Chapter 17

Lattice Basis Reduction

The goal of lattice basis reduction is to transform a given lattice basis into a "nice" lattice basis consisting of vectors that are short and close to orthogonal. To achieve this one needs both a suitable mathematical definition of "nice basis" and an efficient algorithm to compute a basis satisfying this definition.

Reduction of lattice bases of rank 2 in $\mathbb{R}^2$ was given by Lagrange[1] and Gauss. The algorithm is closely related to Euclid's algorithm and we briefly present it in Section 17.1. The main goal of this section is to present the lattice basis reduction algorithm of Lenstra, Lenstra and Lovász, known as the LLL or L^3 algorithm.[2] This is a very important algorithm for practical applications. Some basic references for the LLL algorithm are Section 14.3 of Smart [571], Section 2.6 of Cohen [135] and Chapter 17 of Trappe and Washington [608]. More detailed treatments are given in von zur Gathen and Gerhard [237], Grötschel, Lovász and Schrijver [268], Section 1.2 of Lovász [394], and Nguyen and Vallée [463]. I also highly recommend the original paper [372].

The LLL algorithm generalises the Lagrange-Gauss algorithm and exploits the Gram-Schmidt orthogonalisation. Note that the Gram-Schmidt process is not useful, in general, for lattices since the coefficients $\mu_{i,j}$ do not usually lie in $\mathbb{Z}$ and so the resulting vectors are not usually elements of the lattice. The LLL algorithm uses the Gram-Schmidt vectors to determine the quality of the lattice basis, but ensures that the linear combinations used to update the lattice vectors are all over $\mathbb{Z}$.

[1] The algorithm was first written down by Lagrange and later by Gauss, but is usually called the "Gauss algorithm". We refer to [454] or Chapter 2 of [463] for the original references.

[2] Chapter 1 of [463] gives an excellent survey of the historical development of the algorithm.

17.1 Lattice Basis Reduction in Two Dimensions

Let $\underline{b}_1, \underline{b}_2 \in \mathbb{R}^2$ be linear independent vectors and denote by L the lattice for which they are a basis. The goal is to output a basis for the lattice such that the lengths of the basis vectors are as short as possible (in this case, successive minima). Lagrange and Gauss gave the following criteria for a basis to be reduced and then developed Algorithm 23 to compute such a basis.

Definition 17.1.1. An ordered basis $\underline{b}_1, \underline{b}_2$ for $\mathbb{R}^2$ is **Lagrange-Gauss reduced** if $\|\underline{b}_1\| \leq \|\underline{b}_2\| \leq \|\underline{b}_2 + q\underline{b}_1\|$ for all $q \in \mathbb{Z}$.

The following theorem shows that the vectors in a Lagrange-Gauss reduced basis are as short as possible. This result holds for any norm, though the algorithm presented below is only for the Euclidean norm.

Theorem 17.1.2. *Let λ_1, λ_2 be the successive minima of L. If L has an ordered basis $\{\underline{b}_1, \underline{b}_2\}$ that is Lagrange-Gauss reduced then $\|\underline{b}_i\| = \lambda_i$ for $i = 1, 2$.*

Proof: By definition we have

$$\|\underline{b}_2 + q\underline{b}_1\| \geq \|\underline{b}_2\| \geq \|\underline{b}_1\|$$

for all $q \in \mathbb{Z}$.

Let $\underline{v} = l_1\underline{b}_1 + l_2\underline{b}_2$ be any non-zero point in L. If $l_2 = 0$ then $\|\underline{v}\| \geq \|\underline{b}_1\|$. If $l_2 \neq 0$ then write $l_1 = ql_2 + r$ with $q, r \in \mathbb{Z}$ such that $0 \leq r < |l_2|$. Then $\underline{v} = r\underline{b}_1 + l_2(\underline{b}_2 + q\underline{b}_1)$ and, by the triangle inequality

$$\begin{aligned} \|\underline{v}\| &\geq |l_2|\,\|\underline{b}_2 + q\underline{b}_1\| - r\|\underline{b}_1\| \\ &= (|l_2| - r)\|\underline{b}_2 + q\underline{b}_1\| + r(\|\underline{b}_2 + q\underline{b}_1\| - \|\underline{b}_1\|) \\ &\geq \|\underline{b}_2 + q\underline{b}_1\| \geq \|\underline{b}_2\| \geq \|\underline{b}_1\|. \end{aligned}$$

This completes the proof. □

Definition 17.1.3. Let $\underline{b}_1, \ldots, \underline{b}_n$ be a list of vectors in $\mathbb{R}^n$. We write[3] $B_i = \|\underline{b}_i\|^2 = \langle \underline{b}_i, \underline{b}_i \rangle$.

A crucial ingredient for the Lagrange-Gauss algorithm is that

$$\|\underline{b}_2 - \mu\underline{b}_1\|^2 = B_2 - 2\mu\langle \underline{b}_1, \underline{b}_2 \rangle + \mu^2 B_1 \tag{17.1}$$

is minimised at $\mu = \langle \underline{b}_1, \underline{b}_2 \rangle / B_1$ (to see this, note that the graph as a function of μ is a parabola and that the minimum can be found by differentiating with respect to μ). Since we are working in a lattice we therefore replace $\underline{b}_2$ by $\underline{b}_2 - \lfloor \mu \rceil \underline{b}_1$ where $\lfloor \mu \rceil$ is the nearest integer to μ. Hence lines 3 and 9 of Algorithm 23 reduce the size of $\underline{b}_2$ as much as possible using $\underline{b}_1$. In the one-dimensional case the formula $\underline{b}_2 - \lfloor \mu \rceil \underline{b}_1$ is the familiar operation $r_{i+1} = r_{i-1} - \lfloor r_{i-1}/r_i \rceil r_i$ from Euclid's algorithm.

Lemma 17.1.4. *An ordered basis $\{\underline{b}_1, \underline{b}_2\}$ is Lagrange-Gauss reduced if and only if*

$$\|\underline{b}_1\| \leq \|\underline{b}_2\| \leq \|\underline{b}_2 \pm \underline{b}_1\|.$$

Proof: The forward implication is trivial. For the converse, suppose $\|\underline{b}_2\| \leq \|\underline{b}_2 \pm \underline{b}_1\|$. We use the fact that the graph of $F(\mu) = \|\underline{b}_2 + \mu\underline{b}_1\|^2$ is a parabola. It follows that the

Algorithm 23 Lagrange-Gauss lattice basis reduction

INPUT: Basis $\underline{b}_1, \underline{b}_2 \in \mathbb{Z}^2$ for a lattice L
OUTPUT: Basis $(\underline{b}_1, \underline{b}_2)$ for L such that $\|\underline{b}_i\| = \lambda_i$

$B_1 = \|\underline{b}_1\|^2$
$\mu = \langle \underline{b}_1, \underline{b}_2 \rangle / B_1$
$\underline{b}_2 = \underline{b}_2 - \lfloor \mu \rceil \underline{b}_1$
$B_2 = \|\underline{b}_2\|^2$
while $B_2 < B_1$ **do**
    Swap $\underline{b}_1$ and $\underline{b}_2$
    $B_1 = B_2$
    $\mu = \langle \underline{b}_1, \underline{b}_2 \rangle / B_1$
    $\underline{b}_2 = \underline{b}_2 - \lfloor \mu \rceil \underline{b}_1$
    $B_2 = \|\underline{b}_2\|^2$
end while
return $(\underline{b}_1, \underline{b}_2)$

miniumum of $F(\mu)$ is taken for $-1 < \mu < 1$. Hence $\|\underline{b}_2\| \leq \|\underline{b}_2 + q\underline{b}_1\|$ for $q \in \mathbb{Z}$ such that $|q| > 1$. □

Algorithm 23 gives the Lagrange-Gauss algorithm for lattices in $\mathbb{Z}^2$. Note that the computation of μ is as an exact value in $\mathbb{Q}$. All other arithmetic is exact integer arithmetic.

Lemma 17.1.5. *Algorithm 23 terminates and outputs a Lagrange-Gauss reduced basis for the lattice L.*

Exercise 17.1.6. Prove Lemma 17.1.5.

Example 17.1.7. We run the Lagrange-Gauss algorithm on $\underline{b}_1 = (1, 5)$ and $\underline{b}_2 = (6, 21)$. In the first step, $\mu = 111/26 \approx 4.27$ and so we update $\underline{b}_2 = \underline{b}_2 - 4\underline{b}_1 = (2, 1)$. We then swap $\underline{b}_1$ and $\underline{b}_2$ so that the values in the loop are now $\underline{b}_1 = (2, 1)$ and $\underline{b}_2 = (1, 5)$. This time, $\mu = 7/5 = 1.4$ and so we set $\underline{b}_2 = \underline{b}_2 - \underline{b}_1 = (-1, 4)$. Since $\|\underline{b}_2\| > \|\underline{b}_1\|$ the algorithm halts and outputs $\{(2, 1), (-1, 4)\}$.

Exercise 17.1.8. Run the Lagrange-Gauss reduction algorithm on the basis $\{(3, 8), (5, 14)\}$.

Lemma 17.1.9. *Let $\underline{b}_1, \underline{b}_2$ be the initial vectors in an iteration of the Lagrange-Gauss algorithm and suppose $\underline{b}'_1 = \underline{b}_2 - m\underline{b}_1$ and $\underline{b}'_2 = \underline{b}_1$ are the vectors that will be considered in the next step of the algorithm. Then $\|\underline{b}'_1\|^2 < \|\underline{b}_1\|^2/3$, except perhaps for the last two iterations.*

Proof: Note that $m = \lfloor \mu \rceil = \lfloor \langle \underline{b}_1, \underline{b}_2 \rangle / \langle \underline{b}_1, \underline{b}_1 \rangle \rceil = \langle \underline{b}_1, \underline{b}_2 \rangle / \langle \underline{b}_1, \underline{b}_1 \rangle + \epsilon$ where $|\epsilon| \leq 1/2$. Hence,

$$\langle \underline{b}_1, \underline{b}'_1 \rangle = \langle \underline{b}_1, \underline{b}_2 - (\langle \underline{b}_1, \underline{b}_2 \rangle / \langle \underline{b}_1, \underline{b}_1 \rangle + \epsilon)\, \underline{b}_1 \rangle = -\epsilon \langle \underline{b}_1, \underline{b}_1 \rangle = -\epsilon \|\underline{b}_1\|^2.$$

We show that $\|\underline{b}'_1\|^2 < \|\underline{b}_1\|^2/3$ unless we are in the last two iterations of the algorithm. To do this, suppose that $\|\underline{b}'_1\|^2 \geq \|\underline{b}_1\|^2/3$. Then

$$|\langle \underline{b}'_1, \underline{b}'_2 \rangle| = |\langle \underline{b}'_1, \underline{b}_1 \rangle| = |\epsilon| \|\underline{b}_1\|^2 \leq \tfrac{1}{2} \|\underline{b}_1\|^2 \leq \tfrac{3}{2} \|\underline{b}'_1\|^2.$$

It follows that, in the next iteration of the algorithm, we will be taking $m = \lfloor \mu \rceil \in \{-1, 0, 1\}$ and so the next iteration would, at most, replace $\underline{b}'_1$ with $\underline{b}'_2 \pm \underline{b}'_1 = \underline{b}_1 \pm (\underline{b}_2 - m\underline{b}_1)$. But, if this were smaller than $\underline{b}'_1$ then we would have already computed $\underline{b}'_1$ differently in the current iteration. Hence, the next step is the final iteration. □

[3]The reader is warned that the notation B_i will have a different meaning when we are discussing the LLL algorithm.

Theorem 17.1.10. *Let $X \in \mathbb{Z}_{\geq 2}$ and let $\underline{b}_1, \underline{b}_2$ be vectors in $\mathbb{Z}^2$ such that $\|\underline{b}_i\|^2 \leq X$. Then the Lagrange-Gauss algorithm performs $O(\log(X)^3)$ bit operations.*

Proof: Lemma 17.1.9 shows that there are $O(\log(X))$ iterations in the Lagrange-Gauss algorithm. Since the squared Euclidean lengths of all vectors in the algorithm are bounded by X, it follows that entries of vectors are integers bounded by $\sqrt{X}$. Similarly, the numerator and denominator of $\mu \in \mathbb{Q}$ require $O(\log(X))$ bits. The result follows. □

A much more precise analysis of the Lagrange-Gauss reduction algorithm is given by Vallée [613]. Indeed, the algorithm has complexity $O(\log(X)^2)$ bit operations; see Nguyen and Stehlé [454].

The above discussion is for the Euclidean norm, but the Lagrange-Gauss reduction algorithm can be performed for any norm (the only change is how one computes μ). We refer to Kaib and Schnorr [325] for analysis and details.

Finally, we remark that there is a natural analogue of Definition 17.1.1 for any dimension. Hence, it is natural to try to generalise the Lagrange-Gauss algorithm to higher dimensions. Generalisations to dimension three have been given by Vallée [612] and Semaev [537]. There are a number of problems when generalising to higher dimensions. For example, choosing the right linear combination to size reduce $\underline{b}_n$ using $\underline{b}_1, \ldots, \underline{b}_{n-1}$ is solving the CVP in a sublattice (which is a hard problem). Furthermore, there is no guarantee that the resulting basis actually has good properties in high dimension. We refer to Nguyen and Stehlé [460] for a full discussion of these issues and an algorithm that works in dimensions 3 and 4.

17.1.1 Connection Between Lagrange-Gauss Reduction and Euclid's Algorithm

The Lagrange-Gauss algorithm is closely related to Euclid's algorithm. We briefly discuss some similarities and differences. Recall that if $a, b \in \mathbb{Z}$ then Euclid's algorithm (using signed remainders) produces a sequence of integers r_i, s_i, t_i such that

$$as_i + bt_i = r_i$$

where $|r_i t_i| < |a|$ and $|r_i s_i| < |b|$. The precise formulae are $r_{i+1} = r_{i-1} - qr_i$ and $s_{i+1} = s_{i-1} - qs_i$ where $q = \lfloor r_{i-1}/r_i \rceil$. The sequence $|r_i|$ is strictly decreasing. The initial values are $r_{-1} = a, r_0 = b, s_{-1} = 1, s_0 = 0, t_{-1} = 0, t_0 = 1$. In other words the lattice with basis matrix

$$B = \begin{pmatrix} 0 & b \\ 1 & a \end{pmatrix} = \begin{pmatrix} s_0 & r_0 \\ s_{-1} & r_{-1} \end{pmatrix}$$

contains the vectors

$$(s_i, r_i) = (t_i, s_i)B.$$

These vectors are typically shorter than the original vectors of the lattice.

We claim that if s_i is sufficiently small compared with r_i then one step of the Lagrange-Gauss algorithm on B corresponds to one step of Euclid's algorithm (with negative remainders).

To see this, let $\underline{b}_1 = (s_i, r_i)$ and consider the Lagrange-Gauss algorithm with $\underline{b}_2 = (s_{i-1}, r_{i-1})$. First compute the value

$$\mu = \frac{\langle \underline{b}_1, \underline{b}_2 \rangle}{\langle \underline{b}_1, \underline{b}_1 \rangle} = \frac{s_i s_{i-1} + r_i r_{i-1}}{s_i^2 + r_i^2}.$$

If s_i is sufficiently small relative to r_i (e.g., in the first step, when $s_0 = 0$) then

$$\lfloor \mu \rceil = \lfloor r_i r_{i-1} / r_i^2 \rceil = \lfloor r_{i-1}/r_i \rceil = q.$$

Hence the operation $\underline{v} = \underline{b}_2 - \lfloor \mu \rceil \underline{b}_1$ is $\underline{v} = (s_{i-1} - qs_i, r_{i-1} - qr_i)$, which agrees with Euclid's algorithm. Finally, the Lagrange-Gauss algorithm compares the lengths of the vectors $\underline{v}$ and $\underline{b}_1$ to see if they should be swapped. When s_{i+1} is small compared with r_{i+1} then $\|\underline{v}\|$ is smaller than $\|b_1\|$. Hence the vectors are swapped and the matrix becomes

$$\begin{pmatrix} s_{i-1} - qs_i & r_{i-1} - qr_i \\ s_i & r_i \end{pmatrix}.$$

just as in Euclid's algorithm.

The algorithms start to deviate once s_i become large (this can already happen on the second iteration, as the below example shows). Further, Euclid's algorithm runs until $r_i = 0$ (in which case $s_i \approx b$) whereas Lagrange-Gauss reduction stops when $r_i \approx s_i$.

Example 17.1.11. Let $a = 19$ and $b = 8$. The sequence of remainders in the signed Euclidean algorithm is $3, -1$ while the Lagrange-Gauss lattice basis reduction algorithm computes remainders $3, 2$.

Example 17.1.12. Consider $a = 8239876$ and $b = 1020301$, which have gcd equal to one. Let

$$B = \begin{pmatrix} 0 & b \\ 1 & a \end{pmatrix}.$$

Running the Lagrange-Gauss algorithm on this matrix gives

$$\begin{pmatrix} 540 & 379 \\ 619 & -1455 \end{pmatrix}.$$

One can verify that

$$379 = 540a + t_4 b \text{ where } t_4 = -4361$$

and

$$-1455 = 619a + t_5 b \text{ where } t_5 = -4999.$$

17.2 LLL-Reduced Lattice Bases

This section presents the crucial definition from [372] and some of its consequences. The main result is Theorem 17.2.12, which shows that an LLL-reduced lattice basis does have good properties.

Recall first that if $\underline{b}_1, \ldots, \underline{b}_n$ is a set of vectors in $\mathbb{R}^m$ then one can define the Gram-Schmidt orthogonalisation $\underline{b}_1^*, \ldots, \underline{b}_n^*$ as in Section A.10.2. We use the notation $\mu_{i,j} = \langle \underline{b}_i, \underline{b}_j^* \rangle / \langle \underline{b}_j^*, \underline{b}_j^* \rangle$ throughout.

As we have noted in Example 16.3.3, computational problems in lattices can be easy if one has a basis that is orthogonal, or "sufficiently close to orthogonal". A simple but important observation is that one can determine when a basis is close to orthogonal by considering the lengths of the Gram-Schmidt vectors. More precisely, a lattice basis is "close to orthogonal" if the lengths of the Gram-Schmidt vectors do not decrease too rapidly.

Example 17.2.1. Two bases for $\mathbb{Z}^2$ are $\{(1,0),(0,1)\}$ and $\{(23,24),(24,25)\}$. In the first case, the Gram-Schmidt vectors both have length 1. In the second case the Gram-Schmidt vectors are $\underline{b}_1^* = (23,24)$ and $\underline{b}_2^* = (24/1105, -23/1105)$, which have lengths $\sqrt{1105} \approx 33.24$ and $1/\sqrt{1105} \approx 0.03$ respectively. The fact that the lengths of the Gram-Schmidt vectors dramatically decrease reveals that the original basis is not of good quality.

We now list some easy properties of the Gram-Schmidt orthogonalisation.

Lemma 17.2.2. *Let $\{\underline{b}_1, \ldots, \underline{b}_n\}$ be linearly independent in $\mathbb{R}^m$ and let $\{\underline{b}_1^*, \ldots, \underline{b}_n^*\}$ be the Gram-Schmidt orthogonalisation.*

1. *$\|\underline{b}_i^*\| \leq \|\underline{b}_i\|$ for $1 \leq i \leq n$.*
2. *$\langle \underline{b}_i, \underline{b}_i^* \rangle = \langle \underline{b}_i^*, \underline{b}_i^* \rangle$ for $1 \leq i \leq n$.*
3. *Denote the closest integer to $\mu_{k,j}$ by $\lfloor \mu_{k,j} \rceil$. If $\underline{b}'_k = \underline{b}_k - \lfloor \mu_{k,j} \rceil \underline{b}_j$ for $1 \leq k \leq n$ and $1 \leq j < k$ and if $\mu'_{k,j} = \langle \underline{b}'_k, \underline{b}_j^* \rangle / \langle \underline{b}_j^*, \underline{b}_j^* \rangle$ then $|\mu'_{k,j}| \leq 1/2$.*

Exercise 17.2.3. Prove Lemma 17.2.2.

Definition 17.2.4. Let $\{\underline{b}_1, \ldots, \underline{b}_n\}$ be an ordered basis for a lattice. Denote by $\{\underline{b}_1^*, \ldots, \underline{b}_n^*\}$ the Gram-Schmidt orthogonalisation and write $B_i = \|\underline{b}_i^*\|^2 = \langle \underline{b}_i^*, \underline{b}_i^* \rangle$. Let

$$\mu_{i,j} = \langle \underline{b}_i, \underline{b}_j^* \rangle / \langle \underline{b}_j^*, \underline{b}_j^* \rangle$$

for $1 \leq j < i \leq n$ be the coefficients from the Gram-Schmidt process. Fix $1/4 < \delta < 1$. The (ordered) basis is **LLL reduced** (with factor δ) if the following conditions hold:

- (Size reduced) $|\mu_{i,j}| \leq 1/2$ for $1 \leq j < i \leq n$.
- (Lovász condition)
$$B_i \geq \big(\delta - \mu_{i,i-1}^2\big) B_{i-1}$$
for $2 \leq i \leq n$.

It is traditional to choose $\delta = 3/4$ in the Lovász condition.

Exercise 17.2.5. Which of the following basis matrices represents an LLL reduced basis (with $\delta = 3/4$)?

$$\begin{pmatrix} 1 & 0 \\ 0 & 2 \end{pmatrix}, \begin{pmatrix} 0 & 4 \\ 1 & 0 \end{pmatrix}, \begin{pmatrix} 1 & -2 \\ 3 & 1 \end{pmatrix}, \begin{pmatrix} 5 & 0 \\ 0 & 4 \end{pmatrix}, \begin{pmatrix} 10 & 0 \\ 0 & 9 \end{pmatrix}.$$

Exercise 17.2.6. Prove that an equivalent formulation (more in the flavour of the Lagrange-Gauss method) of the Lovász condition is

$$B_i + \mu_{i,i-1}^2 B_{i-1} = \|\underline{b}_i^* + \mu_{i,i-1} \underline{b}_{i-1}^*\|^2 \geq \delta B_{i-1}.$$

Exercise 17.2.7. Find an ordered basis $\{\underline{b}_1, \underline{b}_2\}$ in $\mathbb{R}^2$ that is LLL-reduced, but has the property that $\|\underline{b}_2\| < \|\underline{b}_1\|$ and that the ordered basis $\{\underline{b}_2, \underline{b}_1\}$ is not LLL-reduced.

For the moment we do not concern ourselves with the question of whether an LLL reduced basis can exist for every lattice L. In Section 17.4 we will present the LLL algorithm, which constructs such a basis for any lattice (hence giving a constructive existence proof for an LLL reduced basis).

The following properties of an LLL reduced basis hold.

Lemma 17.2.8. *Let $\{\underline{b}_1, \ldots, \underline{b}_n\}$ be an LLL reduced basis with $\delta = 3/4$ for a lattice $L \subset \mathbb{R}^m$. Let the notation be as above. In particular, $\|\underline{b}\|$ is the Euclidean norm.*

1. *$B_j \leq 2^{i-j} B_i$ for $1 \leq j \leq i \leq n$.*
2. *$B_i \leq \|\underline{b}_i\|^2 \leq (\frac{1}{2} + 2^{i-2}) B_i$ for $1 \leq i \leq n$.*

3. $\|\underline{b}_j\| \leq 2^{(i-1)/2}\|\underline{b}_i^*\|$ *for* $1 \leq j \leq i \leq n$.

Proof:

1. The Lovász condition implies $B_i \geq (\frac{3}{4} - \frac{1}{4})B_{i-1} = \frac{1}{2}B_{i-1}$ and the result follows by induction.

2. From $\underline{b}_i = \underline{b}_i^* + \sum_{j=1}^{i-1} \mu_{i,j}\underline{b}_j^*$ we have

$$\begin{aligned} \|\underline{b}_i\|^2 &= \langle \underline{b}_i, \underline{b}_i \rangle \\ &= \left\langle \underline{b}_i^* + \sum_{j=1}^{i-1} \mu_{i,j}\underline{b}_j^* \ , \ \underline{b}_i^* + \sum_{j=1}^{i-1} \mu_{i,j}\underline{b}_j^* \right\rangle \\ &= B_i + \sum_{j=1}^{i-1} \mu_{i,j}^2 B_j, \end{aligned}$$

   which is clearly $\geq B_i$. By part 1 this is at most $B_i(1+\frac{1}{4}\sum_{j=1}^{i-1} 2^{i-j}) = B_i(1+\frac{1}{4}(2^i - 2)) = B_i(\frac{1}{2} + 2^{i-2})$.

3. Since $j \geq 1$ we have $\frac{1}{2} + 2^{j-2} \leq 2^{j-1}$. Part 2 can therefore be written as $\|b_j\|^2 \leq 2^{j-1}B_j$. By part 1, $B_j \leq 2^{i-j}B_i$ and so $\|b_j\|^2 \leq 2^{j-1}2^{i-j}B_i = 2^{i-1}B_i$. Taking square roots gives the result.

□

We now give the same result for a slightly different value of δ.

Lemma 17.2.9. *Let* $\{\underline{b}_1, \ldots, \underline{b}_n\}$ *be an LLL reduced basis with* $\delta = 1/4 + 1/\sqrt{2} \approx 0.957$ *for a lattice* $L \subset \mathbb{R}^m$. *Let the notation be as above. In particular,* $\|\underline{b}\|$ *is the Euclidean norm.*

1. $B_j \leq 2^{(i-j)/2}B_i$ *for* $1 \leq j \leq i \leq n$.
2. $B_i \leq \|\underline{b}_i\|^2 \leq (\frac{1}{6} + 2^{(i-1)/2})B_i$ *for* $1 \leq i \leq n$.
3. $\|\underline{b}_j\| \leq 2^{i/4}\|\underline{b}_i^*\|$ *for* $1 \leq j \leq i \leq n$.

Exercise 17.2.10.★ Prove Lemma 17.2.9.

Lemma 17.2.11. *Let* $\{\underline{b}_1, \ldots, \underline{b}_n\}$ *be an ordered basis for a lattice* $L \subset \mathbb{R}^m$ *and let* $\{\underline{b}_1^*, \ldots, \underline{b}_n^*\}$ *be the Gram-Schmidt orthogonalisation. Let* λ_1 *be the length of the shortest non-zero vector in the lattice. Then*

$$\lambda_1 \geq \min_{1 \leq i \leq n} \|\underline{b}_i^*\|.$$

Furthermore, let $\underline{w}_1, \ldots, \underline{w}_i \in L$ *be linearly independent lattice vectors such that* $\max\{\|\underline{w}_1\|, \ldots, \|\underline{w}_i\|\} = \lambda_i$, *as in the definition of successive minima. Write* $\underline{w}_j = \sum_{k=1}^{n} z_{j,k}\underline{b}_k$. *For* $1 \leq j \leq i$ *denote by* $k(j)$ *the largest value for* k *such that* $1 \leq k \leq n$ *and* $z_{j,k} \neq 0$. *Then* $\|\underline{w}_j\| \geq \|\underline{b}_{k(j)}^*\|$.

Proof: Let $\underline{x} = (x_1, \ldots, x_n) \in \mathbb{Z}^n$ be arbitrary such that $\underline{x} \neq \underline{0}$. Let i be the largest index such that $x_i \neq 0$. We will show that $\|\underline{x}B\| \geq \|b_i^*\|$, from which the result follows.

We have $\underline{x}B = \sum_{j=1}^{i} x_j\underline{b}_j$. Since $\underline{b}_i^*$ is orthogonal to the span of $\{\underline{b}_1, \ldots, \underline{b}_{i-1}\}$ we have $\langle \underline{x}B, \underline{b}_i^* \rangle = x_i\langle \underline{b}_i^*, \underline{b}_i^* \rangle = x_i\|\underline{b}_i^*\|^2$. Since $x_i \in \mathbb{Z}$ and $x_i \neq 0$ it follows that $|\langle \underline{x}B, \underline{b}_i^* \rangle| \geq \|\underline{b}_i^*\|^2$. By part 4 of Lemma A.10.3 it follows that

$$\|\underline{x}B\| \geq \|\underline{b}_i^*\|,$$

which completes the proof. □

Theorem 17.2.12 shows that an LLL reduced lattice basis has good properties. In particular, the first vector of an LLL-reduced lattice basis has length at most $2^{(n-1)/2}$ times the length of a shortest non-zero vector.

Theorem 17.2.12. *Let $\{\underline{b}_1, \ldots, \underline{b}_n\}$ be an LLL reduced basis with $\delta = 3/4$ for a lattice $L \subset \mathbb{R}^m$. Let the notation be as above. In particular, $\|\underline{b}\|$ is the Euclidean norm.*

1. $\|\underline{b}_1\| \leq 2^{(n-1)/2}\lambda_1$.
2. $\|\underline{b}_j\| \leq 2^{(n-1)/2}\lambda_i$ *for* $1 \leq j \leq i \leq n$. *(This may look strange, but it tends to be used for fixed i and varying j, rather than the other way around.)*
3. $2^{(1-i)/2}\lambda_i \leq \|\underline{b}_i\| \leq 2^{(n-1)/2}\lambda_i$.
4. $\det(L) \leq \prod_{i=1}^{n} \|\underline{b}_i\| \leq 2^{n(n-1)/4} \det(L)$.
5. $\|\underline{b}_1\| \leq 2^{(n-1)/4} \det(L)^{1/n}$.

Proof:

1. From part 1 of Lemma 17.2.8 we have $\|b_i^*\| \geq 2^{(i-1)/2}\|b_1^*\|$. Hence, part 1 implies

$$\begin{aligned} \lambda_1 &\geq \min_{1 \leq i \leq n} \|\underline{b}_i^*\| \\ &\geq \min_{1 \leq i \leq n} 2^{(1-i)/2}\|\underline{b}_1^*\| \\ &= 2^{(1-n)/2}\|\underline{b}_1^*\|. \end{aligned}$$

   The result follows since $\underline{b}_1^* = \underline{b}_1$.

2. Let $\underline{w}_1, \ldots, \underline{w}_i \in L$ be linearly independent lattice vectors such that $\max\{\|\underline{w}_1\|, \ldots, \|\underline{w}_i\|\} = \lambda_i$. Let $k(j)$ be defined as in Lemma 17.2.11 so that $\|\underline{w}_j\| \geq \|\underline{b}_{k(j)}^*\|$.

   Renumber the vectors $\underline{w}_j$ so that $k(1) \leq k(2) \leq \cdots \leq k(i)$. We claim that $j \leq k(j)$. If not then $\underline{w}_1, \ldots, \underline{w}_j$ would belong to the span of $\{\underline{b}_1, \ldots, \underline{b}_{j-1}\}$ and would be linearly dependent.

   Finally,

$$\|\underline{b}_j\| \leq 2^{(k(j)-1)/2}\|\underline{b}_{k(j)}^*\| \leq 2^{(n-1)/2}\|\underline{w}_j\| \leq 2^{(n-1)/2}\lambda_i,$$

   which proves the result.

3. The upper bound on $\|\underline{b}_i\|$ is given by part 2.

   Since $\{\underline{b}_1, \ldots, \underline{b}_i\}$ are linearly independent we have $\lambda_i \leq \max_{1 \leq j \leq i} \|\underline{b}_j\|$ and by part 3 of Lemma 17.2.8 each $\|\underline{b}_j\| \leq 2^{(i-1)/2}\|\underline{b}_i^*\|$. Using $\|\underline{b}_i^*\| \leq \|\underline{b}_i\|$ we obtain the lower bound on $\|\underline{b}_i\|$.

4. By Lemma 16.1.14 we have $\det(L) = \prod_{i=1}^{n} \|\underline{b}_i^*\|$. The result follows from $\|\underline{b}_i^*\| \leq \|\underline{b}_i\| \leq 2^{(i-1)/2}\|\underline{b}_i^*\|$.

5. By part 3 of Lemma 17.2.8 we have $\|\underline{b}_1\| \leq 2^{(i-1)/2}\|\underline{b}_i^*\|$ and so

$$\|\underline{b}_1\|^n \leq \prod_{i=1}^{n} 2^{(i-1)/2}\|\underline{b}_i^*\| = 2^{n(n-1)/4} \det(L).$$

□

Corollary 17.2.13. *If* $\|\underline{b}_1\| \leq \|\underline{b}_i^*\|$ *for all* $1 \leq i \leq n$ *then* $\underline{b}_1$ *is a correct solution to SVP.*

Exercise 17.2.14. Prove Corollary 17.2.13.

Exercise 17.2.15. Suppose L is a lattice in $\mathbb{Z}^m$ and let $\{\underline{b}_1, \ldots, \underline{b}_n\}$ be an LLL-reduced basis. Rename these vectors as $\underline{v}_1, \ldots, \underline{v}_n$ such that $1 \leq \|\underline{v}_1\| \leq \|\underline{v}_2\| \leq \cdots \leq \|\underline{v}_n\|$. Show that one does not necessarily have $\|\underline{v}_1\| = \|\underline{b}_1\|$. Show that, for $1 \leq i \leq n$,

$$\|\underline{v}_i\| \leq \left(2^{n(n-1)/4} \det(L)\right)^{1/(n+1-i)}.$$

As a final remark, the results in this section have only given upper bounds on the sizes of $\|\underline{b}_i\|$ in an LLL-reduced lattice basis. In many practical instances, one finds that LLL-reduced lattice vectors are much shorter than these bounds might suggest.

17.3 The Gram-Schmidt Algorithm

The LLL algorithm requires computing a Gram-Schmidt basis. For the complexity analysis of the LLL algorithm it is necessary to give a more careful description and analysis of the Gram-Schmidt algorithm than was done in Section A.10.2. We present pseudocode in Algorithm 24 (the "downto" in line 4 is not necessary, but we write it that way for future reference in the LLL algorithm).

Algorithm 24 Gram-Schmidt algorithm

INPUT: $\{\underline{b}_1, \ldots, \underline{b}_n\}$ in $\mathbb{R}^m$
OUTPUT: $\{\underline{b}_1^*, \ldots, \underline{b}_n^*\}$ in $\mathbb{R}^m$
1: $\underline{b}_1^* = \underline{b}_1$
2: **for** $i = 2$ to n **do**
3: $\quad \underline{v} = \underline{b}_i$
4: $\quad$ **for** $j := i - 1$ downto 1 **do**
5: $\quad\quad \mu_{i,j} = \langle \underline{b}_i, \underline{b}_j^* \rangle / \langle \underline{b}_j^*, \underline{b}_j^* \rangle$
6: $\quad\quad \underline{v} = \underline{v} - \mu_{i,j} \underline{b}_j^*$
7: $\quad$ **end for**
8: $\quad \underline{b}_i^* = \underline{v}$
9: **end for**
10: **return** $\{\underline{b}_1^*, \ldots, \underline{b}_n^*\}$

When working in $\mathbb{R}$ the standard way to implement this algorithm is using floating-point arithmetic. However, problems can arise (especially if the $\underline{b}_i^*$ decrease quickly in size). Such issues are beyond the scope of this book; we refer to Higham [284] for details.

If the input vectors lie in $\mathbb{Z}^m$ then one can perform Algorithm 24 using exact arithmetic over $\mathbb{Q}$. However, the integers can become very large (this is called **coefficient explosion**). We now analyse the size of the integers and prove the complexity of the exact version of the Gram-Schmidt algorithm. These results are used later when determining the complexity of LLL.

Definition 17.3.1. Let $\underline{b}_1, \ldots, \underline{b}_n$ be an ordered set of vectors in $\mathbb{Z}^m$. Define $B_i = \|\underline{b}_i^*\|^2$ (as before). For $1 \leq i \leq n-1$ define the $i \times m$ matrix $B_{(i)}$ whose rows are $\underline{b}_1, \ldots, \underline{b}_i$. Define $d_0 = 0$ and, for $1 \leq i \leq n$,

$$d_i = \det(B_{(i)} B_{(i)}^T) \;=\; \det(\langle \underline{b}_j, \underline{b}_k \rangle_{1 \leq j,k \leq i}) \;\in\; \mathbb{Z},$$

which is the square of the volume of the sublattice generated by $B_{(i)}$.

Lemma 17.3.2. *Let the notation be as above.*

1. $d_i = \prod_{j=1}^{i} B_j$ *for* $1 \leq i \leq n$.
2. $B_i = d_i/d_{i-1}$ *for* $1 \leq i \leq n$.
3. $d_{i-1}\underline{b}_i^* \in L \subseteq \mathbb{Z}^n$ *for* $1 \leq i \leq n$, *where* L *is the lattice spanned by* $\{\underline{b}_1, \ldots, \underline{b}_n\}$.
4. $d_j\mu_{i,j} \in \mathbb{Z}$ *for* $1 \leq j < i \leq n$.

Proof:

1. Write $L_{(i)}$ for the lattice spanned by the first i vectors (i.e., L is given by the matrix $B_{(i)}$). Then $d_i = \det(L_{(i)})^2 = \prod_{j=1}^{i} \|\underline{b}_j^*\|^2 = \prod_{j=1}^{i} B_j$ by Lemma 16.1.14.
2. This property follows immediately from the previous one.
3. Write $\underline{b}_i^* = \underline{b}_i - \sum_{j=1}^{i-1} a_{i,j}\underline{b}_j$ for some $a_{i,j} \in \mathbb{R}$. Note that the sum is over vectors $\underline{b}_j$ not $\underline{b}_j^*$, so the $a_{i,j}$ are not the same as the $\mu_{i,j}$. Since $\langle \underline{b}_l, \underline{b}_i^* \rangle = 0$ for $1 \leq l < i$ we have
$$\langle \underline{b}_l, \underline{b}_i \rangle = \sum_{j=1}^{i-1} a_{i,j} \langle \underline{b}_l, \underline{b}_j \rangle,$$
which corresponds to the matrix product
$$(\langle \underline{b}_i, \underline{b}_1 \rangle, \ldots, \langle \underline{b}_i, \underline{b}_{i-1} \rangle) = (a_{i,1}, \ldots, a_{i,i-1}) B_{(i-1)} B_{(i-1)}^T.$$
Inverting $B_{(i-1)}B_{(i-1)}^T$ to solve for the $a_{i,j}$ gives $d_{i-1}a_{i,j} \in \mathbb{Z}$. It follows that $d_{i-1}\underline{b}_i^* \in L \subset \mathbb{Z}^n$ as required.
4. By the previous results we have $d_j\mu_{i,j} = d_{j-1}B_j\langle \underline{b}_i, \underline{b}_j^* \rangle / B_j = \langle \underline{b}_i, d_{j-1}\underline{b}_j^* \rangle \in \mathbb{Z}$.

□

Exercise 17.3.3. Consider the vector $\underline{v} = \underline{b}_i - \sum_{k=j}^{i-1} \mu_{i,k}\underline{b}_k^*$ in line 6 of Algorithm 24 during iteration j. Show that

$$\|\underline{v}\|^2 = \|\underline{b}_i\|^2 - \sum_{k=j}^{i-1} \mu_{i,k}^2 \|\underline{b}_k^*\|^2.$$

Deduce that $\|\underline{v}\| \leq \|\underline{b}_i\|$ and that $d_{i-1}\underline{v} \in \mathbb{Z}^m$ throughout the loop in line 4 of the algorithm.

Theorem 17.3.4. *Let* $\underline{b}_1, \ldots, \underline{b}_n$ *be vectors in* $\mathbb{Z}^m$. *Let* $X \in \mathbb{Z}_{\geq 2}$ *be such that* $\|\underline{b}_i\|^2 \leq X$ *for* $1 \leq i \leq n$. *Then the Gram-Schmidt algorithm performs* $O(n^4 m \log(X)^2)$ *bit operations. The output size is* $O(n^2 m \log(X))$.

Proof: One runs Algorithm 24 using exact $\mathbb{Q}$ arithmetic for the vectors $\underline{b}_i^*$. Lemma 17.3.2 shows that the denominators in $\underline{b}_i^*$ are all factors of d_{i-1}, which has size $\prod_{j=1}^{i-1} B_j \leq \prod_{j=1}^{i-1} \|\underline{b}_j\|^2 \leq X^{i-1}$. Also, $\|\underline{b}_i^*\| \leq \|\underline{b}_i\| \leq X$, so the numerators are bounded by X^i. The size of each vector $\underline{b}_i^*$ and, by Exercise 17.3.3, the intermediate steps $\underline{v}$ in the computation are therefore $O(mi \log(X))$ bits, which gives the output size of the algorithm. The computation $\langle \underline{b}_i, \underline{b}_j^* \rangle$ requires $O(mn \log(X)^2)$ bit operations and the computation $\langle \underline{b}_j^*, \underline{b}_j^* \rangle$ requires $O(mn^2 \log(X)^2)$ bit operations. As there are $O(n^2)$ vector operations to perform, one gets the stated running time. □

Corollary 17.3.5. *Let the notation be as in Theorem 17.3.4 and let L be the lattice in $\mathbb{Z}^m$ with basis $\{\underline{b}_1, \ldots, \underline{b}_n\}$. Then one can compute $\det(L)^2$ in $O(n^4 m \log(X)^2)$ bit operations.*[4]

Proof: Lemma 16.1.14 implies $\det(L)^2 = \prod_{i=1}^{n} \|\underline{b}_i^*\|^2$. One computes $\underline{b}_i^*$ using exact (naive) arithmetic over $\mathbb{Q}$ in $O(n^4 m \log(X)^2)$ bit operations. One computes each $\|\underline{b}_i^*\|^2 \in \mathbb{Q}$ in $O(mn^2 \log(X)^2)$ bit operations. Since $\|\underline{b}_i^*\|^2 \leq X$ and $d_{i-1}\|\underline{b}_i^*\|^2 \in \mathbb{Z}$ it follows that $\|\underline{b}_i^*\|^2$ is a ratio of integers bounded by X^n. One computes the product of the $\|\underline{b}_i^*\|^2$ in $O(n^3 \log(X)^2)$ bit operations (since the integers in the product are bounded by X^{n^2}). Finally, one can reduce the fraction using Euclid's algorithm and division in $O(n^4 \log(X)^2)$ bit operations. □

17.4 The LLL Algorithm

The Lenstra-Lenstra-Lovász (LLL) algorithm is an iterative algorithm that transforms a given lattice basis into an LLL-reduced one. Since the definition of LLL-reduced uses Gram-Schmidt vectors, the algorithm performs the Gram-Schmidt method as a subroutine. The first condition of Definition 17.2.4 is easily met by taking suitable integer linear combinations. If the second condition is not met then $\underline{b}_i$ is not significantly longer than $\underline{b}_{i-1}$. In this case we swap $\underline{b}_i$ and $\underline{b}_{i-1}$ and backtrack. The swapping of vectors is familiar from the Lagrange-Gauss 2-dimensional lattice basis reduction algorithm and also Euclid's algorithm. We give the precise details in Algorithm 25.

Algorithm 25 LLL algorithm with Euclidean norm (typically, choose $\delta = 3/4$)

INPUT: $\underline{b}_1, \ldots, \underline{b}_n \in \mathbb{Z}^m$.
OUTPUT: LLL reduced basis $\underline{b}_1, \ldots, \underline{b}_n$

Compute the Gram-Schmidt basis $\underline{b}_1^*, \ldots, \underline{b}_n^*$ and coefficients $\mu_{i,j}$ for $1 \leq j < i \leq n$
Compute $B_i = \langle \underline{b}_i^*, \underline{b}_i^* \rangle = \|\underline{b}_i^*\|^2$ for $1 \leq i \leq n$
$k = 2$
while $k \leq n$ **do**
    for $j = (k-1)$ downto 1 **do** ▷ Perform size reduction
        Let $q_j = \lfloor \mu_{k,j} \rceil$ and set $\underline{b}_k = \underline{b}_k - q_j \underline{b}_j$
        Update the values $\mu_{k,j}$ for $1 \leq j < k$
    end for
    if $B_k \geq (\delta - \mu_{k,k-1}^2) B_{k-1}$ **then** ▷ Check Lovász condition
        $k = k + 1$
    else
        Swap $\underline{b}_k$ with $\underline{b}_{k-1}$
        Update the values $\underline{b}_k^*$, $\underline{b}_{k-1}^*$, B_k, B_{k-1}, $\mu_{k-1,j}$ and $\mu_{k,j}$ for $1 \leq j < k$, and $\mu_{i,k}, \mu_{i,k-1}$ for $k < i \leq n$
        $k = \max\{2, k-1\}$
    end if
end while

Lemma 17.4.1. *Throughout the LLL algorithm the values $\underline{b}_i^*$ and B_i for $1 \leq i \leq n$ and $\mu_{i,j}$ for $1 \leq j < i \leq n$ are all correct Gram-Schmidt values.*

[4]Since $\det(L)^2 \in \mathbb{Z}$ while $\det(L)$ may not be rational if $n < m$, we prefer to work with $\det(L)^2$.

Exercise 17.4.2. Prove Lemma 17.4.1. In other words, show that line 6 of the LLL algorithm does not change $\underline{b}_i^*$ or B_i for $1 \leq i \leq n$. Similarly, line 12 of the algorithm does not change any values except those mentioned in line 13.

It is illuminating to compare the LLL algorithm with the Lagrange-Gauss reduction algorithm. The basic concept of size reduction followed by a swap is the same, however there are two crucial differences.

1. The size reduction operation in the Lagrange-Gauss algorithm gives the minimal value for $\|\underline{b}_2 + q\underline{b}_1\|$ over $q \in \mathbb{Z}$. In LLL the coefficient $\mu_{k,j}$ is chosen to depend on $\underline{b}_k$ and $\underline{b}_j^*$ so it does not necessarily minimise $\|\underline{b}_k\|$. Indeed $\|\underline{b}_k\|$ can grow during the algorithm. Of course, in the two-dimensional case of LLL then $\mu_{2,1}$ is the same as the value used in the Lagrange-Gauss algorithm and so the size reduction step is the same.

2. The size check in LLL (the Lovász condition) is on the lengths of the Gram-Schmidt vectors, unlike the size check in the Lagrange-Gauss algorithm, which is on the length of the basis vectors themselves.

These features of LLL may seem counterintuitive, but they are essential to the proof that the algorithm runs in polynomial-time.

Lemma 17.4.3. *If $\underline{b}_k$ and $\underline{b}_{k-1}$ are swapped then the Gram-Schmidt vectors $\underline{b}_i^*$ for $1 \leq i \leq n$ are changed as follows*

1. *For $1 \leq i < k-1$ and $k < i < n$ then $\underline{b}_i^*$ is unchanged.*

2. *The new value for $\underline{b}_{k-1}^*$ is $\underline{b}_k^* + \mu_{k,k-1}\underline{b}_{k-1}^*$ and the new value for B_{k-1} is $B'_k = B_k + \mu_{k,k-1}^2 B_{k-1}$.*

3. *The new value for $\underline{b}_k^*$ is $(B_k/B'_{k-1})\underline{b}_{k-1}^* - (\mu_{k,k-1}B_{k-1}/B'_{k-1})\underline{b}_k^*$ and the new value for B_k is $B_{k-1}B_k/B'_{k-1}$.*

Proof: Denote by $\underline{b}'_i$ the new basis (i.e., $\underline{b}'_{k-1} = \underline{b}_k$ and $\underline{b}'_k = \underline{b}_{k-1}$), $\underline{b}_i'^*$ and $\mu'_{i,j}$ the new Gram-Schmidt values and B'_i the squares of the lengths of the $\underline{b}_i'^*$. Clearly $\underline{b}_i'^* = \underline{b}_i^*$ for $1 \leq i < k-1$ and $\mu'_{i,j} = \mu_{i,j}$ for $1 \leq j < i < k-1$. Now

$$\begin{aligned}
\underline{b}_{k-1}'^* &= \underline{b}'_{k-1} - \sum_{j=1}^{k-2} \mu'_{k-1,j}\underline{b}_j'^* \\
&= \underline{b}_k - \sum_{j=1}^{k-2} \mu_{k,j}\underline{b}_j^* \\
&= \underline{b}_k^* + \mu_{k,k-1}\underline{b}_{k-1}^*.
\end{aligned}$$

Hence, $B'_{k-1} = B_k + \mu_{k,k-1}^2 B_{k-1}$.

Similarly,

$$\begin{aligned}
\underline{b'_k}^* &= \underline{b}'_k - \sum_{j=1}^{k-1} \mu'_{k,j} \underline{b'_j}^* \\
&= \underline{b}_{k-1} - \sum_{j=1}^{k-2} \mu_{k-1,j} \underline{b}^*_j \quad - \quad \left(\langle \underline{b}_{k-1}, \underline{b'}^*_{k-1} \rangle / B'_{k-1}\right) \underline{b'}^*_{k-1} \\
&= \underline{b}^*_{k-1} - \left(\langle \underline{b}_{k-1}, \underline{b}^*_k + \mu_{k,k-1} \underline{b}^*_{k-1} \rangle / B'_{k-1}\right) (\underline{b}^*_k + \mu_{k,k-1} \underline{b}^*_{k-1}) \\
&= \underline{b}^*_{k-1} - \left(\mu_{k,k-1} B_{k-1} / B'_{k-1}\right) (\underline{b}^*_k + \mu_{k,k-1} \underline{b}^*_{k-1}) \\
&= \left(1 - \mu^2_{k,k-1} B_{k-1} / B'_{k-1}\right) \underline{b}^*_{k-1} - \left(\mu_{k,k-1} B_{k-1} / B'_{k-1}\right) \underline{b}^*_k.
\end{aligned}$$

The result for $\underline{b'_k}^*$ follows since $1 - \mu^2_{k,k-1} B_{k-1}/B'_{k-1} = B_k / B'_{k-1}$. Finally,

$$B'_k = (B^2_k \langle \underline{b}^*_{k-1}, \underline{b}^*_{k-1} \rangle / {B'_{k-1}}^2 + \mu^2_{k,k-1} B^2_{k-1} \langle \underline{b}^*_k, \underline{b}^*_k \rangle / {B'_{k-1}}^2 = B_{k-1} B_k / B'_{k-1}.$$

□

Exercise 17.4.4. Give explicit formulae for updating the other Gram-Schmidt values in lines 7 and 13 of Algorithm 25.

Exercise 17.4.5. Show that it is not necessary to store or update the values $\underline{b}^*_i$ for $1 \leq i \leq n$ in the LLL algorithm once the values B_i have been computed.

Exercise 17.4.6. Show that the condition in line 9 of Algorithm 25 can be checked immediately after $\mu_{k,k-1}$ has been computed. Hence, show that the cases $1 \leq j < k-1$ in the loop in lines 5 to 8 of Algorithm 25 can be postponed to line 10.

Lemma 17.4.7. *If the LLL algorithm terminates then the output basis is LLL reduced.*

Exercise 17.4.8. Prove Lemma 17.4.7. Indeed, the fact that the Lovász conditions are satisfied is immediate. Prove the bounds on the $\mu_{i,j}$ using the three following steps. Let $1 \leq j < k$ and let $b'_k = b_k - \lfloor \mu_{k,j} \rceil b_j$.

1. Prove that $\langle b_j, b^*_j \rangle = \langle b^*_j, b^*_j \rangle$ and $\langle b_j, b^*_i \rangle = 0$ if $j < i$.
2. Hence, writing $\mu'_{k,j} = \langle b'_k, b^*_j \rangle / \langle b^*_j, b^*_j \rangle$, prove that $|\mu'_{k,j}| \leq 1/2$ for $1 \leq j < k$.
3. For $j < i < k$ denote $\mu'_{k,i} = \langle b'_k, b^*_i \rangle / \langle b^*_i, b^*_i \rangle$. Prove that $\mu'_{k,i} = \mu_{k,i}$.

In the next section we show that the LLL algorithm does terminate. Before then we give an example and some further discussion.

Example 17.4.9. Let L be the lattice with basis matrix

$$B = \begin{pmatrix} 1 & 0 & 0 \\ 4 & 2 & 15 \\ 0 & 0 & 3 \end{pmatrix}.$$

We will perform the LLL algorithm to reduce this lattice basis.

We start with $k = 2$ and compute $\mu_{2,1} = 4/1 = 4$. So $q_1 = 4$ and we define

$$\underline{b}_2 = \underline{b}_2 - 4\underline{b}_1 = (4, 2, 15) - (4, 0, 0) = (0, 2, 15).$$

We now want to check the second LLL condition. To do this we need $\underline{b}^*_2$. We compute $\mu_{2,1} = 0$ and hence $\underline{b}^*_2 = \underline{b}_2$. Then $B_1 = 1$ and $B_2 = \langle \underline{b}^*_2, \underline{b}^*_2 \rangle = 229$. Clearly, $B_2 >$

$(3/4 - \mu_{2,1}^2)B_1$ and so we set $k = 3$. Now consider $\underline{b}_3$. We compute $\mu_{3,2} = 45/229 \approx 0.19$ and, since $q_2 = 0$ there is no reduction to be performed on $\underline{b}_3$. We compute $\mu_{3,1} = 0$, so again no size reduction is required. We now compute

$$\underline{b}_3^* = \underline{b}_3 - \tfrac{45}{229}\underline{b}_2^* = (0, -90/229, 12/229).$$

We have $B_2 = 229$ and $B_3 = \langle \underline{b}_3^*, \underline{b}_3^* \rangle = 8244/52441 \approx 0.157$. From this one can check that $B_3 < (3/4 - \mu_{3,2}^2)B_2 \approx 166.1$. Hence we swap $\underline{b}_2$ and $\underline{b}_3$ and set $k = 2$.

At this point we have the vectors

$$\underline{b}_1 = (1, 0, 0) \text{ and } \underline{b}_2 = (0, 0, 3)$$

and $\underline{b}_1^* = \underline{b}_1$, $\underline{b}_2^* = \underline{b}_2$. First check that $\mu_{2,1} = 0$ and so no size reduction on $\underline{b}_2$ is required. Second, $B_1 = 1$ and $B_2 = 9$ and one checks that $B_2 > (3/4 - \mu_{2,1}^2)B_1 = 0.75$. Hence we set $k = 3$. Now

$$\underline{b}_3 = (0, 2, 15)$$

and we compute $\mu_{3,2} = 45/9 = 5$. Hence we reduce

$$\underline{b}_3 = \underline{b}_3 - 5\underline{b}_2 = (0, 2, 0).$$

Now compute $\mu_{3,1} = 0$, so no reduction is required.

One computes $\mu_{3,2} = 0$, $\underline{b}_3^* = \underline{b}_3$ and $B_3 = 4$. Hence, $B_3 < (3/4 - \mu_{3,2}^2)B_2 = 27/4 = 6.75$ and so we should swap $\underline{b}_2$ and $\underline{b}_3$ and set $k = 2$. One can check that the $k = 2$ phase runs without making any changes. We have $B_1 = 1$ and $B_2 = 4$. Consider now $k = 3$ again. We have $\mu_{3,2} = \mu_{3,1} = 0$ and so $\underline{b}_3$ remains unchanged. Finally, $B_3 = 9 > (3/4 - \mu_{3,2}^2)B_2 = 3$ and so we set $k = 4$ and halt.

Exercise 17.4.10. Perform the LLL algorithm by hand on the basis

$$\{(-1, 5, 0),\ (2, 5, 0),\ (8, 6, 16)\}.$$

Exercise 17.4.11. Perform the LLL algorithm by hand on the basis

$$\{(0, 3, 4), (-1, 3, 3), (5, 4, -7)\}.$$

Remark 17.4.12. Part 1 of Theorem 17.2.12 shows we have $\|\underline{b}_1\| \leq 2^{(n-1)/2}\lambda_1$. In other words, the LLL algorithm solves SVP up to an exponential factor but is not guaranteed to output a shortest vector in the lattice. Hence, LLL does not officially solve SVP.

In practice, at least for relatively small dimensions, the vector $\underline{b}_1$ output by the LLL algorithm is often much closer to the shortest vector than this bound would suggest, and in many cases will be a shortest vector in the lattice. In Example 17.4.9, the theoretical bound gives $\|\underline{b}_1\| \leq 2$, so $(0, 2, 0)$ would have been a possible value for $\underline{b}_1$ (but it wasn't).

17.5 Complexity of LLL

We now show that the LLL algorithm terminates and runs in polynomial-time. The original paper of Lenstra, Lenstra and Lovász [372] proves polynomial termination for any lattice in $\mathbb{R}^m$ but only gives a precise complexity for lattices in $\mathbb{Z}^m$.

Theorem 17.5.1. *Let L be a lattice in $\mathbb{Z}^m$ with basis $\underline{b}_1, \ldots, \underline{b}_n$ and let $X \in \mathbb{Z}_{\geq 2}$ be such that $\|\underline{b}_i\|^2 \leq X$ for $1 \leq i \leq n$. Let $1/4 < \delta < 1$. Then the LLL algorithm with factor δ terminates and performs $O(n^2 \log(X))$ iterations.*

Proof: We need to bound the number of "backtracks" in Algorithm 25. This number is at most n plus the number of swaps. So it suffices to bound the number of swaps by $O(n^2 \log(X))$.

For $1 \leq i \leq n-1$ define the $i \times m$ matrix $B_{(i)}$ formed by the first i basis vectors for the lattice. Define $d_i = \det(B_{(i)} B_{(i)}^T) \in \mathbb{Z}$, which is the square of the volume of the sublattice generated by the rows of $B_{(i)}$. Hence

$$d_i = \prod_{j=1}^{i} B_j = \prod_{j=1}^{i} \|\underline{b}_i^*\|^2 \leq \prod_{j=1}^{i} \|\underline{b}_i\|^2 \leq X^i.$$

Define

$$D = \prod_{i=1}^{n-1} d_i = \prod_{i=1}^{n-1} B_i^{n-i}.$$

It follows that $D \leq X^{(n-1)n/2}$.

Two vectors $\underline{b}_k$ and $\underline{b}_{k-1}$ are swapped when $B_k < (\delta - \mu_{k,k-1}^2) B_{k-1}$. By Lemma 17.4.3, the new values for B_{k-1} and B_k are $B'_{k-1} = B_k + \mu_{k,k-1}^2 B_{k-1}$ and $B'_k = B_{k-1} B_k / B'_{k-1}$. Let d'_i be the new values for the d_i. We have $d'_i = d_i$ when $1 \leq i < k-1$. By the Lovász condition $B'_{k-1} \leq \delta B_{k-1}$. Hence, $d'_{k-1} \leq \delta d_{k-1}$. Finally, since $B'_{k-1} B'_k = B_{k-1} B_k$ we have $d'_i = d_i$ for $k \leq i \leq n$. Hence, swapping $\underline{b}_k$ and $\underline{b}_{k-1}$ always strictly reduces D.

On the other hand, we always have[5] $d_i \in \mathbb{Z}$ and so $D \geq 1$. It follows that the number of swaps in the LLL algorithm is at most[6] $\log_\delta(X^{(n-1)n/2}) = O(n^2 \log(X))$. Hence the algorithm requires $O(n^2 \log(X))$ iterations of the main loop. □

Algorithm 25 and Theorem 17.5.1 provide a proof that an LLL-reduced basis exists for every lattice.

Exercise 17.5.2. Let $n \in \mathbb{N}$. Show that Hermite's constant satisfies $\gamma_n \leq 2^{(n-1)/4}$ (this bound can be improved to $(4/3)^{(n-1)/2}$; see [372]).

It is clear that if $L \subset \mathbb{Z}^m$ then LLL can be implemented using exact $\mathbb{Q}$ arithmetic, and hence exact integer arithmetic. But we need to show that the size of the integers does not explode. The analysis given already for the Gram-Schmidt algorithm (for example, Lemma 17.3.2) provides most of the tools we need.

Theorem 17.5.3. *Let L be a lattice in $\mathbb{Z}^m$ with basis $\underline{b}_1, \ldots, \underline{b}_n$ and let $X \in \mathbb{Z}_{\geq 2}$ be such that $\|\underline{b}_i\|^2 \leq X$ for $1 \leq i \leq n$. Then the LLL algorithm requires arithmetic operations on integers of size $O(n \log(X))$.*

Proof: (Sketch) The bounds on the sizes of the $\underline{b}_i^*$ follow the same methods as used in the proof of Theorem 17.3.4. Since $\|\underline{b}_i^*\|$ is never increased during the algorithm (indeed, the vectors are specifically permuted to reduce the $\|\underline{b}_i^*\|$) we have $\|\underline{b}_i^*\| \leq X^{1/2}$ at the end of each iteration. Since $d_{i-1} \underline{b}_i^* \in \mathbb{Z}^n$ and $|d_{i-1}| \leq X^{i-1}$ it follows that the entries of $\underline{b}_i^*$ can be written as $n_{i,j}^* / d_{i-1}$ where $|n_{i,j}^*| \leq X^i$.

Let us now consider the values $\|\underline{b}_i\|^2$ at the end of each iteration. These values all start bounded by X. As the algorithm proceeds the values are either not yet changed (and hence still bounded by X) or have been modified so that the Gram-Schmidt basis is size reduced (and possibly swapped to an earlier position in the list of vectors). After

[5] To apply this argument it is necessary to use the square of the determinant. An integer lattice that does not have full rank does not necessarily have integer determinant.

[6] Recall that $1/4 < \delta < 1$ is considered as a fixed constant.

each size reduction step (and before swapping) we have

$$\underline{b}_i = \underline{b}_i^* + \sum_{j=1}^{i-1} \mu_{i,j} \underline{b}_j^*$$

with $-1/2 \leq \mu_{i,j} \leq 1/2$ and so

$$\|\underline{b}_i\|^2 = \|\underline{b}_i^*\|^2 + \sum_{j=1}^{i-1} \mu_{i,j}^2 \|\underline{b}_j^*\|^2 \leq nX. \tag{17.2}$$

Hence, we have $\|\underline{b}_i\| \leq \sqrt{nX}$ at the end of each iteration and so the entries of $\underline{b}_i$ are all integers bounded by $\sqrt{nX}$.

The remaining detail is to bound the sizes of the $\mu_{i,j}$ and the sizes of intermediate values in line 6 of Algorithm 25. We refer to the proof of Proposition 1.26 of [372] for the bounds $|\mu_{i,j}| \leq 2^{n-i}(nX^{n-1})^{1/2}$ and for further details. □

Corollary 17.5.4. *Let L be a lattice in $\mathbb{Z}^m$ with basis $\underline{b}_1, \ldots, \underline{b}_n$ and let $X \in \mathbb{Z}_{\geq 2}$ be such that $\|\underline{b}_i\|^2 \leq X$ for $1 \leq i \leq n$. Then the LLL algorithm requires $O(n^3 m \log(X))$ arithmetic operations on integers of size $O(n \log(X))$. Using naive arithmetic gives running time $O(n^5 m \log(X)^3)$ bit operations.*

Proof: Theorem 17.5.1 implies that the algorithm requires $O(n^2 \log(X))$ iterations of the main loop. Within each iteration there are n operations on vectors of length m. Hence $O(n^3 m \log(X))$ arithmetic operations. Theorem 17.5.3 implies that all arithmetic operations are on integers of size $O(n \log(X))$. □

Remark 17.5.5.

1. Since the input size is $O(nm \log(X))$ and $n \leq m$ the running time is cubic in the input size.

2. Note that the bounds on the sizes of integers involved in the LLL algorithm are $O(n \log(X))$ bits for the values $\mu_{i,j}$ and entries of $\underline{b}_i^*$ while only $O(\log(n) + \log(X))$ bits are needed for the entries in the vectors $\underline{b}_i$. This is not just an artifact of the proof, but is a genuine phenomenon; it can already be seen in Example 17.4.9 where the $\underline{b}_i$ all have very small integer coordinates and yet $\mu_{2,1} = 45/229$.

   This leads to the idea of representing the $\mu_{i,j}$ and $\underline{b}_i^*$ using approximate (floating-point) arithmetic and keeping exact integer arithmetic only for the $\underline{b}_i$. Variants of LLL using floating-point arithmetic for the Gram-Schmidt vectors are much faster than the basic LLL algorithm presented in this chapter. Indeed, the basic algorithm is almost never used in practice.

   A problem with using floating-point approximations is that comparisons now become inexact, and this leads to problems with both termination and correctness of the output. Implementing and analysing floating-point LLL algorithms is beyond the scope of this book. We refer to Stehlé [581] and Schnorr [524] for surveys of this topic.

3. One can show (e.g., using equation (17.2)) that the complexity statement holds also for $X = \max\{\|\underline{b}_i^*\| : 1 \leq i \leq n\}$, which could be smaller than $\max\{\|\underline{b}_i\| : 1 \leq i \leq n\}$.

4. Sometimes one is interested in reducing lattice bases that are in $\mathbb{Q}^m$ and not $\mathbb{Z}^m$. Suppose all rational numbers in the basis B have numerator and denominator bounded by X. One can obtain an integer matrix by multiplying B by an integer that clears all denominators, but the resulting integers could be as big as X^{mn}.

This gives a worst-case complexity of $O(n^8 m^4 \log(X)^3)$ bit operations for lattice basis reduction.

Some applications such as simultaneous Diophantine approximation (see Section 19.5) and the hidden number problem (see Section 21.7.1) have at most m non-integer entries, giving a complexity of $O(n^5 m^4 \log(X)^3)$ bit operations.

17.6 Variants of the LLL Algorithm

There are many refinements of the LLL algorithm that are beyond the scope of the brief summary in this book. We list some of these now.

- As mentioned earlier, it is necessary to use floating-point arithmetic to obtain a fast version of the LLL algorithm. A variant of floating-point LLL whose running time grows quadratically in $\log(X)$ (rather than cubicly, as usual) is the L^2 algorithm of Nguyen and Stehlé [453] (also see Stehlé [581]).

- Schnorr-Euchner "deep insertions". The idea is that, rather than just swapping $\underline{b}_k$ and $\underline{b}_{k-1}$ in the LLL algorithm, one can move $\underline{b}_k$ much earlier in the list of vectors if B_k is sufficiently small. With standard LLL we have shown that swapping $\underline{b}_k$ and $\underline{b}_{k-1}$ changes B_k to $B_k + \mu_{k,k-1}^2 B_{k-1}$. A similar argument shows that inserting $\underline{b}_k$ between $\underline{b}_{i-1}$ and $\underline{b}_i$ for some $1 < i < k$ changes B_k to

$$B = B_k + \sum_{j=1}^{k-1} \mu_{k,j}^2 B_j$$

  Hence, one can let i be the smallest index such that $B < \frac{3}{4} B_i$ and insert $\underline{b}_k$ between $\underline{b}_{i-1}$ and $\underline{b}_i$ (i.e., reorder the vectors $\underline{b}_i, \ldots, \underline{b}_k$ as $\underline{b}_k, \underline{b}_i, \ldots, \underline{b}_{k-1}$). We refer to Schnorr and Euchner [525] and Section 2.6.2 of Cohen [135] for more details.

- Our presentation of the LLL algorithm was for the Euclidean norm. The algorithm has been extended to work with any norm by Lovász and Scarf [395] (also see Kaib and Ritter [324]).

  In practice, if one wants results for a norm other than the Euclidean norm, one usually performs ordinary LLL reduction with respect to the Euclidean norm and then uses the standard relations between norms (Lemma A.10.2) to determine the quality of the resulting vectors.

- Another important approach to lattice basis reduction is the block Korkine-Zolotarev algorithm due to Schnorr [520]. We mention this further in Section 18.5.