1 Introduction

In this lecture, we focus on computational problems in the field of lattice theory. These problems are often variations on finding a short lattice in a given lattice or finding a lattice point near a given target point. Also, two algorithms are treated that solve these problems in a 'approximate' sense. Those are the Babai Nearest-Plane algorithm – also known as *size reduction* – and the famous Lenstra-Lenstra-Lovàsz algorithm, which is more well-known by its acronym LLL.

2 Computational problems over lattices

Definition 1 (SVP variations) *Given a basis* $\mathbf{B}$ *of a lattice* $\Lambda \subseteq \mathbb{R}^n$

- ***(SVP)*** *Shortest Vector Problem: Find some* $\mathbf{v} \in \Lambda$ *such that* $\|\mathbf{v}\| = \lambda_1(\Lambda)$.
- ***(α-SVP)*** *Approximate Shortest Vector Problem: Find some* $\mathbf{v} \in \Lambda \backslash \{0\}$ *such that* $\|\mathbf{v}\| \leq \alpha \cdot \lambda_1(\Lambda)$.
- ***(r-HermiteSVP)*** *Hermite Approximate Shortest Vector Problem:*
  Find some $\mathbf{v} \in \Lambda \backslash \{0\}$ *such that* $\|\mathbf{v}\| \leq r \cdot \det(\Lambda)^{1/n}$.

The parameter $\alpha \geq 1$ in α-SVP is called the *approximation factor*. The bigger this factor, the easier α-SVP becomes. For $\alpha = 1$, α-SVP and SVP are the same problem. The problem r-HermiteSVP is very similar to α-SVP except for the fact that it measures the length of the vector v with respect to the (n-th root of the) determinant of Λ instead of $\lambda_1(\Lambda)$, the length of the shortest vector.

The following definition is about the closest vector problem. Given a *target* $t \in \text{span}(\mathbf{B})$, this problem asks to find the closest vector $\mathbf{v} \in \Lambda$ to $\mathbf{t}$. Note that $\mathbf{t}$ is only in the linear span of the lattice, and generally is not in the lattice Λ itself.

Definition 2 (CVP variations) *Given a basis* $\mathbf{B}$ *of a lattice* $\Lambda \subseteq \mathbb{R}^n$ *and given a target* $\mathbf{t} \in \text{span}(\mathbf{B})$.

- ***(CVP)*** *Closest Vector Problem:*
  Find some $\mathbf{v} \in \Lambda$ *such that* $\|\mathbf{v} - \mathbf{t}\|$ *is minimized. I.e.* $\|\mathbf{v} - \mathbf{t}\| = d(\Lambda, \mathbf{t})$.
- ***(α-CVP)*** *α-Approximate Closest Vector Problem (Relative):*
  Find some $\mathbf{v} \in \Lambda$ *such that* $\|\mathbf{v} - \mathbf{t}\| \leq \alpha \cdot d(\Lambda, \mathbf{t})$.
- ***(r-AbsCVP)*** *r-Approximate Closest Vector Problem (Absolute):*
  Find some $\mathbf{v} \in \Lambda$ *such that* $\|\mathbf{v} - \mathbf{t}\| \leq r$.

Again, the parameter $\alpha \geq 1$ in α-CVP plays essentially the same rôle as in the α-SVP; to relax the problem somewhat.

One could ask why the r-absolute approximate closest vector problem needs a separate definition; is it not just a rephrasing of the relative approximate closest vector problem with $\alpha = r/d(\Lambda, \mathbf{t})$? The answer is: Yes, it is. But often, $d(\Lambda, \mathbf{t})$ is not known explicitly. This makes it hard to check whether some $\mathbf{v} \in \Lambda$ is really a solution or not. Having a fixed 'absolute' value r to compare $\|\mathbf{v} - \mathbf{t}\|$ with, overcomes this issue. That is why the r-AbsCVP has a separate definition.

The upcoming definition is one of a *promise problem*. In a promise problem the input is promised to satisfy certain requirements (hence the name). For those who aren't convinced that this really alters the problem, let's sketch the following example.

Exercise 1 Suppose you are given a large number $n \in \mathbb{N}$, and you are asked to factorize this number into prime factors. How would you solve this? Let's add a certain promise on n: any prime p that divides n satisfies $p \leq 100$. How would you solve this problem now?

Definition 3 (BDD variations) *Given a basis* $\mathbf{B}$ *of a lattice* $\Lambda \subseteq \mathbb{R}^n$ *and given a target* $\mathbf{t} \in \text{span}(\mathbf{B})$. *Let* $r \leq \frac{1}{2}\lambda_1(\Lambda)$ *and* $\alpha \leq \frac{1}{2}$.

- ***(r-AbsBDD)*** *r-Approximate Bounded Distance Decoding (Absolute):*
  Promised: $d(\Lambda, \mathbf{t}) \leq r$.
  Find the unique $\mathbf{v} \in \Lambda$ *such that* $\|\mathbf{v} - \mathbf{t}\| \leq r$
- ***(α-BDD)*** *α-Approximate Bounded Distance Decoding (Relative):*
  Promised: $d(\Lambda, \mathbf{t}) \leq \alpha\lambda_1(\Lambda)$.
  Find the unique $\mathbf{v} \in \Lambda$ *such that* $\|\mathbf{v} - \mathbf{t}\| \leq \alpha \cdot \lambda_1(\Lambda)$.

As you can see, the BDD-variants are nothing more than their respective CVP-variants with a promise on the input target $\mathbf{t}$. This promise consists of the target point $\mathbf{t}$ being *very* close to the lattice Λ. This closeness is measured by the parameter r (or α, in the relative version).

Despite the familiarity between BDD and CVP, they fundamentally differ at the following point. The problem α-BDD becomes *easier* whenever α decreases; whereas α-CVP becomes *harder* whenever α decreases.

Exercise 2 Show that the word 'unique' in Definition 3 is justified, i.e., show that there exists only one $\mathbf{v} \in V$ with $\|\mathbf{v} - \mathbf{t}\| \leq r$.

3 Babai Nearest-Plane algorithm

Definition 4 *Given a basis* $\mathbf{B}$ *and let* $\widetilde{\mathbf{B}}$ *be its Gram-Schmidt orthogonalized basis. Denote by* $\mathcal{P}_{sym}(\widetilde{\mathbf{B}})$ *the symmetric parallelepiped* $\widetilde{\mathbf{B}}[-1/2, 1/2)^n$ *associated to* $\widetilde{\mathbf{B}}$, *or the symmetric Babai fundamental domain of* $\mathbf{B}$.

The following algorithms attempts to solve α-BDD and α'-CVP; so it tries to find a vector close to some target point $\mathbf{t}$. It outputs some vector $\mathbf{v} \in \Lambda$ and an error $\mathbf{e}$ in the symmetric Babai fundamental domain.

Proof: (of the correctness of algorithm 1)
Clearly, $v + e = t$ and $v \in \Lambda$ at any point in the algorithm. So this leaves us to show that $e \in \mathcal{P}_{sym}(\widetilde{\mathbf{B}})$ after the algorithm finishes. As $\widetilde{\mathbf{B}}$ is a basis, we can always write $e = \sum_i c_i \widetilde{\mathbf{b}}_i$ for some $c_i \in \mathbb{R}$. By orthogonality, we have $\langle e, \widetilde{\mathbf{b}}_i \rangle = c_i \|\widetilde{\mathbf{b}}_i\|^2$ and therefore it is enough to show that $|c_i| = \frac{\langle e, \widetilde{\mathbf{b}}_i \rangle}{\|\widetilde{\mathbf{b}}_i\|^2} \leq \frac{1}{2}$ for all i.

But when $i = j$, we can show that $\frac{\langle e, \widetilde{\mathbf{b}}_j \rangle}{\|\widetilde{\mathbf{b}}_j\|^2} \leq \frac{1}{2}$. Since all operations after this step only alter e by vectors orthogonal to $\widetilde{\mathbf{b}}_j$, this inner product doesn't change.

□

Algorithm 1: Babai Nearest-Plane algorithm

Input : A basis $\mathbf{B} = (\mathbf{b}_1, \ldots, \mathbf{b}_n)$ of a lattice Λ and a target $t \in \text{span}(\Lambda)$.
Output: $(\mathbf{v}, \mathbf{e})$ such that $\mathbf{v} + \mathbf{e} = \mathbf{t}$, $\mathbf{v} \in \Lambda$ and $\mathbf{e} \in \mathcal{P}_{sym}(\widetilde{\mathbf{B}})$.

$\mathbf{e} := \mathbf{t}$
$\mathbf{v} := 0$
for $i = n$ *down to* 1 **do**
  $k := \lceil \frac{\langle \mathbf{e}, \widetilde{\mathbf{b}}_i \rangle}{\|\widetilde{\mathbf{b}}_i\|^2} \rfloor$
  $\mathbf{e} := \mathbf{e} - k\mathbf{b}_i$
  $\mathbf{v} := \mathbf{v} + k\mathbf{b}_i$
end
return $(\mathbf{v}, \mathbf{e})$

Exercise 3 Show that Babai Nearest-Plane algorithm solves r-AbsBDD for any $r \leq \frac{1}{2} \min_i \|\widetilde{\mathbf{b}}_i\|$.

Exercise 4 Show that Babai Nearest-Plane algorithm solves r-AbsCVP for any $r \geq \frac{1}{2} \sqrt{\sum_{i=1}^{n} \|\widetilde{\mathbf{b}}_i\|^2}$.

4 Hermite's Bound and the LLL-algorithm

4.1 Hermite's Bound

Theorem 5 (Hermite) $\gamma_n \leq \gamma_2^{n-1}$.

One could reasonably ask why this theorem is stated here, as we already proved an asymptotically much stronger bound on γ_n using Minkowski's convex body theorem. The first reason why this theorem is mentioned is because of the historic context[1]; before Minkowski's theorem, Hermite's bound was the best known. The second and most important reason why this theorem is treated here, is because of its similarities with a famous basis reduction algorithm: the LLL algorithm. Some consider LLL as an 'algorithmization' of Hermite's bound.

We will soon prove Hermite's theorem 'by algorithm', see algorithm 2.

Definition 6 *Let $\Lambda = \mathcal{L}(\mathbf{B})$ a lattice generated by the basis $\mathbf{B}$. We will denote by $\mathbf{B}_{i:j}$ the basis $(\pi_i(\mathbf{b}_i), \ldots, \pi_i(\mathbf{b}_j))$.*

Definition 7 *Let $\Lambda = \mathcal{L}(\mathbf{B})$ a lattice generated by the basis $\mathbf{B}$. We will denote $\Lambda_{i:j}$ for the lattice generated by $\mathbf{B}_{i:j} = (\pi_i(\mathbf{b}_i), \ldots, \pi_i(\mathbf{b}_j))$, i.e.,*

$$\Lambda_{i:j} = \mathcal{L}(\pi_i(\mathbf{b}_i), \ldots, \pi_i(\mathbf{b}_j)).$$

Exercise 5 Verify for yourself that $\Lambda_{1:n} = \Lambda$ and $\Lambda_{i:n} = \Lambda_i$.

Exercise 6 Show that $\widetilde{\mathbf{B}}_{i:j} = (\widetilde{\mathbf{B}}_{i:k} | \widetilde{\mathbf{B}}_{k+1:j})$ for any $i \leq k \leq j$.

Lemma 8 $\det(\Lambda_{i:j}) = \prod_{k=i}^{j} \|\widetilde{\mathbf{b}}_k\|$.

Algorithm 2: Hermite reduction algorithm

Input : A basis $\mathbf{B} = (\mathbf{b}_1, \ldots, \mathbf{b}_n)$ of a lattice Λ.
Output: A basis $\mathbf{B}$ such that $\|\mathbf{b}_1\|^2 \leq \gamma_2^{n-1} \cdot \det(\Lambda)^{2/n}$.

while $\exists i$ *such that* $\|\widetilde{\mathbf{b}}_i\| > \gamma_2 \|\widetilde{\mathbf{b}}_{i+1}\|$ **do**
  Find matrix $\mathbf{U} \in \mathbb{Z}^{2\times 2}$ such that $\mathbf{B}_{i:i+1}\mathbf{U}$ is Lagrange reduced
  Set $(\mathbf{b}'_i, \mathbf{b}'_{i+1}) = (\mathbf{b}_i, \mathbf{b}_{i+1})\mathbf{U}$
  Set $\mathbf{B} = (\mathbf{b}_1, \ldots, \mathbf{b}_{i-1}, \mathbf{b}'_i, \mathbf{b}'_{i+1}, \mathbf{b}_{i+2}, \ldots, \mathbf{b}_n)$
end
return $\mathbf{B}$

Proof: The proof is left as an exercise to the reader. □

Proof: (of Hermite's theorem)
First, we prove the correctness of algorithm 2, if it terminates. After that, we will show that the algorithm does terminate indeed. Lastly, we see why the correctness and termination of the Hermite reduction algorithm implies Hermite's bound.

Correctness If the algorithm terminates, any pair of basis vectors $\widetilde{\mathbf{b}}_i, \widetilde{\mathbf{b}}_{i+1}$ satisfies $\|\widetilde{\mathbf{b}}_i\| \leq \gamma_2 \|\widetilde{\mathbf{b}}_{i+1}\|$. Using inductive reasoning, one can conclude that $\|\widetilde{\mathbf{b}}_1\| = \|\widetilde{\mathbf{b}}_1\| \leq \gamma_2^{i-1} \|\widetilde{\mathbf{b}}_i\|$. Multiplying together for all i and using the determinant formula and the triangular number formula, yields

$$\|\mathbf{b}_1\|^n \leq \prod_{i=1}^{n} \gamma_2^{i-1} \|\widetilde{\mathbf{b}}_i\| = \gamma_2^{\frac{n(n-1)}{2}} \det(\Lambda).$$

Taking $n/2$-th roots shows that the output of the Hermite reduction algorithm – if it indeed terminates – satisfies the requirements.

Termination Before proving termination of the algorithm, we would like to show that Lagrange-reduction on a pair of vectors $(\mathbf{b}_i, \mathbf{b}_{i+1})$ indeed leads to inequality $\|\widetilde{\mathbf{b}}_i\| \leq \gamma_2 \|\widetilde{\mathbf{b}}_{i+1}\|$. After Lagrange-reduction, we have $\|\widetilde{\mathbf{b}}_i\|^2 \leq \gamma_2 \cdot \det(\mathbf{B}_{i:i+1}) = \gamma_2 \|\widetilde{\mathbf{b}}_i\| \|\widetilde{\mathbf{b}}_{i+1}\|$, where the last equality comes from Lemma 8. Dividing out appropriately yields $\|\widetilde{\mathbf{b}}_i\| \leq \gamma_2 \|\widetilde{\mathbf{b}}_{i+1}\|$. So Lagrange-reduction indeed 'resolves' the wrong-way inequality $\|\widetilde{\mathbf{b}}_i\| > \gamma_2 \|\widetilde{\mathbf{b}}_{i+1}\|$.

To prove that the algorithm terminates one can use an induction argument. Let us assume, by hypothesis, that the Hermite reduction algorithm always terminates on lattices with dimension smaller than n. We will prove that this algorithm also terminates on lattices with dimension precisely n.

To show that, we need a few claims.

- The norm of $\mathbf{b}_1$ doesn't change if a Lagrange reduction doesn't involve $\mathbf{b}_1$. This is obviously true, because then $\mathbf{b}_1$ is not affected and the norm stays the same.
- When a Lagrange reduction *does* involve $\mathbf{b}_1$, it will replace $\mathbf{b}_1$ with a new one with strictly smaller norm. This is true by the following reasoning. Because the Hermite reduction algorithm only applies Lagrange reduction whenever $\|\mathbf{b}_1\| > \gamma_2 \|\widetilde{\mathbf{b}}_2\|$, we know that $\|\mathbf{b}_1\|^2 >$

[1] Minkowski (1864-1909) established his convex body theorem in 1891; Hermite (1822-1904) stated his bound around 1850.

$\gamma_2\|\widetilde{\mathbf{b}}_2\|\|\mathbf{b}_1\| = \gamma_2 \cdot \det(\Lambda_{1:2})$ *before* the Lagrange reduction happens. However, after Lagrange reduction, we have $\|\mathbf{b}_1\| \leq \gamma_2 \cdot \det(\Lambda_{1:2})$. So the new $\mathbf{b}_1$ has indeed a strictly smaller norm.

- $\|\mathbf{b}_1\|$ has a lower bound. Namely, $\lambda_1(\Lambda) \leq \|\mathbf{b}_1\|$.

Since Λ is a discrete subset of $\mathbb{R}^n$, the norm of $\mathbf{b}_1$ has a lower bound and the norm is decreasing during the algorithm, this norm must eventually stabilize. This means that no Lagrange-reduction involving $\mathbf{b}_1$ happens after that stabilization.

Therefore, after this stabilization, the algorithm takes place in $\Lambda_{2:n}$ alone (because no operations on $\mathbf{b}_1$ are done anymore). By the induction hypothesis, it must terminate. Thus, the entire algorithm on Λ terminates, too.

Implication of Hermite's bound Lastly, the correctness and termination of this algorithm implies Hermite's bound; in any lattice of dimension n we can find a vector $\mathbf{b}_1$ whose square norm is bounded by $\gamma_2^{n-1} \cdot \det(\Lambda)^{2/n}$. Thus,

$$\gamma(\Lambda) = \frac{\lambda_1(\Lambda)^2}{\det(\Lambda)^{2/n}} \leq \frac{\|\mathbf{b}_1\|^2}{\det(\Lambda)^{2/n}} \leq \frac{\gamma_2^{n-1} \cdot \det(\Lambda)^{2/n}}{\det(\Lambda)^{2/n}} = \gamma_2^{n-1}.$$

Therefore, $\gamma_n = \sup_\Lambda \gamma(\Lambda) \leq \gamma_2^{n-1}$. □

4.2 Weakly LLL-reduced bases

The issue with Hermite's algorithm is that it may be terribly slow. The idea of the Lenstra-Lenstra-Lovász algorithm consists in slightly relaxing the termination condition to make the algorithm faster. Below, we present a weak version of the notion of LLL-reduced basis, which is sufficient to show that the number of iterations in the algorithm is polynomial. However, the size of the numbers occurring in this computation is not controlled; it might still be possible that the numbers involved in the computation are too big to efficiently compute with. The true LLL algorithm, which will be presented in the next section, resolves this problem.

Definition 9 (ε-Weakly LLL reduced) $\mathbf{B}$ *is said to be ε-WLLL reduced if*

$$\|\widetilde{\mathbf{b}}_i\| \leq (\gamma_2 + \varepsilon)\|\widetilde{\mathbf{b}}_{i+1}\| \quad \textit{for all } i.$$

For a fixed i, this is known as the Lovàsz condition *on* $(\widetilde{\mathbf{b}}_i, \widetilde{\mathbf{b}}_{i+1})$.

Not that above algorithm only pays for a Lagrange reduction whenever the norm of $\widetilde{\mathbf{b}}_i$ is significantly larger than what it could be. This 'significant amount' is measured by $\varepsilon > 0$. So, in fact, above algorithm is exactly the Hermite reduction algorithm, with the sole difference that it adds some 'slack' in the form of the parameter ε.

Theorem 10 *If $\mathbf{B} \in \mathbb{Z}^{n \times n}$, then the ε-Weakly LLL algorithm terminates after poly$(n, \log\|\mathbf{B}\|_\infty, 1/\varepsilon)$ iterations.*

Proof: Define $P = \prod_{i=1}^n \det(\Lambda_{1:i})$, to be the 'potential' of a the current basis $\mathbf{B}$ (note that the notation $\Lambda_{1:i}$ indeed hides a dependence in the choice of the basis). We will show that P decreases

Algorithm 3: ε-Weakly LLL-reduction algorithm

Input : A basis $\mathbf{B} = (\mathbf{b}_1, \ldots, \mathbf{b}_n)$ of a lattice Λ.
Output: A basis $\mathbf{B}$ such that $\|\mathbf{b}_1\|^2 \leq (\gamma_2 + \varepsilon)^{n-1} \cdot \det(\Lambda)^{2/n}$.

while $\exists i$ *such that* $\|\widetilde{\mathbf{b}}_i\| > (\gamma_2 + \varepsilon)\|\widetilde{\mathbf{b}}_{i+1}\|$ **do**
  Find matrix $\mathbf{U} \in \mathbb{Z}^{2\times 2}$ such that $\mathbf{B}_{i:i+1}\mathbf{U}$ is Lagrange reduced
  Set $(\mathbf{b}'_i, \mathbf{b}'_{i+1}) = (\mathbf{b}_i, \mathbf{b}_{i+1})\mathbf{U}$
  Set $\mathbf{B} = (\mathbf{b}_1, \ldots, \mathbf{b}_{i-1}, \mathbf{b}'_i, \mathbf{b}'_{i+1}, \mathbf{b}_{i+2}, \ldots, \mathbf{b}_n)$
end
return $\mathbf{B}$

by a constant factor in each iteration and that P has a lower bound. This combined shows that the algorithm terminates within a polynomially bounded number of iterations.

One application of Lagrange on $\mathbf{B}_{i:i+1}$ in an iteration only changes $\det(\Lambda_{1:i})$ and keeps all other determinants in the product of the potential fixed. Before this iteration $\|\widetilde{\mathbf{b}}_i\|^2 \geq (\gamma_2 + \varepsilon)\det(\Lambda_{i:i+1})$. After the Lagrange reduction we must have $\|\widetilde{\mathbf{b}}_i\|^2 \leq \gamma_2 \cdot \det(\Lambda_{i:i+1})$. Therefore, after this Lagrange reduction, $\det(\Lambda_{1:i}) = \prod_{k=1}^{i}\|\widetilde{\mathbf{b}}_k\|$ must at least be diminished by a factor $\sqrt{\frac{\gamma_2}{\gamma_2+\varepsilon}}$. Thus, also the potential P must be reduced by (at least) this factor $f = \sqrt{\frac{\gamma_2}{\gamma_2+\varepsilon}}$.

As $\Lambda_{1:i} \subseteq \mathbb{Z}^n$, we must have that $\det(\Lambda_{1:i}) \in \mathbb{Z}\backslash\{0\}$. This has as a direct consequence that $P \geq 1$; at any time in the algorithm, P must be larger than or equal to one.

The initial potential P_{init} (that is, the potential before running the algorithm) is bounded by $\prod_{i=1}^{n}\|\mathbf{b}_i\| = \|\mathbf{B}\|_\infty^{n(n+1)/2}$, where $\|\mathbf{B}\|_\infty = \max_i\|\mathbf{b}_i\|$. Let N be the number of iterations, then we know that

$$1 \leq P < f^N \cdot P_{init} \leq f^N \cdot \|\mathbf{B}\|_\infty^{n(n+1)/2}$$

This means that the algorithm is surely terminated whenever $f^N \cdot \|\mathbf{B}\|_\infty^{n(n+1)/2} \leq 1$, i.e. whenever $N\log(f) + \frac{n(n+1)}{2}\log(\|\mathbf{B}\|_\infty) \leq 0$. Reshaping the formula yields that the algorithm is necessarily ended when

$$N \geq \frac{n(n+1)\log(\|\mathbf{B}\|_\infty)}{-2 \cdot \log(f)}$$

So, the number of iterations is bounded by $\frac{n(n+1)\log(\|\mathbf{B}\|_\infty)}{-2\cdot\log(f)}$. Using the fact that $-\log(f) = 1/2 \cdot \log(1 + \varepsilon/\gamma_2) = \frac{\varepsilon}{2\gamma_2} + O(\varepsilon^2)$ yields the result. □

However, it is not enough to prove that the number of iterations is bounded, in order to show that algorithm 3 is truly poly-time. The time consumed computing with the (rational) numbers in the algorithm should also be taken in consideration; without more analysis we don't know whether the rational numbers occurring in $\widetilde{\mathbf{B}}$ have very large numerators and denominators (which might slow down the overall computation drastically). Also, we don't know whether the size of the integer coefficients of $\mathbf{B}$ can be controlled. We will see that bounding the integer coefficients requires a modification of the algorithm (namely, the 'real' LLL algorithm), which will be treated in the next section.

The denominators of the entries in $\widetilde{\mathbf{B}}$ are bounded by $\det(\mathcal{L}(\mathbf{B}))^2$, which follows from the following lemma.

LEMMA 11 *If* $\mathbf{B} \in \mathbb{Z}^{n\times n}$, *we have* $\widetilde{\mathbf{B}} \in \frac{1}{\det(\mathbf{B}\cdot\mathbf{B}^T)}\mathbb{Z}^{n\times n}$.

PROOF: We prove this by induction on the number of basis vectors. Let $\mathbf{B} = (\mathbf{b}_1, \ldots, \mathbf{b}_n)$ be a basis. Write $\mathbf{C} = (\pi_2(\mathbf{b}_1), \ldots, \pi_2(\mathbf{b}_n))$, a projected basis. In order to apply the induction hypothesis, we want something that is in $\mathbb{Z}^{n\times(n-1)}$. But we know that $\pi_2(\mathbf{v}) = \mathbf{v} - \frac{\langle \mathbf{v}, \mathbf{b}_1\rangle}{\|\mathbf{b}_1\|^2} \cdot \mathbf{b}_1 \in \frac{1}{\|\mathbf{b}_1\|^2}\mathbb{Z}^n$. Write $d = \|\mathbf{b}_1\|^2$. Then $d\mathbf{C} \in \mathbb{Z}^{n\times(n-1)}$, so we can apply the induction hypothesis:

$$(d\widetilde{\mathbf{C}}) = d\widetilde{\mathbf{C}} \in \frac{1}{\det(\mathbf{C}\mathbf{C}^T)}\mathbb{Z}^{n\times(n-1)}, \text{ i.e. } \widetilde{\mathbf{C}} \in \frac{1}{d\cdot\det(\mathbf{C}\mathbf{C}^T)}\mathbb{Z}^{n\times(n-1)}$$

Now use that $\mathbf{C}$ is full rank, and therefore $\det(\mathbf{C}\mathbf{C}^T) = \prod_{i=2}^n \|\widetilde{\mathbf{b}}_i\|^2$. Since $\widetilde{\mathbf{B}} = (b_1|\widetilde{\mathbf{C}})$ and $\det(\mathbf{B}\mathbf{B}^T) = \|\mathbf{b}_1\|^2 \cdot \det(\mathbf{C}\mathbf{C}^T) = d \cdot \det(\mathbf{C}\mathbf{C}^T)$, we have

$$\widetilde{\mathbf{B}} = (b_1|\widetilde{\mathbf{C}}) \in \frac{1}{\det(\mathbf{B}\mathbf{B}^T)}\mathbb{Z}^{n\times n}$$

□

5 LLL-reduced bases

DEFINITION 12 *A basis of a lattice* $\mathbf{B}$ *is said* size reduced *when*

$$\forall i < j \quad |\langle \widetilde{\mathbf{b}}_i, \mathbf{b}_j\rangle| \leq \frac{1}{2}\|\widetilde{\mathbf{b}}_i\|^2$$

Above definition is equivalent to saying that the off-diagonal of the so-called μ-matrix of the Gram-Schmidt orthogonalization is bounded by a half, that is, $|\mu_{ij}| \leq 1/2$ for all $i \neq j$.

DEFINITION 13 *A basis* $\mathbf{B}$ *is ε-LLL reduced if it is both ε-WLLL reduced and size reduced.*

LEMMA 14 *A size-reduced basis* $\mathbf{B}$ *of a lattice* $\Lambda \subseteq \mathbb{Z}^n$ *has basis vectors* $\mathbf{b}_i$ *satisfying* $\log(\|\mathbf{b}_i\|), \log(\|\widetilde{\mathbf{b}}_i\|) \leq$ *poly*$(n) \cdot \log(\det(\Lambda))$.

PROOF: As $\mathbf{B} \in \mathbb{Z}^{n\times n}$, we have $\prod_{i=1}^n \|\widetilde{\mathbf{b}}_i\| = \det(\Lambda) \geq 1$, which in turn means that $\|\widetilde{\mathbf{b}}_i\| \geq 1/\det(\Lambda)$. The last inequality follows from distinguishing the cases $\|\widetilde{\mathbf{b}}_i\| < 1$ and ≥ 1. The last case is trivial, whereas the first case follows from $\|\widetilde{\mathbf{b}}_i\| < \|\widetilde{\mathbf{b}}_i\|\det(\Lambda) < \det(\Lambda)$. This, by the determinant formula, immediately yields $\|\widetilde{\mathbf{b}}_i\| \leq \det(\Lambda)^{n-1}$.

As the basis is size-reduced, we can write any basis vector $\mathbf{b}_j = \sum_{i=1}^n c_i\widetilde{\mathbf{b}}_i$ with $|c_i| \leq 1/2$. Therefore $\|\mathbf{b}_j\| \leq \sum_i |c_i|\|\widetilde{\mathbf{b}}_i\| \leq \frac{n}{2}\det(\Lambda)^{n-1}$.

So, both $\log(\|\mathbf{b}_i\|), \log(\|\widetilde{\mathbf{b}}_i\|)$ are bounded by $n^2 \cdot \log(\det(\Lambda))$, which proves the claim. □

Thus, as long as the basis $\mathbf{B}$ is size-reduced during the computation, one can be sure that the coefficients of the basis are not growing too large. Therefore, to have a poly-time basis reduction algorithm, we should have an algorithm similar to the Weakly LLL-reduction algorithm, with the extra requirement that, in the meantime, the basis is always size-reduced. This is roughly what the LLL algorithm does.

Algorithm 4: Size-reduction algorithm

Input : A basis $\mathbf{B} = (\mathbf{b}_1, \ldots, \mathbf{b}_n)$ of a lattice Λ.
Output: A size-reduced basis $\mathbf{B}$, i.e., $\forall i < j \quad |\langle \widetilde{\mathbf{b}}_i, \mathbf{b}_j \rangle| \leq \frac{1}{2}\|\widetilde{\mathbf{b}}_i\|^2$

Set $\mathbf{B}' = (\mathbf{b}_1, \ldots, \mathbf{b}_{n-1})$.
Size-reduce $\mathbf{B}'$ recursively.
Apply the Babai Nearest-Plane algorithm with basis $\mathbf{B}'$ and target $\mathbf{b}_n$.
Return $(\mathbf{B}'|\mathbf{e})$, where $\mathbf{e}$ is the output of the Babai Nearest Plane algorithm.

5.1 The size-reduction algorithm

The size-reduction algorithm (algorithm 4) uses the Babai Nearest-Plane with a target which is not in the span of $\mathbf{B}'$. One can easily prove that applying the BNP-algorithm on $\mathbf{b}_n$ is equivalent to applying it to $\pi(\mathbf{b}_n)$, where π is the map projecting the space $\mathrm{span}(\mathbf{B})$ on $\mathrm{span}(\mathbf{B}')$.

Exercise 7 Let $\mathbf{B} = (\mathbf{B}'|\mathbf{b}_n)$ be a basis of a lattice. Show that $|\langle \widetilde{\mathbf{b}}_i, \mathbf{b}_n \rangle| = |\langle \widetilde{\mathbf{b}}_i, \pi(\mathbf{b}_n) \rangle|$, where π is the map projecting the space $\mathrm{span}(\mathbf{B})$ on $\mathrm{span}(\mathbf{B}')$. [Hint: Show that $\pi = \mathrm{id} - \pi_n$.]

LEMMA 15 *The size-reduction algorithm (algorithm 4) indeed outputs a size-reduced basis as in definition 12.*

PROOF: The design of the size-reduction algorithm makes it easy to prove this claim by induction. As $\mathbf{B}'$ is size-reduced by induction, we only have to check whether for all $i < n$ we have the inequality $|\langle \widetilde{\mathbf{b}}_i, \mathbf{b}_n \rangle| \leq \frac{1}{2}\|\widetilde{\mathbf{b}}_i\|^2$.

Let $\pi : \mathrm{span}(\mathbf{B}) \to \mathrm{span}(\mathbf{B}')$ be the projection map that projects $\mathrm{span}(\mathbf{B})$ on $\mathrm{span}(\mathbf{B}')$. We have

$$|\langle \widetilde{\mathbf{b}}_i, \mathbf{b}_n \rangle| = |\langle \widetilde{\mathbf{b}}_i, \pi(\mathbf{b}_n) \rangle| \leq \frac{1}{2}\|\widetilde{\mathbf{b}}_i\|^2,$$

where the inequality follows from the construction of the Babai nearest-plane algorithm, and the equality follows from the last exercise. □

5.2 The LLL algorithm

Algorithm 5: LLL-reduction algorithm

Input : A basis $\mathbf{B} = (\mathbf{b}_1, \ldots, \mathbf{b}_n)$ of a lattice Λ.
Output: An LLL-reduced basis $\mathbf{B}$

Size-reduce the basis $\mathbf{B}$
while $\exists i$ *such that* $\|\widetilde{\mathbf{b}}_i\| > (\gamma_2 + \varepsilon)\|\widetilde{\mathbf{b}}_{i+1}\|$ **do**
  Lagrange reduce $\mathbf{B}_{i:i+1}$ as in the WLLL algorithm.
  Size-reduce the new basis $\mathbf{B}$.
end
Return $\mathbf{B}$

Exercise 8 Show that size-reduction doesn't affect the potential $P = \prod_{i=1}^{n} \det(\Lambda_{1:i})$, as defined in the proof of theorem 10. Explain why you can use this proof to show that the LLL-reduction algorithm uses only polynomially many iterations.

As we already proved that the denominators of $\widetilde{\mathbf{b}}_i$ stay bounded, we only have to prove that the coefficients of $\mathbf{b}_i$ doesn't grow to big. As we saw that size-reduced bases indeed satisfy this requirement (see lemma 14), we are almost done. The final step to finish the prove that the LLL-algorithm terminates in polynomially bounded time, is showing that Lagrange-reduction will not let grow the basis too much.

LEMMA 16 *Let* $(\mathbf{b}_1, \mathbf{b}_2)$ *two independent vectors. Let* $(\mathbf{b}_1', \mathbf{b}_2') = (\mathbf{b}_1, \mathbf{b}_2)\mathbf{U}$ *be the associated Lagrange-reduced basis.*

Then $\log\|\mathbf{b}_1'\|, \log\|\mathbf{b}_2'\|$ *are polynomially bounded by the logarithms of the norms of the input basis vectors. Moreover, during the Lagrange reduction, the intermediate basis vectors will also satisfy this bound.*

PROOF: In the previous lecture notes, we proved that the Lagrange reduction terminates within $O(\log \frac{\|\mathbf{b}_1\|}{\sqrt{\det\Lambda}})$ iterations. At each iteration, the Lagrange algorithm multiplies the basis by a matrix of the form $\mathbf{U}_0 = \begin{bmatrix} 0 & 1 \\ 1 & -k \end{bmatrix}$ with $k = \lceil \frac{\langle \mathbf{b}_1, \mathbf{b}_2 \rangle}{\|\mathbf{b}_1\|^2} \rfloor$. As $|\langle \mathbf{b}_1, \mathbf{b}_2 \rangle| \leq \|\mathbf{b}_1\|\|\mathbf{b}_2\|$, we have that $|k|$ must be bounded by $\frac{\|\mathbf{b}_2\|}{\|\mathbf{b}_1\|}$. The ℓ_2-matrix norm of such an unimodular $\mathbf{U}_0$ matrix is bounded by $\sqrt{k^2+2} \leq |k| + 2$. Therefore,

$$\|\mathbf{b}_1'\| \leq \|\mathbf{b}_1\|\|\mathbf{U}\| = \|\mathbf{b}_1\|\|\mathbf{U}_0 \cdots \mathbf{U}_r\| \leq \|\mathbf{b}_1\|\|\mathbf{U}_0\| \cdots \|\mathbf{U}_r\|.$$

Here, r is the number of iterations, which is bounded by $O(\log \frac{\|\mathbf{b}_1\|}{\sqrt{\det\Lambda}})$. Taking logarithms, gives us

$$\log\|\mathbf{b}_1'\| \leq \log\|\mathbf{b}_1\| + \log\|\mathbf{U}_0\| + \ldots + \log\|\mathbf{U}_r\| \leq \log\|\mathbf{b}_1\| + O\left(\log \frac{\|\mathbf{b}_1\|}{\sqrt{\det\Lambda}}\right) \cdot \log\left(\frac{\|\mathbf{b}_2\|}{\|\mathbf{b}_1\|} + 2\right),$$

which is clearly polynomially bounded in $\log\|\mathbf{b}_1\|$ and $\log\|\mathbf{b}_2\|$. A similar bound can be established for $\mathbf{b}_2'$. Furthermore, every occurring intermediate basis vector must satisfy this bound, too. □

Combined all the previous lemmata, we arrive at the following result.

THEOREM 17 *The LLL-basis reduction algorithm (as in Algorithm 5) with as input a basis* $\mathbf{B} \subseteq \mathbb{Z}^{n \times n}$ *runs in polynomial time.*