7.13 Lattice Reduction Algorithms

We have now seen several cryptosystems whose security depends on the difficulty of solving apprSVP and/or apprCVP in various types of lattices. In this section we describe an algorithm called LLL that solves these problems to within a factor of C^n, where C is a small constant and n is the dimension of the lattice. Thus in small dimensions, the LLL algorithm comes close to solving SVP and CVP, but in large dimensions it does not do as well. Ultimately, the security of lattice-based cryptosystems depends on the inability of LLL and other lattice reduction algorithms to efficiently solve apprSVP and apprCVP to within a factor of, say, $\mathcal{O}(\sqrt{n}\,)$. We begin in Sect. 7.13.1 with Gauss's lattice reduction algorithm, which rapidly solves SVP in lattices of dimension 2. Next, in Sect. 7.13.2, we describe and analyze the LLL algorithm. Section 7.13.3 explains how to combine LLL and Babai's algorithm to solve apprCVP, and we conclude in Sect. 7.13.4 by briefly describing some generalizations of LLL.

7.13.1 Gaussian Lattice Reduction in Dimension 2

The algorithm for finding an optimal basis in a lattice of dimension 2 is essentially due to Gauss. The underlying idea is to alternately subtract multiples of one basis vector from the other until further improvement is not possible.

So suppose that $L \subset \mathbb{R}^2$ is a 2-dimensional lattice with basis vectors $\boldsymbol{v}_1$ and $\boldsymbol{v}_2$. Swapping $\boldsymbol{v}_1$ and $\boldsymbol{v}_2$ if necessary, we may assume that $\|\boldsymbol{v}_1\| < \|\boldsymbol{v}_2\|$. We now try to make $\boldsymbol{v}_2$ smaller by subtracting a multiple of $\boldsymbol{v}_1$. If we were allowed to subtract an arbitrary multiple of $\boldsymbol{v}_1$, then we could replace $\boldsymbol{v}_2$ with the vector

$$\boldsymbol{v}_2^* = \boldsymbol{v}_2 - \frac{\boldsymbol{v}_1 \cdot \boldsymbol{v}_2}{\|\boldsymbol{v}_1\|^2}\boldsymbol{v}_1,$$

which is orthogonal to $\boldsymbol{v}_1$. The vector $\boldsymbol{v}_2^*$ is the projection of $\boldsymbol{v}_2$ onto the orthogonal complement of $\boldsymbol{v}_1$. (See Fig. 7.7.)

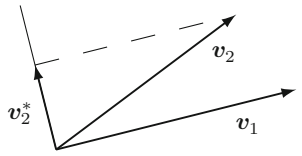

Figure 7.7: $\boldsymbol{v}_2^*$ is the projection of $\boldsymbol{v}_2$ onto the orthogonal complement of $\boldsymbol{v}_1$

Of course, this is cheating, since the vector $\boldsymbol{v}_2^*$ is unlikely to be in L. In reality we are allowed to subtract only integer multiples of $\boldsymbol{v}_1$ from $\boldsymbol{v}_2$. So we do the best that we can and replace $\boldsymbol{v}_2$ with the vector

$$\boldsymbol{v}_2 - m\boldsymbol{v}_1 \qquad \text{with} \quad m = \left\lfloor \frac{\boldsymbol{v}_1 \cdot \boldsymbol{v}_2}{\|\boldsymbol{v}_1\|^2} \right\rceil.$$

If $\boldsymbol{v}_2$ is still longer than $\boldsymbol{v}_1$, then we stop. Otherwise, we swap $\boldsymbol{v}_1$ and $\boldsymbol{v}_2$ and repeat the process. Gauss proved that this process terminates and that the resulting basis for L is extremely good. The next proposition makes this precise.

Proposition 7.66 (Gaussian Lattice Reduction)**.** *Let $L \subset \mathbb{R}^2$ be a 2-dimensional lattice with basis vectors $\boldsymbol{v}_1$ and $\boldsymbol{v}_2$. The following algorithm terminates and yields a good basis for L.*

Loop
  If $\|\boldsymbol{v}_2\| < \|\boldsymbol{v}_1\|$, swap $\boldsymbol{v}_1$ and $\boldsymbol{v}_2$.
  Compute $m = \lfloor \boldsymbol{v}_1 \cdot \boldsymbol{v}_2 / \|\boldsymbol{v}_1\|^2 \rceil$.
  If $m = 0$, return the basis vectors $\boldsymbol{v}_1$ and $\boldsymbol{v}_2$.
  Replace $\boldsymbol{v}_2$ with $\boldsymbol{v}_2 - m\boldsymbol{v}_1$.
Continue Loop

More precisely, when the algorithm terminates, the vector $\boldsymbol{v}_1$ is a shortest nonzero vector in L, so the algorithm solves SVP*. Further, the angle θ between $\boldsymbol{v}_1$ and $\boldsymbol{v}_2$ satisfies $|\cos\theta| \le \|\boldsymbol{v}_1\|/2\|\boldsymbol{v}_2\|$, so in particular, $\frac{\pi}{3} \le \theta \le \frac{2\pi}{3}$.*

Proof. We prove that $\boldsymbol{v}_1$ is a smallest nonzero lattice vector and leave the other parts of the proof to the reader. So we suppose that the algorithm has terminated and returned the vectors $\boldsymbol{v}_1$ and $\boldsymbol{v}_2$. This means that $\|\boldsymbol{v}_2\| \ge \|\boldsymbol{v}_1\|$ and that

$$\frac{|\boldsymbol{v}_1 \cdot \boldsymbol{v}_2|}{\|\boldsymbol{v}_1\|^2} \le \frac{1}{2}. \tag{7.51}$$

(Geometrically, condition (7.51) says that we cannot make $\boldsymbol{v}_2$ smaller by subtracting an integral multiple of $\boldsymbol{v}_1$ from $\boldsymbol{v}_2$.) Now suppose that $\boldsymbol{v} \in L$ is any nonzero vector in L. Writing

$$\boldsymbol{v} = a_1\boldsymbol{v}_1 + a_2\boldsymbol{v}_2 \quad \text{with } a_1, a_2 \in \mathbb{Z},$$

we find that

$$\begin{aligned}
\|\boldsymbol{v}\|^2 &= \|a_1\boldsymbol{v}_1 + a_2\boldsymbol{v}_2\|^2 \\
&= a_1^2\|\boldsymbol{v}_1\|^2 + 2a_1a_2(\boldsymbol{v}_1 \cdot \boldsymbol{v}_2) + a_2^2\|\boldsymbol{v}_2\|^2 \\
&\ge a_1^2\|\boldsymbol{v}_1\|^2 - 2|a_1a_2|\,|\boldsymbol{v}_1 \cdot \boldsymbol{v}_2| + a_2^2\|\boldsymbol{v}_2\|^2 \\
&\ge a_1^2\|\boldsymbol{v}_1\|^2 - |a_1a_2|\|\boldsymbol{v}_1\|^2 + a_2^2\|\boldsymbol{v}_2\|^2 \quad \text{from (7.51)}, \\
&\ge a_1^2\|\boldsymbol{v}_1\|^2 - |a_1a_2|\|\boldsymbol{v}_1\|^2 + a_2^2\|\boldsymbol{v}_1\|^2 \quad \text{since } \|\boldsymbol{v}_2\| \ge \|\boldsymbol{v}_1\|, \\
&= \bigl(a_1^2 - |a_1|\,|a_2| + a_2^2\bigr)\|\boldsymbol{v}_1\|^2.
\end{aligned}$$

For any real numbers t_1 and t_2, the quantity

$$t_1^2 - t_2t_2 + t_2^2 = \left(t_1 - \frac{1}{2}t_2\right)^2 + \frac{3}{4}t_2^2 = \frac{3}{4}t_1^2 + \left(\frac{1}{2}t_1 - t_2\right)^2$$

is not zero unless $t_1 = t_2 = 0$. So the fact that a_1 and a_2 are integers and not both 0 tells us that $\|\boldsymbol{v}\|^2 \geq \|\boldsymbol{v}_1\|^2$. This proves that $\boldsymbol{v}_1$ is a smallest nonzero vector in L. □

Example 7.67. We illustrate Gauss's lattice reduction algorithm (Proposition 7.66) with the lattice L having basis

$$\boldsymbol{v}_1 = (66586820, 65354729) \quad \text{and} \quad \boldsymbol{v}_2 = (6513996, 6393464).$$

We first compute $\|\boldsymbol{v}_1\|^2 \approx 8.71 \cdot 10^{15}$ and $\|\boldsymbol{v}_2\|^2 \approx 8.33 \cdot 10^{13}$. Since $\boldsymbol{v}_2$ is shorter than $\boldsymbol{v}_1$, we swap them, so now $\boldsymbol{v}_1 = (6513996, 6393464)$ and $\boldsymbol{v}_2 = (66586820, 65354729)$.

Next we subtract a multiple of $\boldsymbol{v}_1$ from $\boldsymbol{v}_2$. The multiplier is

$$m = \left\lfloor \frac{\boldsymbol{v}_1 \cdot \boldsymbol{v}_2}{\|\boldsymbol{v}_1\|^2} \right\rceil = \lfloor 10.2221 \rceil = 10,$$

so we replace $\boldsymbol{v}_2$ with

$$\boldsymbol{v}_2 - m\boldsymbol{v}_1 = (1446860, 1420089).$$

This new vector has norm $\|\boldsymbol{v}_2\|^2 \approx 4.11 \cdot 10^{12}$, which is smaller than $\|\boldsymbol{v}_1\|^2 \approx 8.33 \cdot 10^{13}$, so again we swap,

$$\boldsymbol{v}_1 = (1446860, 1420089) \quad \text{and} \quad \boldsymbol{v}_2 = (6513996, 6393464).$$

We repeat the process with $m = \lfloor \boldsymbol{v}_1 \cdot \boldsymbol{v}_2 / \|\boldsymbol{v}_1\|^2 \rceil = \lfloor 4.502 \rceil = 5$, which gives the new vector

$$\boldsymbol{v}_2 - m\boldsymbol{v}_1 = (-720304, -706981)$$

having norm $\|\boldsymbol{v}_2\|^2 \approx 1.01 \cdot 10^{12}$, so again we swap $\boldsymbol{v}_1$ and $\boldsymbol{v}_2$. Continuing this process leads to smaller and smaller bases until, finally, the algorithm terminates. The step by step results of the algorithm, including the value of m used at each stage, are listed in the following table:

Step	$\boldsymbol{v}_1$	$\boldsymbol{v}_2$	m
1	$(6513996, 6393464)$	$(66586820, 65354729)$	10
2	$(1446860, 1420089)$	$(6513996, 6393464)$	5
3	$(-720304, -706981)$	$(1446860, 1420089)$	-2
4	$(6252, 6127)$	$(-720304, -706981)$	-115
5	$(-1324, -2376)$	$(6252, 6127)$	-3
6	$(2280, -1001)$	$(-1324, -2376)$	0

The final basis is quite small, and $(2280, -1001)$ is a solution to SVP for the lattice L.

7.13.2 The LLL Lattice Reduction Algorithm

Gauss's lattice reduction algorithm (Proposition 7.66) gives an efficient way to find a shortest nonzero vector in a lattice of dimension 2, but as the dimension increases, the shortest vector problem becomes much harder . A major advance came in 1982 with the publication of the LLL algorithm [77]. In this section we give a full description of the LLL algorithm, and in the next section we briefly describe some of its generalizations.

Suppose that we are given a basis $\{\boldsymbol{v}_1, \boldsymbol{v}_2, \ldots, \boldsymbol{v}_n\}$ for a lattice L. Our object is to transform the given basis into a "better" basis. But what do we mean by a better basis? We would like the vectors in the better basis to be as short as possible, beginning with the shortest vector that we can find, and then with vectors whose lengths increase as slowly as possible until we reach the last vector in the basis. Alternatively, we would like the vectors in the better basis to be as orthogonal as possible to one another, i.e., so that the dot products $\boldsymbol{v}_i \cdot \boldsymbol{v}_j$ are as close to zero as possible.

Recall that Hadamard's inequality (Proposition 7.19) says that

$$\det L = \operatorname{Vol}(\mathcal{F}) \le \|\boldsymbol{v}_1\|\,\|\boldsymbol{v}_2\| \cdots \|\boldsymbol{v}_n\|, \tag{7.52}$$

where $\operatorname{Vol}(\mathcal{F})$ is the volume of a fundamental domain for L. The closer that the basis comes to being orthogonal, the closer that the inequality (7.52) comes to being an equality.

To assist us in creating an improved basis, we begin by constructing a Gram–Schmidt orthogonal basis as described in Theorem 7.13. Thus we start with $\boldsymbol{v}_1^* = \boldsymbol{v}_1$, and then for $i \ge 2$ we let

$$\boldsymbol{v}_i^* = \boldsymbol{v}_i - \sum_{j=1}^{i-1} \mu_{i,j} \boldsymbol{v}_j^*, \quad \text{where} \quad \mu_{i,j} = \frac{\boldsymbol{v}_i \cdot \boldsymbol{v}_j^*}{\|\boldsymbol{v}_j^*\|^2} \quad \text{for} \quad 1 \le j \le i-1. \tag{7.53}$$

The collection of vectors $\mathcal{B}^* = \{\boldsymbol{v}_1^*, \boldsymbol{v}_2^*, \ldots, \boldsymbol{v}_n^*\}$ is an orthogonal basis for the *vector space* spanned by $\mathcal{B} = \{\boldsymbol{v}_1, \boldsymbol{v}_2, \ldots, \boldsymbol{v}_n\}$, but note that $\mathcal{B}^*$ is *not* a basis for the *lattice* L spanned by $\mathcal{B}$, because the Gram–Schmidt process (7.53) involves taking linear combinations with nonintegral coefficients. However, as we now prove, it turns out that the two bases have the same determinant.

Proposition 7.68. *Let $\mathcal{B} = \{\boldsymbol{v}_1, \boldsymbol{v}_2, \ldots, \boldsymbol{v}_n\}$ be a basis for a lattice L and let $\mathcal{B}^* = \{\boldsymbol{v}_1^*, \boldsymbol{v}_2^*, \ldots, \boldsymbol{v}_n^*\}$ be the associated Gram–Schmidt orthogonal basis as described in Theorem 7.13. Then*

$$\det(L) = \prod_{i=1}^{n} \|\boldsymbol{v}_i^*\|.$$

Proof. Let $F = F(\boldsymbol{v}_1, \ldots, \boldsymbol{v}_n)$ be the matrix (7.11) described in Proposition 7.20. This is the matrix whose rows are the coordinates of $\boldsymbol{v}_1, \ldots, \boldsymbol{v}_n$. The proposition tells us that $\det(L) = |\det F|$.

Let $F^* = F(\boldsymbol{v}_1^*, \dots, \boldsymbol{v}_n^*)$ be the analogous matrix whose rows are the vectors $\boldsymbol{v}_1^*, \dots, \boldsymbol{v}_n^*$. Then (7.53) tells us that the matrices F and F^* are related by

$$MF^* = F,$$

where M is the change of basis matrix

$$M = \begin{pmatrix} 1 & 0 & 0 & \cdots & 0 & 0 \\ \mu_{2,1} & 1 & 0 & \cdots & 0 & 0 \\ \mu_{3,1} & \mu_{3,2} & 1 & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & & \vdots \\ \mu_{n-1,1} & \mu_{n-1,2} & \mu_{n-1,3} & \cdots & 1 & 0 \\ \mu_{n,1} & \mu_{n,2} & \mu_{n,3} & \cdots & \mu_{n,n-1} & 1 \end{pmatrix}.$$

Note that M is lower diagonal with 1's on the diagonal, so $\det(M) = 1$. Hence

$$\det(L) = |\det F| = |\det(MF^*)| = |(\det M)(\det F^*)| = |\det F^*| = \prod_{i=1}^{n} \|\boldsymbol{v}_i^*\|.$$

(The last equality follows from the fact that the $\boldsymbol{v}_i^*$, which are the rows of F^*, are pairwise orthogonal.) □

Definition. Let V be a vector space, and let $W \subset V$ be a vector subspace of V. The *orthogonal complement of* W *(in* V*)* is

$$W^\perp = \{\boldsymbol{v} \in V : \boldsymbol{v} \cdot \boldsymbol{w} = 0 \quad \text{for all } \boldsymbol{w} \in W\}.$$

It is not hard to see that $W^\perp$ is also a vector subspace of V and that every vector $\boldsymbol{v} \in V$ can be written as a sum $\boldsymbol{v} = \boldsymbol{w} + \boldsymbol{w}'$ for unique vectors $\boldsymbol{w} \in W$ and $\boldsymbol{w}' \in W^\perp$. (See Exercise 7.46.)

Using the notion of orthogonal complement, we can describe the intuition behind the Gram–Schmidt construction as follows:

$$\boldsymbol{v}_i^* = \text{Projection of } \boldsymbol{v}_i \text{ onto } \operatorname{Span}(\boldsymbol{v}_1, \dots, \boldsymbol{v}_{i-1})^\perp.$$

Although $\mathcal{B}^* = \{\boldsymbol{v}_1^*, \boldsymbol{v}_2^*, \dots, \boldsymbol{v}_n^*\}$ is not a basis for the original lattice L, we use the set $\mathcal{B}^*$ of associated Gram–Schmidt vectors to define a concept that is crucial for the LLL algorithm.

Definition. Let $\mathcal{B} = \{\boldsymbol{v}_1, \boldsymbol{v}_2, \dots, \boldsymbol{v}_n\}$ be a basis for a lattice L and let $\mathcal{B}^* = \{\boldsymbol{v}_1^*, \boldsymbol{v}_2^*, \dots, \boldsymbol{v}_n^*\}$ be the associated Gram–Schmidt orthogonal basis as described in Theorem 7.13. The basis $\mathcal{B}$ is said to be *LLL reduced* if it satisfies the following two conditions:

$$\textbf{(Size Condition)} \quad |\mu_{i,j}| = \frac{|\boldsymbol{v}_i \cdot \boldsymbol{v}_j^*|}{\|\boldsymbol{v}_j^*\|^2} \le \frac{1}{2} \qquad \text{for all } 1 \le j < i \le n.$$

$$\textbf{(Lovász Condition)} \quad \|\boldsymbol{v}_i^*\|^2 \ge \left(\frac{3}{4} - \mu_{i,i-1}^2\right) \|\boldsymbol{v}_{i-1}^*\|^2 \quad \text{for all } 1 < i \le n.$$

There are several different ways to state the Lovász condition. For example, it is equivalent to the inequality

$$\|\boldsymbol{v}_i^* + \mu_{i,i-1}\boldsymbol{v}_{i-1}^*\|^2 \ge \frac{3}{4}\|\boldsymbol{v}_{i-1}^*\|^2,$$

and it is also equivalent to the statement that

$$\begin{aligned}\big\|&\text{Projection of } \boldsymbol{v}_i \text{ onto } \operatorname{Span}(\boldsymbol{v}_1,\ldots,\boldsymbol{v}_{i-2})^\perp\big\| \\ &\qquad\ge \frac{3}{4}\big\|\text{Projection of } \boldsymbol{v}_{i-1} \text{ onto } \operatorname{Span}(\boldsymbol{v}_1,\ldots,\boldsymbol{v}_{i-2})^\perp\big\|.\end{aligned}$$

The fundamental result of Lenstra, Lenstra, and Lovász [77] says that an LLL reduced basis is a good basis and that it is possible to compute an LLL reduced basis in polynomial time. We start by showing that an LLL reduced basis has desirable properties, after which we describe the LLL lattice reduction algorithm.

Theorem 7.69. *Let L be a lattice of dimension n. Any LLL reduced basis $\{\boldsymbol{v}_1,\boldsymbol{v}_2,\ldots,\boldsymbol{v}_n\}$ for L has the following two properties*:

$$\prod_{i=1}^{n}\|\boldsymbol{v}_i\| \le 2^{n(n-1)/4}\det L, \tag{7.54}$$

$$\|\boldsymbol{v}_j\| \le 2^{(i-1)/2}\|\boldsymbol{v}_i^*\| \quad \textit{for all } 1\le j\le i\le n. \tag{7.55}$$

Further, the initial vector in an LLL reduced basis satisfies

$$\|\boldsymbol{v}_1\| \le 2^{(n-1)/4}|\det L|^{1/n} \quad \textit{and} \quad \|\boldsymbol{v}_1\| \le 2^{(n-1)/2}\min_{\boldsymbol{0}\ne\boldsymbol{v}\in\mathbf{L}}\|\boldsymbol{v}\|. \tag{7.56}$$

Thus an LLL reduced basis solves apprSVP *to within a factor of* $2^{(n-1)/2}$.

Proof. The Lovász condition and the fact that $|\mu_{i,i-1}|\le\frac{1}{2}$ imply that

$$\|\boldsymbol{v}_i^*\|^2 \ge \left(\frac{3}{4}-\mu_{i,i-1}^2\right)\|\boldsymbol{v}_{i-1}^*\|^2 \ge \frac{1}{2}\|\boldsymbol{v}_{i-1}^*\|^2. \tag{7.57}$$

Applying (7.57) repeatedly yields the useful estimate

$$\|\boldsymbol{v}_j^*\|^2 \le 2^{i-j}\|\boldsymbol{v}_i^*\|^2. \tag{7.58}$$

We now compute

$$\begin{aligned}\|\boldsymbol{v}_i\|^2 &= \left\|\boldsymbol{v}_i^* + \sum_{j=1}^{i-1}\mu_{i,j}\boldsymbol{v}_j^*\right\|^2 && \text{from (7.53)},\\ &= \|\boldsymbol{v}_i^*\|^2 + \sum_{j=1}^{i-1}\mu_{i,j}^2\|\boldsymbol{v}_j^*\|^2 && \text{since } \boldsymbol{v}_1^*,\ldots,\boldsymbol{v}_n^* \text{ are orthogonal},\end{aligned}$$

$$
\begin{aligned}
&\le \|\boldsymbol{v}_i^*\|^2 + \sum_{j=1}^{i-1} \frac{1}{4}\|\boldsymbol{v}_j^*\|^2 && \text{since } |\mu_{i,j}| \le \frac{1}{2},\\
&\le \|\boldsymbol{v}_i^*\|^2 + \sum_{j=1}^{i-1} 2^{i-j-2}\|\boldsymbol{v}_i^*\|^2 && \text{from (7.58)},\\
&= \frac{1+2^{i-1}}{2}\|\boldsymbol{v}_i^*\|^2\\
&\le 2^{i-1}\|\boldsymbol{v}_i^*\|^2 && \text{since } 1 \le 2^{i-1} \text{ for all } i \ge 1. \qquad (7.59)
\end{aligned}
$$

Multiplying (7.59) by itself for $1 \le i \le n$ yields

$$\prod_{i=1}^{n} \|\boldsymbol{v}_i\|^2 \le \prod_{i=1}^{n} 2^{i-1}\|\boldsymbol{v}_i^*\|^2 = 2^{n(n-1)/2}\prod_{i=1}^{n}\|\boldsymbol{v}_i^*\|^2 = 2^{n(n-1)/2}(\det L)^2,$$

where for the last equality we have used Proposition 7.68. Taking square roots completes the proof of (7.54).

Next, for any $j \le i$, we use (7.59) (with $i = j$) and (7.58) to estimate

$$\|\boldsymbol{v}_j\|^2 \le 2^{j-1}\|\boldsymbol{v}_j^*\|^2 \le 2^{j-1}\cdot 2^{i-j}\|\boldsymbol{v}_i^*\|^2 = 2^{i-1}\|\boldsymbol{v}_i^*\|^2.$$

Taking square roots gives (7.55).

Now we set $j = 1$ in (7.55), multiply over $1 \le i \le n$, and use Proposition 7.68 to obtain

$$\|\boldsymbol{v}_1\|^n \le \prod_{i=1}^{n} 2^{(i-1)/2}\|\boldsymbol{v}_i^*\| = 2^{n(n-1)/4}\prod_{i=1}^{n}\|\boldsymbol{v}_i^*\| = 2^{n(n-1)/4}\det L.$$

Taking nth roots gives the first estimate in (7.56).

To prove the second estimate, let $\boldsymbol{v} \in L$ be a nonzero lattice vector and write

$$\boldsymbol{v} = \sum_{j=1}^{i} a_j\boldsymbol{v}_j = \sum_{j=1}^{i} b_j\boldsymbol{v}_j^*$$

with $a_i \ne 0$. Note that $a_1, \dots, a_i$ are integers, while $b_i, \dots, b_i$ are real numbers. In particular, $|a_i| \ge 1$.

By construction, for any k we know that the vectors $\boldsymbol{v}_1^*, \dots, \boldsymbol{v}_k^*$ are pairwise orthogonal, and we proved (Theorem 7.13) that they span the same space as the vectors $\boldsymbol{v}_1, \dots, \boldsymbol{v}_k$. Hence

$$\boldsymbol{v}\cdot\boldsymbol{v}_i^* = a_i\boldsymbol{v}_i\cdot\boldsymbol{v}_i^* = b_i\boldsymbol{v}_i^*\cdot\boldsymbol{v}_i^* \quad \text{and} \quad \boldsymbol{v}_i\cdot\boldsymbol{v}_i^* = \boldsymbol{v}_i^*\cdot\boldsymbol{v}_i^*,$$

from which we conclude that $a_i = b_i$. Therefore $|b_i| = |a_i| \ge 1$, and using this and (7.55) (with $j = 1$) gives the estimate

$$\|\boldsymbol{v}\|^2 = \sum_{j=1}^{i} b_j^*\|\boldsymbol{v}_j^*\|^2 \ge b_i^2\|\boldsymbol{v}_i^*\|^2 \ge \|\boldsymbol{v}_i^*\|^2 \ge 2^{-(i-1)}\|\boldsymbol{v}_1\|^2 \ge 2^{-(n-1)}\|\boldsymbol{v}_1\|^2.$$

Taking square roots gives the second estimate in (7.56). □

Remark 7.70. Before describing the technicalities of the LLL algorithm, we make some brief remarks indicating the general underlying idea. Given a basis $\{\boldsymbol{v}_1, \boldsymbol{v}_2, \ldots, \boldsymbol{v}_n\}$, it is easy to form a new basis that satisfies the Size Condition. Roughly speaking, we do this by subtracting from $\boldsymbol{v}_k$ appropriate integer multiples of the previous vectors $\boldsymbol{v}_1, \ldots, \boldsymbol{v}_{k-1}$ so as to make $\boldsymbol{v}_k$ smaller. In the LLL algorithm, we do this in stages, rather than all at once, and we'll see that the size reduction condition depends on the ordering of the vectors. After doing size reduction, we check to see whether the Lovász condition is satisfied. If it is, then we have a (nearly) optimal ordering of the vectors. If not, then we reorder the vectors and do further size reduction.

For simplicity, and because it is the case that we need, we state and analyze the LLL algorithm for lattices in $\mathbb{Z}^n$. See Exercise 7.54 for the general case.

Theorem 7.71 (LLL Algorithm)**.** *Let $\{\boldsymbol{v}_1, \ldots, \boldsymbol{v}_n\}$ be a basis for a lattice L that is contained in $\mathbb{Z}^n$. The algorithm described in Fig. 7.8 terminates in a finite number of steps and returns an LLL reduced basis for L.*

More precisely, let $B = \max \|\boldsymbol{v}_i\|$. Then the algorithm executes the main k loop (Steps [4–14]*) no more than $\mathcal{O}(n^2 \log n + n^2 \log B)$ times. In particular, the LLL algorithm is a polynomial-time algorithm.*

Remark 7.72. The problem of efficiently implementing the LLL algorithm presents many challenges. First, size reduction and the Lovász condition use the Gram–Schmidt orthogonalized basis $\boldsymbol{v}_1^*, \ldots, \boldsymbol{v}_n^*$ and the associated projection factors $\mu_{i,j} = \boldsymbol{v}_i \cdot \boldsymbol{v}_j^* / \|\boldsymbol{v}_j^*\|^2$. In an efficient implementation of the LLL algorithm, one should compute these quantities as needed and store them for future use, recomputing only when necessary. We have not addressed this issue in Fig. 7.8, since it is not relevant for understanding the LLL algorithm, nor for proving that it returns an LLL reduced basis in polynomial time. See Exercise 7.50 for a more efficient version of the LLL algorithm.

Another major challenge arises from the fact that if one attempts to perform LLL reduction on an integer lattice using exact values, the intermediate calculations involve enormous numbers. Thus in working with lattices of high dimension, it is generally necessary to use floating point approximations, which leads to problems with round-off errors. We do not have space here to discuss this practical difficulty, but the reader should be aware that it exists.

Remark 7.73. Before embarking on the somewhat technical proof of Theorem 7.71, we discuss the intuition behind the swap step (Step [11]). The swap step is executed when the Lovász condition fails for $\boldsymbol{v}_k$, so

$$\begin{aligned}\Big\|\text{Projection of } \boldsymbol{v}_k \text{ onto } \operatorname{Span}(\boldsymbol{v}_1, \ldots, \boldsymbol{v}_{k-2})^{\perp}\Big\| \\ < \frac{3}{4}\Big\|\text{Projection of } \boldsymbol{v}_{k-1} \text{ onto } \operatorname{Span}(\boldsymbol{v}_1, \ldots, \boldsymbol{v}_{k-2})^{\perp}\Big\|. \quad (7.60)\end{aligned}$$

The goal of LLL is to produce a list of short vectors in increasing order of length. For each $1 \le \ell \le n$, let L_ℓ denote the lattice spanned by $\boldsymbol{v}_1, \ldots, \boldsymbol{v}_\ell$.

[1] Input a basis $\{\boldsymbol{v}_1, \ldots, \boldsymbol{v}_n\}$ for a lattice L
[2] Set $k = 2$
[3] Set $\boldsymbol{v}_1^* = \boldsymbol{v}_1$
[4] Loop while $k \leq n$
[5] Loop Down $j = k-1, k-2, \ldots, 2, 1$
[6] Set $\boldsymbol{v}_k = \boldsymbol{v}_k - \lfloor \mu_{k,j} \rceil \boldsymbol{v}_j$ [Size Reduction]
[7] End j Loop
[8] If $\|\boldsymbol{v}_k^*\|^2 \geq \left(\frac{3}{4} - \mu_{k,k-1}^2\right) \|\boldsymbol{v}_{k-1}^*\|^2$ [Lovász Condition]
[9] Set $k = k + 1$
[10] Else
[11] Swap $\boldsymbol{v}_{k-1}$ and $\boldsymbol{v}_k$ [Swap Step]
[12] Set $k = \max(k-1, 2)$
[13] End If
[14] End k Loop
[15] Return LLL reduced basis $\{\boldsymbol{v}_1, \ldots, \boldsymbol{v}_n\}$

Note: At each step, $\boldsymbol{v}_1^*, \ldots, \boldsymbol{v}_k^*$ is the orthogonal set of vectors obtained by applying Gram–Schmidt (Theorem 7.13) to the current values of $\boldsymbol{v}_1, \ldots, \boldsymbol{v}_k$, and $\mu_{i,j}$ is the associated quantity $(\boldsymbol{v}_i \cdot \boldsymbol{v}_j^*)/\|\boldsymbol{v}_j^*\|^2$.

Figure 7.8: The LLL lattice reduction algorithm

Note that as LLL progresses, the sublattices L_ℓ change due to the swap step; only L_n remains the same, since it is the entire lattice. What LLL attempts to do is to find an ordering of the basis vectors (combined with size reductions whenever possible) that minimizes the determinants $\det(L_\ell)$, i.e., LLL attempts to minimize the volumes of the fundamental domains of the sublattices $L_1, \ldots, L_n$.

If the number 3/4 in (7.60) is replaced by the number 1, then the LLL algorithm does precisely this; it swaps $\boldsymbol{v}_k$ and $\boldsymbol{v}_{k-1}$ whenever doing so reduces the value of $\det L_{k-1}$. Unfortunately, if we use 1 instead of 3/4, then it is an open problem whether the LLL algorithm terminates in polynomial time.

If we use 3/4, or any other constant strictly less than 1, then LLL runs in polynomial time, but we may miss an opportunity to reduce the size of a determinant by passing up a swap. For example, in the very first step, we swap only if $\|\boldsymbol{v}_2\| < \frac{3}{4}\|\boldsymbol{v}_1\|$, while we could reduce the determinant by swapping whenever $\|\boldsymbol{v}_2\| < \|\boldsymbol{v}_1\|$. In practice, one often takes a constant larger than 3/4, but less than 1, in the Lovász condition. (See Exercise 7.51.)

Note that an immediate effect of swapping at stage k is (usually) to make the new value of $\mu_{k,k-1}$ larger. This generally allows us to size reduce the

new $\boldsymbol{v}_k$ using the new $\boldsymbol{v}_{k-1}$, so swapping results in additional size reduction among the basis vectors, making them more orthogonal.

Proof (*sketch*) *of Theorem 7.71.* For simplicity, and because it is the case that we need, we will assume that $L \subset \mathbb{Z}^n$ is a lattice whose vectors have integral coordinates.

It is clear that if the LLL algorithm terminates, then it terminates with an LLL reduced basis, since the j-loop (Steps [5–7]) ensures that the basis satisfies the size condition, and the fact that $k = n+1$ on termination means that every vector in the basis has passed the Lovász condition test in Step [8].

However, it is not clear that the algorithm actually terminates, because the k-increment in Step [9] is offset by the k-decrement in Step [12]. What we will do is show that Step [12] is executed only a finite number of times. Since either Step [9] or Step [12] is executed on each iteration of the k-loop, this ensures that k eventually becomes larger than n and the algorithm terminates.

Let $\boldsymbol{v}_1, \ldots, \boldsymbol{v}_n$ be a basis of L and let $\boldsymbol{v}_1^*, \ldots, \boldsymbol{v}_n^*$ be the associated Gram–Schmidt orthogonalized basis from Theorem 7.13. For each $\ell = 1, 2, \ldots, n$, we let

$$L_\ell = \text{lattice spanned by } \boldsymbol{v}_1, \ldots, \boldsymbol{v}_\ell,$$

and we define quantities

$$d_\ell = \prod_{i=1}^{\ell} \|\boldsymbol{v}_i^*\|^2 \qquad \text{and} \qquad D = \prod_{\ell=1}^{n} d_\ell = \prod_{i=1}^{n} \|\boldsymbol{v}_i^*\|^{2(n+1-i)}.$$

Using an argument similar to the proof of Theorem 7.68, one can show that $\det(L_\ell)^2 = d_\ell$; see Exercise 7.14(b,d).

During the LLL algorithm, the value of D changes only when we execute the swap step (Step [11]). More precisely, when [11] is executed, the only d_ℓ that changes is d_{k-1}, since if $\ell < k-1$, then d_ℓ involves neither $\boldsymbol{v}_{k-1}^*$ nor $\boldsymbol{v}_k^*$, while if $\ell \ge k$, then the product defining d_ℓ includes both $\boldsymbol{v}_{k-1}^*$ and $\boldsymbol{v}_k^*$, so the product doesn't change if we swap them.

We can estimate the change in d_{k-1} by noting that when [11] is executed, the Lovász condition in Step [8] is false, so we have

$$\|\boldsymbol{v}_k^*\|^2 < \left(\frac{3}{4} - \mu_{k,k-1}^2\right) \|\boldsymbol{v}_{k-1}^*\|^2 \le \frac{3}{4} \|\boldsymbol{v}_{k-1}^*\|^2.$$

Hence the effect of swapping $\boldsymbol{v}_k^*$ and $\boldsymbol{v}_{k-1}^*$ in Step [11] is to change the value of d_{k-1} as follows:

$$\begin{aligned} d_{k-1}^{\text{new}} &= \|\boldsymbol{v}_1^*\|^2 \cdot \|\boldsymbol{v}_2^*\|^2 \cdots \|\boldsymbol{v}_{k-2}^*\|^2 \cdot \|\boldsymbol{v}_k^*\|^2 \\ &= \|\boldsymbol{v}_1^*\|^2 \cdot \|\boldsymbol{v}_2^*\|^2 \cdots \|\boldsymbol{v}_{k-2}^*\|^2 \cdot \|\boldsymbol{v}_{k-1}^*\|^2 \cdot \frac{\|\boldsymbol{v}_k^*\|^2}{\|\boldsymbol{v}_{k-1}^*\|^2} \\ &= d_{k-1}^{\text{old}} \cdot \frac{\|\boldsymbol{v}_k^*\|^2}{\|\boldsymbol{v}_{k-1}^*\|^2} \le \frac{3}{4} d_{k-1}^{\text{old}}. \end{aligned}$$

Hence if the swap step [11] is executed N times, then the value of D is reduced by a factor of at least $(3/4)^N$, since each swap reduces the value of some d_ℓ by at least a factor of $3/4$ and D is the product of all of the d_ℓ's.

Since we have assumed that the lattice L is contained in $\mathbb{Z}^n$, the basis vectors $\boldsymbol{v}_1, \ldots, \boldsymbol{v}_\ell$ of L_ℓ have integer coordinates. It follows from the definition of d_ℓ and Exercise 7.14(d) that

$$d_\ell = \prod_{i=1}^{\ell} \|\boldsymbol{v}_i^*\|^2 = \det\Big((\boldsymbol{v}_i \cdot \boldsymbol{v}_j)_{1\le i,j\le \ell} \Big),$$

which shows d_ℓ is a positive *integer*. Hence

$$D = \prod_{\ell=1}^{n} d_\ell \ge 1. \tag{7.61}$$

Hence D is bounded away from 0 by a constant depending only on the dimension of the lattice L, so it can be multiplied by $3/4$ only a finite number of times. This proves that the LLL algorithm terminates.

In order to give an upper bound on the running time, we do some further estimations. Let D_{init} denote the initial value of D for the original basis, let D_{final} denote the value of D for the basis when the LLL algorithm terminates, and as above, let N denote the number of times that the swap step (Step [11]) is executed. (Note that the k loop is executed at most $2N + n$ times, so it suffices to find a bound for N.) The lower bound for D is valid for every basis produced during the execution of the algorithm, so by our earlier results we know that

$$1 \le D_{\text{final}} \le (3/4)^N D_{\text{init}}.$$

Taking logarithms yields (note that $\log(3/4) < 1$)

$$N = \mathcal{O}(\log D_{\text{init}}).$$

To complete the proof, we need to estimate the size of D_{init}. But this is easy, since by the Gram–Schmidt construction we certainly have $\|\boldsymbol{v}_i^*\| \le \|\boldsymbol{v}_i\|$, so

$$D_{\text{init}} = \prod_{i=1}^{n} \|\boldsymbol{v}_i^*\|^{n+1-i} \le \prod_{i=1}^{n} \|\boldsymbol{v}_i\|^{n+1-i} \le \big(\max_{1\le i\le n} \|\boldsymbol{v}_i\| \big)^{2(1+2+\cdots+n)} = B^{n^2+n}.$$

Hence $\log D_{\text{init}} = \mathcal{O}(n^2 \log B)$. □

Remark 7.74. Rather than counting the number of times that the main loop is executed, we might instead count the number of basic arithmetic operations required by LLL. This means counting how many times the internal j-loop is executed and also how many times we perform operations on the coordinates of a vector. For example, adding two vectors or multiplying a vector by a constant is n basic operations. Counted in this way, it is proven in [77] that the LLL algorithm (if efficiently implemented) terminates after no more than $\mathcal{O}\big(n^6 (\log B)^3\big)$ basic operations.

Example 7.75. We illustrate the LLL algorithm on the 6-dimensional lattice L with (ordered) basis given by the rows of the matrix

$$M = \begin{pmatrix} 19 & 2 & 32 & 46 & 3 & 33 \\ 15 & 42 & 11 & 0 & 3 & 24 \\ 43 & 15 & 0 & 24 & 4 & 16 \\ 20 & 44 & 44 & 0 & 18 & 15 \\ 0 & 48 & 35 & 16 & 31 & 31 \\ 48 & 33 & 32 & 9 & 1 & 29 \end{pmatrix}.$$

The smallest vector in this basis is $\|\boldsymbol{v}_2\| = 51.913$.

The output from LLL is the basis consisting of the rows of the matrix

$$M^{LLL} = \begin{pmatrix} 7 & -12 & -8 & 4 & 19 & 9 \\ -20 & 4 & -9 & 16 & 13 & 16 \\ 5 & 2 & 33 & 0 & 15 & -9 \\ -6 & -7 & -20 & -21 & 8 & -12 \\ -10 & -24 & 21 & -15 & -6 & -11 \\ 7 & 4 & -9 & -11 & 1 & 31 \end{pmatrix}.$$

We check that both matrices have the same determinant,

$$\det(M) = \det(M^{LLL}) = \pm 777406251.$$

Further, as expected, the LLL reduced matrix has a much better (i.e., larger) Hadamard ratio than the original matrix,

$$\mathcal{H}(M) = 0.46908 \quad \text{and} \quad \mathcal{H}(M^{LLL}) = 0.88824,$$

so the vectors in the LLL basis are more orthogonal. (The Hadamard ratio is defined in Remark 7.27.) The smallest vector in the LLL reduced basis is $\|\boldsymbol{v}_1\| = 26.739$, which is a significant improvement over the original basis. This may be compared with the Gaussian expected shortest length (Remark 7.32) of $\sigma(L) = (3! \det L)^{1/3}/\sqrt{\pi} = 23.062$.

The LLL algorithm executed 19 swap steps (Step [11] in Fig. 7.8). The sequence of k values from start to finish was

$$2,\ 2,\ 3,\ 2,\ 3,\ 4,\ 3,\ 2,\ 2,\ 3,\ 4,\ 5,\ 4,\ 3,\ 2,\ 3,\ 4,\ 5,\ 4,\ 3,\ 4,\ 5,\ 6,\ 5,$$
$$4,\ 3,\ 4,\ 5,\ 6,\ 5,\ 4,\ 3,\ 2,\ 2,\ 3,\ 2,\ 3,\ 4,\ 5,\ 6.$$

Notice how the algorithm almost finished twice (it got to $k = 6$) before finally terminating the third time. This illustrates how the value of k moves up and down as the algorithm proceeds.

We next reverse the order of the rows of M and apply LLL. Then LLL executes only 11 swap steps and gives the basis

$$M^{LLL} = \begin{pmatrix} -7 & 12 & 8 & -4 & -19 & -9 \\ 20 & -4 & 9 & -16 & -13 & -16 \\ -28 & 11 & 12 & -9 & 17 & -14 \\ -6 & -7 & -20 & -21 & 8 & -12 \\ -7 & -4 & 9 & 11 & -1 & -31 \\ 10 & 24 & -21 & 15 & 6 & 11 \end{pmatrix}.$$

We find the same smallest vector, but the Hadamard ratio $\mathcal{H}(M^{LLL}) = 0.878973$ is a bit lower, so the basis isn't quite as good. This illustrates the fact that the output from LLL is dependent on the order of the basis vectors.

We also ran LLL with the original matrix, but using 0.99 instead of $\frac{3}{4}$ in the Lovász Step [8]. The algorithm did 22 swap steps, which is more than the 19 swap steps required using $\frac{3}{4}$. This is not surprising, since increasing the constant makes the Lovász condition more stringent, so it is harder for the algorithm to get to the k-increment step. Using 0.99, the LLL algorithm returns the basis

$$M^{LLL} = \begin{pmatrix} -7 & 12 & 8 & -4 & -19 & -9 \\ -20 & 4 & -9 & 16 & 13 & 16 \\ 6 & 7 & 20 & 21 & -8 & 12 \\ -28 & 11 & 12 & -9 & 17 & -14 \\ -7 & -4 & 9 & 11 & -1 & -31 \\ -10 & -24 & 21 & -15 & -6 & -11 \end{pmatrix}.$$

Again we get the same smallest vector, but now the basis has $\mathcal{H}(M^{LLL}) = 0.87897$. This is actually slightly worse than the basis obtained using $\frac{3}{4}$, again illustrating the unpredictable dependence of the LLL algorithm's output on its parameters.

7.13.3 Using LLL to Solve apprCVP

We explained in Sect. 7.6 that if a lattice L has an orthogonal basis, then it is very easy to solve both SVP and CVP. The LLL algorithm does not return an orthogonal basis, but it does produce a basis in which the basis vectors are *quasi-orthogonal*, i.e., they are reasonably orthogonal to one another. Thus we can combine the LLL algorithm (Fig. 7.8) with Babai's algorithm (Theorem 7.34) to form an algorithm that solves apprCVP.

Theorem 7.76 (LLL apprCVP Algorithm)**.** *There is a constant C such that for any lattice L of dimension n given by a basis $\boldsymbol{v}_1, \ldots, \boldsymbol{v}_n$, the following algorithm solves* apprCVP *to within a factor of C^n.*

Apply LLL to $\boldsymbol{v}_1, \ldots, \boldsymbol{v}_n$ to find an LLL reduced basis.
Apply Babai's algorithm using the LLL reduced basis.

Proof. We leave the proof for the reader; see Exercise 7.52. □

Remark 7.77. In [8], Babai suggested two ways to use LLL as part of an apprCVP algorithm. The first method uses the closest vertex algorithm that we described in Theorem 7.34. The second method uses the closest plane algorithm. Combining the closest plane method with an LLL reduced basis tends to give a better result than using the closest vertex method. See Exercise 7.53 for further details.