt{n}$, where size-reduction of a vector requires $O(2^{2\sigma}) = O(n)$ arithmetic steps.

The inequalities of Def.3 and $D_{l,k} \le (\alpha/\delta)^{k^2} D_{l+1,k}$ of Def.2 coincide for $k = 2^\sigma$ when setting $\delta := \delta_\sigma$ in Def.2, and $\delta_\sigma := \delta^{n4^{-\sigma}}$ for the δ of Def.3. Note that δ_σ can be arbitrarily small, e.g. $\delta_\sigma \ll \frac{1}{4}$, δ_σ decreases with σ. In particular for $2^\sigma = k \ge \sqrt{n}$ we have that $\alpha^{4^\sigma}\delta^{-n} \le (\alpha/\delta)^{k^2}$ and thus the inequalities of Def.3 are stronger than the ones of Def.2. Next we show via Lemma 2 that the vectors of SLLL-bases approximate the successive minima in nearly the same way as for LLL-bases.

Theorem 6. *Every size-reduced SLLL-basis satisfies*

1. $\lambda_j^2 \leq \alpha^{j-1}\delta^{-7n}\|\boldsymbol{q}_j\|^2$ *for* $j = 1, \ldots, n$,
2. $\|\boldsymbol{b}_l\|^2 \leq \alpha^{j-1}\delta^{-7n}\|\boldsymbol{q}_j\|^2$ *for* $l \leq j$,
3. $\|\boldsymbol{b}_j\|^2 \leq \alpha^{n-1}\delta^{-7n}\lambda_j^2$ *for* $j = 1, \ldots, n$.

Proof. We first prove 1. and 2. There clearly exists l, $1 \leq l \leq j$ so that $\lambda_j \leq \|\boldsymbol{b}_l\|$. Using Lemma 2 and size-reducedness we get

$$\lambda_j^2 \leq \|\boldsymbol{b}_l\|^2 \leq \|\boldsymbol{q}_l\|^2 + \tfrac{1}{4}\sum_{i=1}^{l-1}\|\boldsymbol{q}_i\|^2$$
$$\leq \|\boldsymbol{q}_j\|^2\alpha^{j-1}\delta^{-7n}[\alpha^{1-l} + \tfrac{1}{4}\sum_{i=1}^{l-1}\alpha^{1-i}].$$

This upper bound on $\|\boldsymbol{b}_l\|^2$ holds for all l and j with $l \leq j$. To finish the proof of 1. and 2. it remains to show that $\alpha^{1-l} + \frac{1}{4}\sum_{i=1}^{l-1}\alpha^{1-i} \leq 1$. This is trivial for $l = 1$ and holds for $l \geq 2$ as $\alpha \geq 4/3$ and $\sum_{i=1}^{l-1}\alpha^{1-i} \leq \frac{1-\alpha^{1-l}}{1-3/4}$.
3. We note that every lattice basis satisfies $\lambda_j \geq \|\boldsymbol{b}_l\| \geq \|\boldsymbol{q}_l\|$ for some $l \geq j$, and thus $\lambda_j^2 \geq \|\boldsymbol{q}_l\|^2 \geq \alpha^{-l+i}\delta^{7n}\|\boldsymbol{q}_i\|^2$ holds for all $i \leq l$ by Lemma 2. Hence

$$\|\boldsymbol{b}_j\|^2 \leq \|\boldsymbol{q}_j\|^2 + \tfrac{1}{4}\sum_{i=1}^{j-1}\|\boldsymbol{q}_i\|^2 \leq \delta^{-7n}[\alpha^{l-j} + \tfrac{1}{4}\sum_{i=1}^{j-1}\alpha^{l-i}]\lambda_j^2 \leq \delta^{-7n}\alpha^{l-1}\lambda_j^2$$

holds since $\boldsymbol{b}_j$ is size-reduced, and $\|\boldsymbol{q}_i\|^2 \leq \delta^{-7n}\alpha^{l-i}\lambda_j^2$. □

Bounds for other bases. 1. The proof of Theorem 6 shows that LLL-bases satisfy the inequalities of Theorem 6 with δ^{-7n} replaced by 1, because they satisfy the inequalities of Lemma 2 with δ^{-7n} replaced by 1. Therefore LLL-bases satisfy for $j = 1, ..., n$: $\alpha^{1-j} \leq \|\boldsymbol{q}_j\|^2/\lambda_j^2 \leq \|\boldsymbol{b}_j\|^2/\lambda_j^2 \leq \alpha^{n-1}$.
2. Every size-reduced basis satisfies the inequalities of Lemma 2 with $\alpha^{j-i}\delta^{-7n}$ replaced by M_1^2, i.e., $\|\boldsymbol{q}_i\|^2 \leq M_1^2\|\boldsymbol{q}_j\|^2$ for $i < j$. Retracing the proof of Theorem 6 shows that every size-reduced basis satisfies for $j = 1, ..., n$

$$\tfrac{4}{j+3}/M_1^2 \leq \|\boldsymbol{q}_j\|^2/\lambda_j^2 \leq \|\boldsymbol{b}_j\|^2/\lambda_j^2 \leq \tfrac{j+3}{4}M_1^2.$$

Lemma 2. *Every SLLL-basis* $\boldsymbol{b}_1, \ldots, \boldsymbol{b}_n$ *satisfies*

$$\|\boldsymbol{q}_i\|^2 \leq \alpha^{j-i}\delta^{-7n}\|\boldsymbol{q}_j\|^2 \quad \textit{for } 1 \leq i < j \leq n.$$

Proof. Every SLLL-basis satisfies

$$D_{l,2^\sigma}^{2^{-\sigma}} \leq (\alpha/\delta_\sigma)^{2^\sigma} D_{l+1,2^\sigma}^{2^{-\sigma}} \tag{5}$$

for $\delta_\sigma := \delta^{n\,4^{-\sigma}}$ and $\sigma = 0, ..., s$ and all l, because $(\alpha/\delta_\sigma)^{4^\sigma} = \alpha^{4^\sigma}\delta^{-n}$.

Moreover, we have for *all* l and $\sigma = 0, ..., s$:

$$D_{l,2^\sigma}^{2^{-\sigma}} \leq (\alpha/\delta_\sigma)^{2^{\sigma-1}} D_{\frac{l+1}{2},2^{\sigma+1}}^{2^{-\sigma-1}}. \tag{6}$$

This follows by multiplying both sides of (5) by $D_{l,2^\sigma}^{2^{-\sigma}}$, using the equality $D_{l,2^\sigma}D_{l+1,2^\sigma} = D_{\frac{l+1}{2},2^{\sigma+1}}$ and taking square roots on both sides.

Let $i_0, ..., i_{s-1} \in \{0, 1\}$ and $l_0, ..., l_s \in \mathbb{N}$ satisfy

$$i + \sum_{\sigma'=0}^{\sigma-1}(1 + i_{\sigma'})2^{\sigma'} = l_\sigma\, 2^\sigma \qquad \text{for } \sigma = 0, \ldots, s. \tag{7}$$

We prove for $\sigma = 0, ..., s$ by induction on σ:

$$\|\boldsymbol{q}_i\|^2 \leq \prod_{\sigma'=0}^{\sigma-1}(\alpha/\delta_{\sigma'})^{2^{\sigma'}(\frac{1}{2}+i_{\sigma'})}\, D_{l_\sigma,2^\sigma}^{2^{-\sigma}}. \tag{8}$$

The claim for $\sigma = 0$: $\|\boldsymbol{q}_i\|^2 \le D_{l_0,1} = \|\boldsymbol{q}_i\|^2$ holds as $\sum_{\sigma'=0}^{-1} := 0$, $\prod_{\sigma'=0}^{-1} := 1$, $i = l_0$.

Induction from σ to $\sigma+1$. We see from (7) that $2l_{\sigma+1} = l_\sigma + 1 + i_\sigma$. If l_σ is odd than $i_\sigma = 0$ and $l_{\sigma+1} = \frac{l_\sigma+1}{2}$. In this case we combine (8) with inequality (6) for $l := l_\sigma$. This yields (8) for $\sigma+1$. If l_σ is even, $i_\sigma = 1$ then we first combine (8) with (5) for $l := l_\sigma$, and we proceed with $l_\sigma + 1$ as in the previous case with l_σ.

Applying the inequalities (6) to the dual basis $\boldsymbol{b}_1^*, ..., \boldsymbol{b}_n^*$ we get for odd l and $\sigma = 0, ..., s$:

$$D^{2^{-\sigma-1}}_{\frac{l+1}{2},2^{\sigma+1}} \le (\alpha/\delta_\sigma)^{2^{\sigma-1}} D^{2^{-\sigma}}_{l,2^\sigma}. \tag{6*}$$

Let $j_0, ..., j_{s-1} \in \{0,1\}$ and $l_0^*, ..., l_s^* \in \mathbb{N}$ satisfy

$$j - \textstyle\sum_{\sigma'=0}^{\sigma-1} j_{\sigma'}\, 2^{\sigma'} = l_\sigma^*\, 2^\sigma \qquad \text{for } \sigma = 0, \dots, s. \tag{7*}$$

By duality (8) yields for $\sigma = 1, ..., s$:

$$D^{2^{-\sigma}}_{l_\sigma^*,2^\sigma} \le \textstyle\prod_{\sigma'=0}^{\sigma-1} (\alpha/\delta_{\sigma'})^{2^{\sigma'}(\frac{1}{2}+j_{\sigma'})}\ \|\boldsymbol{q}_j\|^2. \tag{8*}$$

The claim of Lemma 2 clearly holds for $j - i \le 7$ since Def.3 for $\sigma = 0$ requires that $\|\boldsymbol{q}_l\|^2 \le \alpha\delta^{-n}\|\boldsymbol{q}_{l+1}\|^2$. To prove the claim for $j - i \ge 8$ we combine the inequalities (8) and (8*) for a suitable σ. If $j - i \ge 2^{s+2}$ we set $\sigma := s$, otherwise we choose σ such that $2^{\sigma+1} \le j - i < 2^{\sigma+2}$, and thus $\sigma \ge 2$. We set $l_\sigma := \lceil (i-1)/2^\sigma + 1 \rceil$, $l_\sigma^* = \lfloor j/2^\sigma \rfloor$. Then there exist $i_{\sigma'}, j_{\sigma'} \in \{0,1\}$ such that (7), (7*) hold for σ.

Obviously $l_\sigma^* - l_\sigma > (j-i)/2^\sigma - 3 \ge 2 - 3 = -1$ holds for $2 \le \sigma \le s$ because $(j-i)/2^\sigma \ge 2$ for $\sigma < s$ and $(j-i)/2^s \ge 4$ for $\sigma = s$. Hence $l_\sigma \le l_\sigma^*$.

Case $l_\sigma = l_\sigma^*$. By (8) and (8*) : $\|\boldsymbol{q}_i\|^2 \le \prod_{\sigma'=0}^{\sigma-1} (\alpha/\delta_{\sigma'})^{2^{\sigma'}(1+i_{\sigma'}+j_{\sigma'})} \|\boldsymbol{q}_j\|^2$, where $i + \sum_{\sigma'=0}^{\sigma-1}(1 + i_{\sigma'} + j_{\sigma'})2^{\sigma'} = j$. We see from $\delta_{\sigma'} = \delta^{n\,4^{-\sigma'}}$ and $\sum_{\sigma'=0}^{\sigma-1}(1 + i_{\sigma'} + j_{\sigma'})2^{-\sigma'} \le 6 - 2^{-\sigma+1}$ that

$$\|\boldsymbol{q}_i\|^2 \le \alpha^{j-i}\delta^{-6n+n2^{-\sigma+1}} \|\boldsymbol{q}_j\|^2. \tag{9}$$

Case $l_\sigma < l_\sigma^*$. We set $l' := l_\sigma^* - l_\sigma$. We combine (8), (8*) and $D^{2^{-\sigma}}_{l_\sigma,2^\sigma} \le (\alpha/\delta_\sigma)^{2^\sigma l'} D^{2^{-\sigma}}_{l_\sigma+l',2^\sigma}$ which follows from (5). This induces into the right side of (9) another factor $\delta_\sigma^{-2^\sigma l'}$.

For $\sigma = s$ we have $\delta_s^{-2^s l'} = \delta^{-n2^{-s}l'} \le \delta^{-n}$ as $l' < (j-i)2^{-s} < n2^{-s} \le 2^s$. Hence $\|\boldsymbol{q}_i\|^2 \le \alpha^{j-i}\delta^{-7n+2m} \|\boldsymbol{q}_j\|^2$.

For $\sigma < s$ we have $l' = 1$ because $i - l_\sigma \ge 2^\sigma - 1$ and $j - i < 2^{2\sigma}$. Hence $\|\boldsymbol{q}_i\|^2 \le \alpha^{j-i}\delta^{-6n} \|\boldsymbol{q}_j\|^2$. □

SLLL uses the procedure $\texttt{LLLSeg}_{l,1}$ that breaks $\texttt{locLLL}(R_{l,1})$ up into parts, each with a bounded transform $\|T_{l,1}\|_1 \le 9 \cdot 2^{n+1}$. This keeps intermediate bases of length $O(4^n M_0)$ and limits *fpa*-errors within $\texttt{LLLSeg}_{l,1}$.

$\texttt{LLLSeg}_{l,1}$ LLL-reduces the basis $R_{l,1} = \begin{bmatrix} r_{l,l} & r_{l,l+1} \\ 0 & r_{l+1,l+1} \end{bmatrix} \subset R$ after dilating row$(2, R_{l,1})$ so that $r_{l,l}/r_{l+1,l+1} \le 2^{n+1}$. After the LLL-reduction of the dilated

$R_{l,1}$ we undo the dilation, by transporting the local transform $T_{l,1} \in \mathbb{Z}^{2\times 2}$ to B. $\texttt{LLLSeg}_{l,1}$ includes global updates between local rounds.

$\texttt{LLLSeg}_{l,1}$

Given $R_{l,1}$*,* $\boldsymbol{b}_1, ..., \boldsymbol{b}_{l+1}, \boldsymbol{h}_1, ..., \boldsymbol{h}_l, \boldsymbol{r}_1, ..., \boldsymbol{r}_l$*,* $\texttt{LLLSeg}_{l,1}$ *LLL-reduces* $R_{l,1}$*.*

1. `IF` $r_{l,l}/r_{l+1,l+1} > 2^{n+1}$ `THEN` [$R'_{l,1} := R_{l,1}$,
   $\mathrm{row}(2, R'_{l,1}) :=\ \mathrm{row}(2, R'_{l,1})\, 2^{-n-1}\, r_{l,l}/r_{l+1,l+1}$ $\texttt{locLLL}(R'_{l,1})$,
   # *global update*: $[\boldsymbol{b}_l, \boldsymbol{b}_{l+1}] := [\boldsymbol{b}_l, \boldsymbol{b}_{l+1}]\, T_{l,1}$, $\texttt{TriCol}_l$, $\texttt{TriCol}_{l+1}$]
2. $\texttt{locLLL}(R_{l,1})$.

Lemma 3. $\texttt{LLLSeg}_{l,1}$ *performs* $O(nd)$ *arithmetic steps. An effectual step 1 decreases* $\mathcal{D}^{(1)}$ *by a factor* $2^{-n/2}$ *via a transform* $T_{l,1}$ *satisfying* $\|T_{l,1}\|_1 \le 9 \cdot 2^{n+1}$*.*

Proof. Consider $R'_{l,1}$ after dilation of $\mathrm{row}(2, R'_{l,1})$ which results in $r'_{l,l}/r'_{l+1,l+1} \le 2^{n+1}$. The local transform $T_{l,1}$ of $\texttt{locLLL}(R'_{l,1})$ satisfies $\|T_{l,1}\|_1 \le 9 \cdot 2^{n+1}$ using Lemma 1 with $k = 1$.

The dilated and LLL-reduced $R'_{l,1}$ satisfies $r'_{l,l}/r'_{l+1,l+1} \le \sqrt{\alpha} \le 2$. Undoing the dilation via $[\boldsymbol{b}_l, \boldsymbol{b}_{l+1}] := [\boldsymbol{b}_l, \boldsymbol{b}_{l+1}]T_{l,1}$ yields a basis $R'_{l,1}$ which is LLL-reduced after dilation. Therefore undoing the dilation shrinks $r'_{l,l}$ and $r'_{l+1,l+1}$ by factors that are bounded by the dilation factor $2^{-n-1} r_{l,l}/r_{l+1,l+1}$, and thus increases $r'_{l,l}/r'_{l+1,l+1}$ at most by the dilation factor. Hence, an effectual step 1 yields

$$r^{\mathrm{new}}_{l,l} \,/\, r^{\mathrm{new}}_{l+1,l+1} \le 2 \cdot 2^{-n-1}\, r_{l,l}/r_{l+1,l+1}.$$

It decreases $r_{l,l}/r_{l+1,l+1}$ by a factor 2^{-n}, decreases $r_{l,l}$ by a factor $2^{-n/2}$, and thus decreases $\mathcal{D}^{(1)} = \prod_{l=1}^{n-1} d_l$ by a factor $2^{-n/2}$. □

SLLL

`INPUT` $\boldsymbol{b}_1, \dots, \boldsymbol{b}_n \in \mathbb{Z}^d$ (a basis with M_0, M_1, M), δ, α, ε

`OUTPUT` $\boldsymbol{b}_1, \dots, \boldsymbol{b}_n$ size-reduced SLLL-basis for δ, ε

1. $\texttt{TriCol}_1$, $\texttt{TriCol}_2$, $l' := 2$, $s := \lceil \frac{1}{2} \log_2 n \rceil$
   # $\texttt{TriCol}_{l'}$ *has always been executed for the current* l'
2. `WHILE` $\exists\, \sigma \le s$, l, $2^\sigma(l+1) \le l'$ such that $D_{l,2^\sigma} > \alpha^{4^\sigma} \delta^{-n} D_{l+1,2^\sigma}$
   # *Clearly* $r_{1,1}, ..., r_{l',l'}$ *and thus* $D_{l,2^\sigma}$, $D_{l+1,2^\sigma}$ *are available*
   `DO` for the minimal such σ and the minimal l:
   `IF` $\sigma = 0$ `THEN` $\texttt{LLLSeg}_{l,1}$ `ELSE` $\texttt{locLLL}(R_{l,2^\sigma})$
   #*global update*: transport $T_{l,2^\sigma}$ to B, $\texttt{TriSeg}_{l,2^\sigma}$
3. `IF` $l' < n$ `THEN` $l' := l' + 1$, $\texttt{TriCol}_{l'}$, `GOTO 2.`

Correctness in ideal arithmetic. All inequalities $D_{l,2^\sigma} \le \alpha^{4^\sigma} \delta^{-n} D_{l+1,2^\sigma}$ hold upon termination of **SLLL**. As $\texttt{TriSeg}_{l,2^\sigma}$ results in size-reduced segments $B_{l,2^\sigma}$, $B_{l+1,2^\sigma}$ the terminal basis is size-reduced.

Theorem 7. *Given a basis with* M_0, M **SLLL** *finds under* fpa *of precision* $t = 7n+2\log_2 M_0$ *for* $n \geq n_0$ *an SLLL-basis for* $\delta\_$. *It runs in* $O(nd \log_2 n \log_{1/\delta} M)$ *arithmetic steps using integers and* fpa *numbers of bit length* $7n+o(n)+2\log_2 M_0$.

For $M_0 = 2^{O(n)}$ and $d = O(n)$ **SLLL** runs in $O(n^4 \log n)$ arithmetic steps, and thus in $O(n^{5+\varepsilon'})$ bit operations for every $\varepsilon' > 0$.

Proof. Time bound. It is crucial that $\mathcal{D}^{(2^\sigma)}$ does not increase within **SLLL**. `locLLL`$(R_{l,2^\sigma})$ leaves $\mathcal{D}^{(2^{\sigma'})}$ unchanged for $\sigma' > \sigma$ and does not increase $\mathcal{D}^{(2^{\sigma'})}$ for $\sigma' \leq \sigma$, because the segments $B_{l,2^\sigma}$ of level σ partition B, and this partition refines as σ decreases.

Each `locLLL`$(R_{l,2^\sigma})$ execution within **SLLL** decreases $D_{l,2^\sigma}$ and $\mathcal{D}^{(2^\sigma)}$ by a factor $\delta^{n/2}$ by the argument of Theorem 4. As initially $\mathcal{D}^{(2^\sigma)} = \prod_{l=1}^{m-1} D_{l,2^\sigma}^{m-l}$ $\leq M^{n2^{-\sigma}}$ the number of `locLLL`$(R_{l,2^\sigma})$-executions for all l is $\leq \log_{\delta^{-n/2}}(M^{n\,2^{-\sigma}})$ $= 2^{-\sigma+1}\log_{1/\delta} M$ for each $\sigma \geq 1$. Each execution requires $O(nd2^\sigma)$ global steps for `TriSeg`$_{l,2^\sigma}$, hence all executions require $O(nd \log_{1/\delta} M)$ global steps for each $\sigma \geq 1$. For $\sigma = 0$ each round of `LLLSeg`$_{l,1}$ requires $O(nd)$ arithmetic steps and decreases $\mathcal{D}^{(1)} \leq M^n$ by a factor $2^{-n/2}$ due to Lemma 3. Hence, there are at most $2\log_2 M$ rounds of `SegLLL`$_{l,1}$ for all l, requiring a total of $O(nd \log_2 M)$ global arithmetic steps. Thus there are $O(nd \log_2 n \log_{1/\delta} M)$ global arithmetic steps for all $\sigma = 0, ..., s$. The number of local steps, induced by local LLL-swaps of `locLLL`$(R_{l,2^\sigma})$, is bounded by $O(n 2^{2\sigma} \log_{1/\delta} M)$ for each $\sigma \leq s$, as for $\mathbf{SLLL}_0$ with $r = 2^\sigma$. In addition there are n `TriCol`$_l$-executions requiring $O(n^2 d)$ arithmetic steps. These steps are within the claimed step bound as $M \geq 2^n$. The required sqrt's can be computed within the claimed step bound by Newton iteration.

Correctness under fpa.We first bound the $M_{l,2^\sigma}$-value of the input $R_{l,2^\sigma}$ of `locLLL` and `LLLSeg`$_{l,1}$. If $\sigma \geq 1$ then $R_{l,2^\sigma}$ is SLLL-reduced as **SLLL** executes `locLLL`$(R_{l,2^\sigma})$ for the smallest possible σ, and thus $R_{l,2^\sigma}$, a basis of dimension $n' = 2^{\sigma+1} \leq 2\sqrt{n}$, is SLLL-reduced as the inequalities of Def.3 already hold for $\sigma' \leq \lceil \frac{1}{2}(\sigma+1) \rceil = \lceil \frac{1}{2}\log_2 n' \rceil$. Therefore, $R_{l,2^\sigma}$ satisfies by Lemma 2 :
$M_{l,2^\sigma} \leq \alpha^{2^{\sigma+1}} \delta^{-7n} \leq 2^n$ for $\delta \geq 0.96$, $\alpha \leq \sqrt{2}$, $2^\sigma \leq 2\sqrt{n}$ and $n \geq 16$.

If $\sigma = 0$ the execution of `LLLSeg`$_{l,1}$ on the dilated input $R'_{l,1}$ performs by Lemma 3 a transform $T_{l,1}$ with $\|T_{l,1}\|_1 \leq 9 \cdot 2^{n+1}$ and the dilated $R'_{l,1}$ satisfies $M_1(R'_{l,1}) \leq 2^{n+1}$.

The fpa-errors of $R_{l,2^\sigma}$, $R'_{l,1}$ *within* **SLLL**. When $r_{i,l}$ is used the basis $\boldsymbol{b}_1, ..., \boldsymbol{b}_{l-1}$ already satisfies the bounds of Lemma 2 and $r_{l-1,l-1}/r_{l,l} \leq 2^{n+1}$ holds after dilation of $R'_{l,1}$. The initial $r_{i,l}$ resulting from `TriCol`$_1$,..., `TriCol`$_l$ satisfies the inequalities (1),(3),(4) with $\bar{M}_0 \leq M_0$ and $\bar{M}_1^2 \leq \alpha^l \delta^{-7n} 4^n \leq 2^{3n}$. Hence, the initial *fpa*-error of $\bar{\mu}_{l,i}$ is bounded according to (4) by $O(d 7^l M_0^2 2^{-3n} 2^{-t})$.

The loss of precision within `locLLL`$(R_{l,2^\sigma})$ described in Cor.1(1) gets corrected by the global update subsequent to `locLLL`$(R_{l,2^\sigma})$. We see that **SLLL** is correct using *fpa* with $t = 7n + o(n) + 2\log_2 M_0$ precision bits.

By Lemma 1 and the argument of Theorem 2 all intermediate basis vectors have length bounded by $2^n M_0 \|T_{l,2^\sigma}\|_1 = 2^{2n+o(n)} M_0$. Therefore, all integers and *fpa*-numbers in **SLLL** have bit length $\leq 7n + o(n) + 2\log_2 M_0$. □

SLLL-bases versus LLL-bases. LLL-bases with δ satisfy the inequalities of Theorem 6 with δ replaced by 1. Thus $\|\boldsymbol{b}_j\|$ approximates λ_j to within a factor $\alpha^{\frac{n-1}{2}}$ for LLL-bases, resp., within a factor $(\alpha/\delta^7)^{\frac{n-1}{2}}$ for SLLL-bases. However, SLLL-bases for $\delta' = \delta^{1/8}$ are "better" than LLL-bases for δ, in the sense that they guarantee a smaller length defect, because $\alpha'/\delta'^7 = \frac{1}{\delta'^8 - \delta'^7/4} = \frac{1}{\delta - \delta'^7/4} < \frac{1}{\delta - 1/4} = \alpha$.

Dependence of time bounds on δ. The time bounds contain a factor $\log_{1/\delta} 2$,

$$\log_{1/\delta} 2 = \log_2(e)/\ln(1/\delta) \leq \log_2(e)\frac{\delta}{1-\delta},$$

since $\ln(1/\delta) \geq 1/\delta - 1$. We see that replacing δ by $\sqrt{\delta}$ essentially halves $1-\delta$ and doubles the SLLL-time bound. Hence, replacing δ by $\delta^{1/8}$ increases the **SLLL**-time bound at most by a factor 3. In practice, the LLL-time may increase slower than by the factor $\frac{\delta}{1-\delta}$ as δ approaches 1, see [KS01b, Fig.3].

Reducing a generator system. There is an algorithm **SLLL'** that, given a generator matrix $B \in \mathbb{Z}^{d\times n}$ of arbitrary rank $\leq n$, transforms B with the performance of **SLLL**, into an SLLL-basis for $\delta\_$ of the lattice generated by the columns of B.

6 SLLL-Reduction via Iterated Subsegments.

We present a variant of SLLL-reduction that extends LLL-operations stepwise to larger and larger submatrices $R_{l,2^\sigma} \subset R$ by transporting local transforms from level $\sigma - 1$ to level σ recursively for $\sigma = 1, ..., s$, where $n = 2^s$. Local LLL-reduction and the transport of local LLL-transforms is done by the new procedure `locSLLL`$(R_{l,2^\sigma})$ that recursively executes `locSLLL`$(R_{l',2^{\sigma-1}})$ for $l' = 2l-1, 2l, 2l+1$. **SLLL**$^+$ does not iterate the global procedure `TriSeg` iterating instead the faster local procedure `locTri`.

Unfortunately **SLLL**$^+$ seems to require under *fpa* $t = O(\log(M_0 M_1)) = O(n \log M_0)$ precision bits to cover the *fpa*-errors that get accumulated by the initial `TriSeg` and by iterating `locTri`. Obviously, $t = O(n \log M_0)$ precision bits erase under *fpa* the advantage of **SLLL**$^+$ over **SLLL**. **SLLL**$^+$ essentially saves a factor n in the number of arithmetic steps compared to **SLLL** but requires *fpa*-numbers that are n-times longer. We can reduce t by using Scaled LLL-reduction of [KS01b], and by a novel partitioning the SLLL$^+$-reduction into transforms $T_{l,2^\sigma}$ with small norm and correcting $R_{l,2^\sigma} T_{l,2^\sigma}$ by a global update. We plan to include this into a separate paper.

Here we merely analyse **SLLL**$^+$ in ideal real arithmetic. **SLLL**$^+$ runs in $O(n^2 d + n \log_2 n \log_{1/\delta} M)$ arithmetic steps, e.g. for $M_0 = 2^{O(n)}$ and $d = O(n)$ it runs in $O(n^3 \log n)$ arithmetic steps.

Definition 4. *A basis* $\boldsymbol{b}_1, \ldots, \boldsymbol{b}_n \in \mathbb{Z}^d$ *with* $n = 2^s$ *is an* SLLL$^+$-basis (or SLLL$^+$-reduced) *for* δ *if it satisfies for* $\sigma = 0, ..., s = \log_2 n$

$$D_{l,2^\sigma} \leq (\alpha/\delta)^{4^\sigma} D_{l+1,2^\sigma} \quad \textit{for} \text{ odd } l \in [1, n/2^\sigma[. \tag{10}$$

Unlike to Def.2 and Def.3 the inequalities (10) are not required for *even* l, this opens new efficiencies for SLLL$^+$-reduction. The inequalities (10) hold for each σ and odd l locally in double segments $[B_{l,2^\sigma}, B_{l+1,2^\sigma}]$, they do not bridge these pairwise disjoint double segments. For $\sigma = 0$ the inequalities (10) mean that $\|\boldsymbol{q}_l\|^2 \leq \alpha/\delta \, \|\boldsymbol{q}_{l+1}\|^2$ holds for odd l.

The inequalities (10) are preserved under duality. If $\boldsymbol{b}_1, ..., \boldsymbol{b}_n$ is an SLLL$^+$-basis then so is the dual basis $\boldsymbol{b}_1^*, ..., \boldsymbol{b}_n^*$. We next extend Theorem 3, and show that the first vector of an SLLL$^+$-basis is almost as short relative to $(\det \mathcal{L})^{\frac{2}{n}}$ as for LLL-bases.

Theorem 8. *Every SLLL$^+$-basis* $\boldsymbol{b}_1, \ldots, \boldsymbol{b}_n$*, where* n *is a power of 2 satisfies* $\|\boldsymbol{b}_1\| \leq (\alpha/\delta)^{\frac{n-1}{4}} (\det \mathcal{L})^{\frac{1}{n}}$ *and* $\|\boldsymbol{q}_n\| \geq (\delta/\alpha)^{\frac{n-1}{4}} (\det \mathcal{L})^{\frac{1}{n}}$.

Proof. Using the inequalities (10) merely for $l = 1$ we prove by induction on σ that $\|\boldsymbol{b}_1\|^{2^{\sigma+1}} \leq (\alpha/\delta)^{4^\sigma/2 - 2^{\sigma-1}} D_{1,2^\sigma}$ holds for $\sigma = 0, ..., s = \log_2 n$.

For $\sigma = s$ this proves the first claim of the theorem as $D_{1,2^s} = (\det \mathcal{L})^2$ and $4^s 2^{-s-1} - \frac{1}{2} = \frac{n-1}{2}$. The second claim holds by duality.

The induction claim for $\sigma = 0$ means that $\|\boldsymbol{b}_1\|^2 \leq \|\boldsymbol{b}_1\|^2$ as $4^0/2 - \frac{1}{2} = 0$.

Induction from σ *to* $\sigma + 1$. By SLLL$^+$-reducedness we have that $D_{1,2^\sigma} \leq (\alpha/\delta)^{4^\sigma} D_{2,2^\sigma}$. We multiply both sides by $D_{1,2^\sigma}$ then the equation $D_{1,2^{\sigma+1}} = D_{1,2^\sigma} D_{2,2^\sigma}$ yields $D_{1,2^\sigma}^2 \leq (\alpha/\delta)^{4^\sigma} D_{1,2^{\sigma+1}}$.

This and the squared induction hypothesis for σ implies

$$\|\boldsymbol{b}_1\|^{2^{\sigma+2}} \leq (\alpha/\delta)^{4^\sigma - 2^\sigma} (\alpha/\delta)^{4^\sigma} D_{1,2^{\sigma+1}}.$$

This proves the claim for $\sigma + 1$ since $4^\sigma - 2^\sigma + 4^\sigma = 4^{\sigma+1}/2 - 2^\sigma$. □

Let the given basis $\boldsymbol{b}_1, ..., \boldsymbol{b}_n \in \mathbb{Z}^d$ have GNF $R \in \mathbb{R}^{n \times n}$. The local procedures `locSLLL`$(R_{l,2^\sigma})$, `locTri`$(R_{l,2^\sigma})$ are given for input on transformed submatrices $R_{l,2^\sigma} = R'_{l,2^\sigma} T_{l,2^\sigma}$, where $R'_{l,2^\sigma}$ is the initial submatrix $R_{l,2^\sigma}$ of R and $T_{l,2^\sigma}$ is the currently performed transform. We let `locSLLL`$(R_{l,1})$ coincide with `locLLL`$(R_{l,1})$, and we recursively define `locSLLL`$(R_{l,2^\sigma})$ for $\sigma = 1, ..., s$.

`locSLLL`$(R_{l,2^\sigma})$ (`locSLLL`$_{l,2^\sigma}$ for short)

\# `locSLLL`$_{l,2^\sigma}$ *locally SLLL$^+$-reduces* $R_{l,2^\sigma}$ *and updates the local transform* $T_{l,2^\sigma}$. *Note that* $R_{l',2^{\sigma-1}} \subset R_{l,2^\sigma}$ *iff* $l' \in \{2l-1, 2l, 2l+1\}$.

1. $T_{l,2^\sigma} := I_{2^{\sigma+1}}, \ l' := 2l - 1$

   \# $T_{l,2^\sigma}$ *is always updated to be the product of all previous transforms* $T_{l',2^{\sigma'}}$ *for* $\sigma' < \sigma$ *performed within* `locSLLL`$_{l,2^\sigma}$.

2. `WHILE` $l' < 2l + 1$ `DO`

   copy $R_{l',2^{\sigma-1}}$ from $R_{l,2^\sigma}$

$\mathtt{locSLLL}_{l',2^{\sigma-1}}$, transport $T_{l',2^{\sigma-1}}$ to $R_{l,2^\sigma}$ and $T_{l,2^\sigma}$,

$\mathtt{locTri}(R_{l,2^\sigma})$, update $T_{l,2^\sigma}$ for the size-reduction performed by `locTri`

`IF` $l' \geq 2l$ `and` $D_{l'-1,2^{\sigma-1}} > (\alpha/\delta)^{4^{\sigma-1}} D_{l',2^{\sigma-1}}$

`THEN` $l' := l' - 1$ `ELSE` $l' := l' + 1$.

$\mathtt{locTri}(R_{l,2^\sigma})$

$\mathtt{locTri}(R_{l,2^\sigma})$ *locally triangulates and size-reduces* $R_{l,2^\sigma}$ *using* $O(2^{3\sigma})$ *arithmetic steps.*

1. Produce a copy $[\boldsymbol{b}'_1, ..., \boldsymbol{b}'_{2^{\sigma+1}}]$ of $R_{l,2^\sigma}$
2. `FOR` $i = 1, ..., 2^{\sigma+1}$ `DO` $\mathtt{TriCol}(\boldsymbol{b}'_1 ..., \boldsymbol{b}'_i, \boldsymbol{h}'_1, ..., \boldsymbol{h}'_{i-1}, \boldsymbol{r}'_1, ..., \boldsymbol{r}'_{i-1})$
3. $D_{j,2^\sigma} := \prod_{i=0}^{2^\sigma-1} r_{2^\sigma j-i,2^\sigma j-i}$, for $j = l, l+1$.

Correctness of $\mathtt{locSLLL}_{l,2^\sigma}$. We see by induction on σ that upon termination of $\mathtt{locSLLL}_{l,2^\sigma}$ the basis $R_{l,2^\sigma}$ is SLLL$^+$-reduced, upper-triangular and size-reduced; its local transform is stored in $T_{l,2^\sigma}$. Local triangulation of a transformed $R_{l,2^\sigma} T_{l,2^\sigma}$ results in the same submatrix $R_{l,2^\sigma} \subset R$ obtained by global triangulation of the transformed B via $\mathtt{TriSeg}_{1,n/2}$.

Upon termination the inequalities (10) hold locally within $R_{l,2^\sigma}$ for even and odd l, but possibly $D_{4l,2^{\sigma-2}} > (\alpha/\delta)^{4^{\sigma-2}} D_{4l+1,2^{\sigma-2}}$ since the final $\mathtt{locSLLL}_{2l+1,2^{\sigma-1}}$-execution may revers the inequality $D_{4l,2^{\sigma-2}} \leq (\alpha/\delta)^{4^{\sigma-2}} D_{4l+1,2^{\sigma-2}}$.

SLLL$^+$

`INPUT` $\boldsymbol{b}_1, \ldots, \boldsymbol{b}_n \in \mathbb{Z}^d$ (a basis with M), $n = 2^s$, δ

`OUTPUT` $\boldsymbol{b}_1, \ldots, \boldsymbol{b}_n$ a size-reduced SLLL$^+$-basis

1. # *compute* $R_{1,n/2}$: $\mathtt{TriSeg}_{1,n/2}$
2. $\mathtt{locSLLL}(R_{1,n/2})$, # *global update* : $B := B\, T_{1,n/2}$

Correctness of **SLLL$^+$** follows from the correctness of $\mathtt{locSLLL}_{1,n/2}$.

Theorem 9. *In ideal arithmetic* **SLLL$^+$** *computes a size-reduced SLLL$^+$-basis for* δ *and runs in* $O(n^2 d + n \log_2 n \log_{1/\delta} M)$ *arithmetic steps.*

Proof. For $\sigma = 0, ..., s-1$ let $\#_{2^\sigma}$ denote the number of $\mathtt{locSLLL}_{l,2^\sigma}$-executions in **SLLL$^+$** due to $D_{l,2^\sigma} > (\alpha/\delta)^{4^\sigma} D_{l+1,2^\sigma}$ for all l. By the argument of Theorem 4 each $\mathtt{locSLLL}_{l,2^\sigma}$ execution counted in $\#_{2^\sigma}$ decreases $\mathcal{D}^{(2^\sigma)}$ by the factor $\delta^{4^\sigma/2}$. Initially the integer $\mathcal{D}^{(2^\sigma)}$ satisfies $\mathcal{D}^{(2^\sigma)} \leq M^{n/2^\sigma}$, and upon termination $\mathcal{D}^{(2^\sigma)} \geq 1$, hence $\#_{2^\sigma} \leq 2n \cdot 2^{-3\sigma} \log_{1/\delta} M$.

Each of the $\mathtt{locSLLL}_{l',2^{\sigma-1}}$-executions within $\mathtt{locTri}(R_{l,2^\sigma})$ requires an overhead of $O(2^{3\sigma})$ arithmetic steps. This covers the matrix transports and the subsequent $\mathtt{locTri}(R_{l,2^\sigma})$-execution. The very first $\mathtt{locSLLL}_{l',2^{\sigma-1}}$ -execution within $\mathtt{locSLLL}_{l,2^\sigma}$ is possibly not counted in $\#_{2^{\sigma-1}}$. We allocate its overhead of $O(2^{3\sigma})$ steps to the overhead of $\mathtt{locSLLL}_{l,2^\sigma}$. We see that the total overhead of all $\mathtt{locSLLL}_{l,2^\sigma}$-executions is $O(2^{3\sigma} + n \log_{1/\delta} M)$ for each $\sigma \leq s$.

Moreover, the initial $\texttt{TriSeg}_{1,n/2}$ and the final update $B := B\,T_{1,n/2}$ require $O(n^2\,d)$ arithmetic steps. We see that **SLLL**$^+$ runs in $O(n^2 d + n\,\log_2 n \log_{1/\delta} M)$ arithmetic steps, where $s = \log_2 n$. □

Further improvements of **SLLL**$^+$. It is still possible to improve the time bound of **SLLL**$^+$ via modular reduction and fast matrix multiplication following [St96]. But this will hardly be practical. Other variants of **SLLL**$^+$ are more promising. **SLLL**$^+$ can be modified to achieve the length defect of SLLL-bases. This is possible by the concept of *strong* SLLL-reduction of [KS02]. Practicability requires an SLLL$^+$-algorithm that runs under *fpa* of $t = O(n + \log_2 M_0)$ precision bits instead of the straightforward method with $t = O(n \log_2 M_0)$. We plan to continue in this direction.

Acknowledgement. Damien Stehle, a student of Phong Nguyen, informed me that I previously misused the analysis of Householder transforms in [LH95]. Prop. 1 corrects this error.

References

[Aj98] *M. Ajtai*, The Shortest Vector Problem in L_2 is NP-hard for Randomized Reductions. Proc. 30th STOC, pp. 10–19, 1998.

[Aj03] *M. Ajtai*, The Worst-case Behavior of Schnorr's Algorithm Approximating the Shortest Nonzero Vector in a Lattice. Proc. 35th STOC, pp. 396–406 2003.

[AKS01] *M. Ajtai, R. Kumar, and D. Sivakumar*, A Sieve Algorithm for the Shortest Lattice Vector Problem. Proc. 33th STOC, pp. 601–610, 2001.

[Ak02] *A. Akhavi*, Random Lattices, Threshold Phenomena and Efficient Reduction Algorithms. *Theoret. Comput. Sci.*, **287**, pp. 359–385, 2002.

[BM03] *J. Blömer and A. May* New Partial Key Exposure Attacks on RSA. Proc. Crypto'2003, Lecture Notes in Comp.Sci., 2729, Springer, New York, pp. 27 - 43, 2003.

[BN00] *D. Bleichenbacher and P.Q. Nguyen*, Noisy Polynomial Interpolation and Noisy Chinese Remaindering. Eurocrypt 2000, Lecture Notes in Comput. Sci., 1807, Springer, New York, pp. 53-69, 2000.

[BS99] *J. Blömer and J.P. Seifert*, On the Complexity of Computing Short Linearly Independent Vectors and Short Bases in a Lattice. Proc. 31th STOC, pp. 711–720, 1999.

[Bo00] *D. Boneh*, Finding Smooth Integers in Small Intervals Using CRT Decoding. Proc. 32th STOC, pp. 265-272, 2000.

[Ca00] *J. Cai*, The Complexity of some Lattice Problems. Algorithmic Number Theory, Lecture Notes in Comput. Sci., 1838, Springer, New York, pp. 1-32, 2000.

[Co97] *D. Coppersmith*, Small Solutions to Polynomial Equations, and Low Exponent RSA Vulnerabilities. *J. Cryptology*, **10**, pp. 233-260, 1997.

[Co01] *D. Coppersmith*, Finding Small Solutions to Small Degree Polynomials. Cryptography and Lattices, Lecture Notes in Comput. Sci., Springer, New York, 2146, pp. 20-31, 2001.

[DV94] *H. Daudé and B. Vallée*, An Upper Bound on the Average Number of Iterations of the LLL algorithm, *Theoret. Comput. Sci.*, **123**, pp. 395–115, 1994.

[E81] *P. van Emde Boas*, Another NP-Complete Partition Problem and the Complexity of Computing Short Vectors in a Lattice. Mathematics Department, University of Amsterdam, TR 81-04, 1981.

[Ga801] *C. F. Gauss*, Disquisitiones Arithmeticae. 1801; English transl., Yale Univ. Press, New Haven, Conn. 1966.

[He85] *B. Helfrich*, Algorithms to Construct Minkowski Reduced and Hermite Reduced Bases. *Theor. Comput. Sci.* **41**, pp. 125–139, 1985.

[HJLS89] *J. Håstad, B. Just, J.C. Lagarias und C.P. Schnorr*, Polynomial Time Algorithms for Finding Integer Relations Among Real Numbers, SIAM Journal on Computing, **18**, pp. 859–881, 1989.

[K84] *R. Kannan*, Minkowski's Convex Body Theorem and Integer Programming. Math. Oper. Res., **12**, pp. 415–440, 1984.

[KS01a] *H. Koy and C.P. Schnorr*, Segment LLL-Reduction. Cryptography and Lattices, Lecture Notes in Comput. Sci., 2146, Springer, New York, pp.67–80, 2001. //www.mi.informatik.uni-frankfurt.de/research/papers.html

[KS01b] *H. Koy and C.P. Schnorr*, Segment LLL-Reduction with Floating Point Orthogonalization. Cryptography and Lattices, Lecture Notes in Comput. Sci., 2146, Springer, New York, pp. 81–96, 2001. //www.mi.informatik.uni-frankfurt.de/research/papers.html

[KS02] *H. Koy and C.P. Schnorr*, Segment and Strong Segment LLL-Reduction of Lattice Bases. TR Universität Franfurt, April 2002, //www.mi.informatik.uni-frankfurt.de/research/papers.html

[Le83] *H.W. Lenstra, Jr.*, Integer Programming With a Fixed Number of Variables. Math. Oper. Res. 8, pp. 538-548, 8, 1983.

[LH95] *C.L. Lawson and R.J. Hanson*, Solving Least Square Problems. Siam Publications, 1995 (first published by Prentice Hall, New Jersey 1974).

[LLL82] *A. K. Lenstra, H. W. Lenstra and L. Lovász*, Factoring Polynomials with Rational Coefficients. *Math. Ann.*, **261**, pp. 515-534, 1982.

[Lo86] *L. Lovász*, An Algorithmic Theory of Numbers, Graphs and Convexity, CBMS-NSF Regional Conference Series in Applied Mathematics, **50**, SIAM Publications, Philadelphia, 1986

[Ma03] *A. May*, New RSA Vulnerabilities Using Lattice Reduction Methods. Dissertation Thesis, University of Paderborn, October 2003.

[Mi01] *D. Micciancio*, A Linear Space Algorithm for Computing the Hermite Normal Form, Proceedings ISSAC 2001, Lecture Notes in Computer Sci., 2146, Springer, New York, pp. 126 –145, 2001.

[MG02] *D. Micciancio and S. Goldwasser*, Complexity of Lattice Problems, A Cryptographic Perspective. Kluwer Academic Publishers, London, 2002.

[ML01] *S. Mehrotra and Z. Li*, Reduction of Lattice Bases Using Modular Arithmetic. TR. Dept. of Industrial Engeneering and Management Sciences, Northwestern University, Evanston, Il. Oct 2001, mehrotra, zhifeng@iems.nwu.edu.

[NS00] *P.Q. Nguyen and J. Stern*, Lattice Reduction in Cryptology, An Update. Algorithmic Number Theory, Lecture Notes in Comput. Sci., 1838, Springer, New York, pp. 85-112, 2000. full version http://www.di.ens.fr/pnguyen,stern/

[S87] *C.P. Schnorr*, A Hierarchy of Polynomial Time Lattice Basis Reduction Algorithms. *Theoret. Comput. Sci.*, **53**, pp. 201–224, 1987.

[S88] *C.P. Schnorr*, A More Efficient Algorithm for Lattice Reduction. *J. of Algor.* **9**, 47–62, 1988.

[SE91] *C.P. Schnorr and M. Euchner*, Lattice Basis Reduction and Solving Subset Sum Problems. Fundamentals of Comput. Theory, Lecture Notes in Comput.

Sci., 591, Springer, New York, pp. 68–85, 1991. The complete paper appeared in Math. Programming Studies, **66A**, 2, 1994, pp. 181–199.

[S03] *C.P. Schnorr*, Lattice Reduction by Random Sampling and Birthday Methods. Proc. STACS 2003, Eds. H. Alt and M. Habib, Lecture Notes in Comput. Sci., 2607, Springer, New York, pp. 145–156, 2003.

[S04] *C.P. Schnorr*, Gittertheorie und Algorithmische Geometrie. Lecture Notes Universität Frankfurt, Frankfurt, 2004. //www.mi.informatik.uni-frankfurt.de/index.html#teaching

[Sc84] *A. Schönhage*, Factorization of Univariate Integer Polynomials by Diophantine Approximation and Improved Lattice Basis Reduction Algorithm. *Proc. 11-th Coll. Automata, Languages and Programming, Antwerpen 1984*, Lecture Notes in Comput. Sci., 172, Springer, New York, pp. 436–447, 1984.

[St96] *A. Storjohann*, Faster Algorithms for Integer Lattice Basis Reduction. TR 249, Swiss Federal Institute of Technology, ETH-Zurich, Department of Computer Science, Zurich, Switzerland, July 1996. //www.inf.ethz.ch/research/publications/html.

[Wi65] *J.H. Wilkinson*, The Algebraic Eigenvalue Problem. Clarendon Press, Oxford, 1965.