1 Introduction

In this course, the first mathematical objects we will consider are known as *lattices*. What is a lattice? It is a set of points in n-dimensional space with a periodic structure, such as the one illustrated in Figure 1. Three-dimensional lattices occur naturally in crystals, as well as in stacks of oranges. Historically, lattices were investigated since the late 18th century by mathematicians such as Lagrange, Gauss, and later Minkowski.

One of the most elusive problem with lattices (of large dimension) is that of sphere packing: how to stack balls as densely as possible. This question can be found in a booklet of Johannes Kepler *Strena seu de Nive Sexangula* (1611), as a tentative explanation for the shape of snowflakes: the hexagonal structure would stem from optimal packing of spherical atoms. This first lecture will culminate with an upper bound on the density of lattice packing in arbitrary dimension (Minkowski's bound).

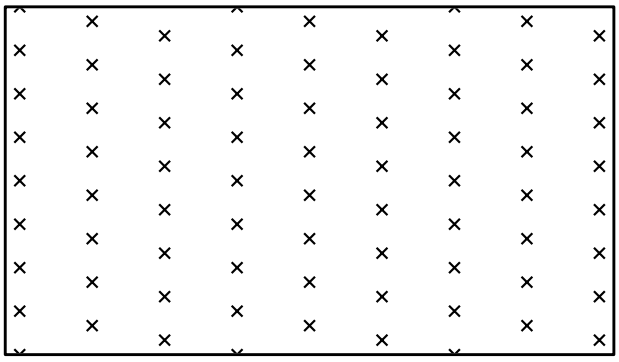

Figure 1: A lattice in $\mathbb{R}^2$

Lattices are a recurring tool in algebra and number theory, for example to prove finiteness of the class group of a number field, or to the efficiently factor rational polynomials.

More recently, lattices have become a topic of active research in computer science. Algorithmic problems based on lattices (e.g. Shortest and Closest Vector Problems, ...) have found a wide variety of applications; they are used within optimization algorithms, in the design of wireless communication protocols, and perhaps the most active research area, in the development of secure cryptographic primitives (cryptography) and in establishing the insecurity of certain cryptographic schemes (cryptanalysis). But lattices and their algorithm find applications in various branch of sciences, such as reconstructing molecule composition from mass spectrometry (chemistry), or detecting pulsars (astronomy).

2 Notation and Basic Concepts

We use $\mathbb{R}$ for the real numbers, $\mathbb{Z}$ for the integers and $\mathbb{N}$ for the natural numbers (positive integers). Correspondingly, we use $\mathbb{R}^n$ and $\mathbb{Z}^n$ to denote the n-dimensional versions for some

$n \in \mathbb{N}$. We write generally matrices as $\mathbf{B}$ in uppercase bold, vectors $\mathbf{x} \in \mathbb{R}^n$ in lowercase bold and scalars as $x \in \mathbb{R}$.

For a set $A \subseteq \mathbb{R}^n$, we use $\mathrm{span}(A)$ to denote its linear span of A, i.e. the smallest linear subspace containing A. We define the dimension $\dim(A)$ of A to be the dimension of the linear span, that is, $\dim(A) := \dim(\mathrm{span}(A))$. For two sets $A, B \subseteq \mathbb{R}^n$, $s, t \in \mathbb{R}$, we define their Minkowski sum $sA + tB := \{s\mathbf{a} + t\mathbf{b} : \mathbf{a} \in A, \mathbf{b} \in B\}$.

3 Definitions and Basic Concepts

The main goal of this lecture is to introduce the basic concept of a lattice, define one of its basic geometric parameters (the shortest non-zero vector), and present various equivalent definitions of a lattice. Our abstract definition of a lattice is given below:

Definition 1 (Lattice) *$\mathcal{L} \subseteq \mathbb{R}^n$ is a lattice if $\mathcal{L}$ is a discrete additive subgroup of $\mathbb{R}^n$. $\mathcal{L}$ is a rank k lattice, or is k-dimensional, if $\dim_{\mathbb{R}}(\mathcal{L}) = k$. $\mathcal{L}$ is said to be full-rank if $\dim_{\mathbb{R}}(\mathcal{L}) = n$ (the dimension of the ambient space).*

Endowment of $\mathbb{R}^n$. For what is to come, $\mathbb{R}^n$ will be endowed with *some* norm denoted $\| \cdot \|$. A norm induces a metric on $\mathbb{R}^n$ which is non-trivial, and any non-trivial metric on $\mathbb{R}^n$ induces the same topology on $\mathbb{R}^n$. Note that a discrete group must be "uniformly discrete", a set S is uniformly discrete in a metric space X if there exists a $\delta > 0$ such that for all $s \in S$, the open ball of radius δ around s contains no other elements of S than s itself.

Another endowment that we will use is its unique Lesbegue measure, denoted $\mathrm{vol}(\cdot)$; for a *measurable* set $S \subset \mathbb{R}^n$, the volume $\mathrm{vol}(S)$ is a non-negative real number or ∞. We recall that a measure is sub-additive under union and additive under disjoint union:

1. $\mathrm{vol}\left(\bigcup_i S_i\right) \leq \sum_i \mathrm{vol}\left(S_i\right)$
2. $\mathrm{vol}\left(\bigsqcup_i S_i\right) = \sum_i \mathrm{vol}\left(S_i\right)$

Being a Lesbegue measure on $\mathbb{R}^n$, it is also invariant by translation, is homogeneous, and is normalized on the canonical hypercube:

1. $\forall \mathbf{x} \in \mathbb{R}^n, \mathrm{vol}(S + \mathbf{x}) = \mathrm{vol}(S)$
2. $\forall \mathbf{T} \in \mathbb{R}^{n \times n}, \mathrm{vol}(\mathbf{T} \cdot S) = |\det(\mathbf{T})| \cdot \mathrm{vol}(S)$
3. $\mathrm{vol}([0,1]^n) = 1$

The second property implies invariance under rotations $\mathbf{T} \in \mathcal{O}_n(\mathbb{R})$. One can easily deduce that a bounded measurable set has finite measure, and that a set with non-empty interior has strictly positive measure.

Issues of measurability shall not arise in this course and will be ignored; most bodies $S \subset \mathbb{R}^n$ we will encounter are convex or are a finite combination of convex bodies.

Remark 2 The term lattice is quite overloaded in the literature, and can also refer to an order, or to subgroup of other groups than $\mathbb{R}^n$. The lattices as defined above are sometimes referred to as "point lattices". The term "Euclidean lattices" is also used when the considered metric is Euclidean.

PROPOSITION 3 *A non-trivial subgroup $L \subset \mathbb{R}^n$ is discrete if and only if it admits a strictly positive minimal distance, that is if $\min_{\mathbf{x}\neq\mathbf{y}\in L} \|\mathbf{x}-\mathbf{y}\|$ is well defined and strictly positive.*

Because a lattice L is a group, its minimum distance can be rewritten as $\min_{\mathbf{x}\in L\setminus\{\mathbf{0}\}} \|\mathbf{x}\|$. It will be denoted $\lambda_1(L)$.

PROOF: Having a minimal distance immediately implies discreteness: around each element of the subgroup L, we can take the open ball of radius $\lambda_1/2$, which does not include any other point in L.

For the other direction, we will show that the minimal distance is attained and is strictly positive. By discreteness and non-triviality of L, the greatest lower bound

$$\lambda_1 \;=\; \inf\{\|\mathbf{x}\| : \mathbf{x} \in L \setminus \{\mathbf{0}\}\}$$

exists and is strictly positive. Consider the open ball B of radius $2\lambda_1$ around the point 0. By definition of λ_1, the ball B will include at least one non-zero element $\mathbf{x} \in L$. This ball is bounded and thus has finite volume $V := \mathrm{vol}(B)$. Moreover, an open ball C of radius $\lambda_1/2$ has a strictly positive volume. For any two $\mathbf{x}, \mathbf{y} \in L \cap B$, the intersection $\mathbf{x}+C \cap \mathbf{y}+C$ is empty, by definition of λ_1. Thus the union

$$U = \bigsqcup_{\mathbf{x}\in B\cap L} \mathbf{x}+C$$

is disjoint, and therefore has volume

$$\mathrm{vol}(U) = \sum_{\mathbf{x}\in B\cap L} \mathrm{vol}(\mathbf{x}+C) = \sum_{\mathbf{x}\in B\cap L} \mathrm{vol}(C) = |B\cap L| \cdot \mathrm{vol}(C).$$

Note that U is also bounded (it is a subset of the open ball D of radius $5\lambda_1/2$) and therefore has finite volume. Therefore, $|B \cap L|$ is finite. This finite set has an element with minimal norm, which by construction is λ_1. Thus the infimum is attained and is thus a minimum. □

Exercise 1 Provide two counterexamples when the either assumption of the above proposition is not met:

- A finitely generated subgroup of $\mathbb{R}$ that is not discrete, and doesn't admit a minimal distance
- A discrete subset (but not uniformly) of $\mathbb{R}$ that is not a group, and doesn't admit a minimal distance

3.1 Bases

DEFINITION 4 (BASIS OF A LATTICE) *A matrix $\mathbf{B} = [\mathbf{b}_1 \dots \mathbf{b}_k] \in \mathbb{R}^{n\times k}$ is a basis of a lattice $L \subset \mathbb{R}^n$ if its column vectors are linearly independent over $\mathbb{R}$ and the $\mathbb{Z}$-span of its columns is exactly L, i.e.*

$$\mathcal{L}(\mathbf{B}) := \mathbf{B}\cdot\mathbb{Z}^k = L.$$

The goal of this section is to prove that all lattices admit a basis. By definition, one can find a set of linearly independent vectors in L. We will show that such a set generates a sublattice of L of finite index. From there, we will reconstruct a basis. For the first step, we resort to the *fundamental parallelepiped*: if $\mathbf{B} \in \mathbb{R}^{m\times k}$ has linearly independent columns we denote $\mathcal{P}(\mathbf{B}) := \mathbf{B}\cdot[-1/2, 1/2)^n$ the parallelepiped spanned by $\mathbf{B}$.

PROPOSITION 5 *For any basis* $\mathbf{B}$ *of* L, $\mathcal{P}(\mathbf{B})$ *is a fundamental domain of* L, *that is any* $\mathbf{t} \in \mathrm{Span}_{\mathbb{R}}(L)$ *can be uniquely written as* $\mathbf{t} = \mathbf{x} + \mathbf{e}$ *where* $\mathbf{x} \in L$ *and* $\mathbf{e} \in \mathcal{P}(\mathbf{B})$.

PROOF: Let $\mathbf{t} \in \mathrm{Span}_{\mathbb{R}}(L)$ be written as a linear combination

$$\mathbf{t} = c_1\mathbf{b}_1 + \ldots + c_k\mathbf{b}_k,$$

where $k > 0$ is the dimension of the lattice L and each $c_i \in \mathbb{R}$. Consider the rounding function $\lfloor \cdot \rceil : \mathbb{R} \to \mathbb{Z}$ that rounds a real number x to the largest integer not greater than $x + 0.5$ (that is, the familiar rounding function where half-integers are rounded up to the next integer). Set $\mathbf{x} := \lfloor c_1 \rceil \mathbf{b}_1 + \ldots + \lfloor c_k \rceil \mathbf{b}_k$. This is a $\mathbb{Z}$-linear combination of the basis vectors in $\mathbf{B}$, so $\mathbf{x} \in L$. Define

$$\mathbf{e} := \mathbf{t} - \mathbf{x} = (c_1 - \lfloor c_1 \rceil)\mathbf{b}_1 + \ldots + (c_k - \lfloor c_k \rceil)\mathbf{b}_k.$$

Each $c_i - \lfloor c_i \rceil \in [-1/2, 1/2)$, so $\mathbf{e} \in \mathcal{P}(\mathbf{B})$.

It remains to show uniqueness. Suppose we have two different such representations $\mathbf{t} = \mathbf{x} + \mathbf{e} = \mathbf{x}' + \mathbf{e}'$, where $\mathbf{x} \neq \mathbf{x}'$ and $\mathbf{e} \neq \mathbf{e}'$. Then $0 = \mathbf{x} - \mathbf{x}' + \mathbf{e} - \mathbf{e}'$. Since $\mathbf{x} - \mathbf{x}' \in L$, we also require that $\mathbf{e} - \mathbf{e}' \in L$. But $\mathbf{e}, \mathbf{e}' \in \mathcal{P}(\mathbf{B})$, so $\mathbf{e} - \mathbf{e}' \in \mathbf{B} \cdot (-1, 1)^k \cap L = \{0\}$. Thus $\mathbf{e} = \mathbf{e}'$ and $\mathbf{x} = \mathbf{x}'$ follows. □

PROPOSITION 6 *Let* L' *be a lattice admitting a basis* $\mathbf{B}$, *and* L *a superlattice of* L' *of the same dimension. The index of the quotient group* $|L/L'| < \infty$ *is finite.*

PROOF: Given $L' \subseteq L$ have the same rank, their $\mathbb{R}$-spans are the same. Consider $S \subset L$ a set of representative of the quotient L/L'. Then $|S| = |L/L'|$ and $S \subset \mathrm{Span}_{\mathbb{R}}(L) = \mathrm{Span}_{\mathbb{R}}(L')$.

Now, by Proposition 5 every $\mathbf{s} \in S$ can be written $\mathbf{s} = \mathbf{t}_s + \mathbf{e}_s$ for some $\mathbf{t}_s \in L'$ and $\mathbf{e}_s \in \mathcal{P}(\mathbf{B})$. Let E be the set of all $\mathbf{e}$ resulting from this decomposition. Since no two representative $\mathbf{s}$ differ by an element of L', we know that each $\mathbf{e}$ is unique to each $\mathbf{s} \in S$, and so $|S| = |E|$. Furthermore, $E \subset L$, so E is uniformly discrete. Following the same argument as in the proof of Proposition 3, we know that only finitely many elements of the uniformly discrete set E can be contained in the bounded set $\mathcal{P}(\mathbf{B})$. But since $E \subseteq \mathcal{P}(\mathbf{B})$, E is finite, and therefore so is $|L/L'|$. □

THEOREM 7 *Any lattice admits a basis.*

PROOF: Let $L \subset \mathbb{R}^n$ be a lattice of rank k. Let $\mathbf{B} \subset \mathbb{R}^{n \times k}$ be matrix whose column vectors are linearly independent vectors of L. By Proposition 6, the quotient $G = L/\mathcal{L}(\mathbf{B})$ is finite.

If the quotient is trivial then $L = \mathcal{L}(\mathbf{B})$, and $\mathbf{B}$ is a basis of L, we are done. Otherwise, choose some $\mathbf{x} \in L \setminus \mathcal{L}(\mathbf{B})$ of prime order p in G. Because $p\mathbf{x} \in \mathcal{L}(\mathbf{B})$, it can be written as $\mathbf{x} = \mathbf{B}\mathbf{y}$ where $\mathbf{y} \in \frac{1}{p}\mathbb{Z}^k$. Without loss of generality (by permutation of the basis vectors of $\mathbf{B}$), we can assume that its first coordinate is not integral: $\mathbf{y} = (\frac{a}{p}, \mathbf{y}')$ where $a \neq 0 \bmod p$. We can further assume that $a = 1$, by replacing $\mathbf{x}$ by $c\mathbf{x} - \frac{ac-1}{p}\mathbf{b}_1$ where $ac = 1 \bmod p$.

The claim is that $\mathbf{B}' := (\mathbf{x}, \mathbf{b}_2, \ldots, \mathbf{b}_n)$ generates a sublattice of L that is a strict superlattice of $\mathcal{L}(\mathbf{B})$, namely $\mathcal{L}(\mathbf{B}) + \mathbf{x}\mathbb{Z}$. That it is a sublattice of L is trivial: by construction $\mathbf{x} \in L$ and for all i, $\mathbf{b}_i \in L$. To show that it is a superlattice of $\mathcal{L}(\mathbf{B})$, it suffices to prove that $\mathbf{b}_1 \in \mathcal{L}(\mathbf{B}')$. Recall that $\mathbf{x} = \mathbf{B} \cdot (\frac{1}{p}, \mathbf{y}')$ for some $\mathbf{y}'$ in $\frac{1}{p} \cdot \mathbb{Z}^{k-1}$. Simply note that $\mathbf{b}_1 = p\mathbf{x} - \mathbf{B}' \cdot (0, p\mathbf{y}')$. Finally, because $\mathbf{x} \notin \mathcal{L}(\mathbf{B})$, the inclusion $\mathcal{L}(\mathbf{B}) \subset \mathcal{L}(\mathbf{B}')$ is indeed strict.

We conclude by repeating the above process, replacing $\mathbf{B}$ by the newly constructed $\mathbf{B}'$, until $\mathbf{B}$ is indeed a basis of L. Note that since $\mathcal{L}(\mathbf{B}) \subsetneq \mathcal{L}(\mathbf{B}')$, the size of the quotient group $|L/\mathcal{L}(\mathbf{B})|$ is finite and stricly decreases at each step. Therefore it must terminate after some finite number of steps, at which point $|L/\mathcal{L}(\mathbf{B})| = 1$, and $\mathcal{L}(\mathbf{B}) = L$. □

PROPOSITION 8 *For any two bases $\mathbf{B}, \mathbf{B}' \in \mathbb{R}^{n\times k}$ of the same lattice L of rank k, there exists $\mathbf{U} \in \mathrm{GL}_k(\mathbb{Z})$ such that $\mathbf{B}' = \mathbf{BU}$. Conversely, if $\mathbf{B}$ is a basis of L, so is $\mathbf{BU}$ for any unimodular matrix $\mathbf{U} \in \mathrm{GL}_k(\mathbb{Z})$.*

PROOF: The lattices $\mathcal{L}(\mathbf{B}), \mathcal{L}(\mathbf{B}')$ are equal and therefore sublattices of one another. Since $\mathcal{L}(\mathbf{B}) \subseteq \mathcal{L}(\mathbf{B}')$, each basis vector in $\mathbf{B}'$ is an integer linear combination of basis vectors in $\mathbf{B}$. This can be written as $\mathbf{B}' = \mathbf{BM}$ for some $\mathbf{M} \in \mathbb{Z}^{k\times k}$. The matrix M has non-zero determinant, since both $\mathbf{B}$ and $\mathbf{B}'$ have rank k. The same is true for the other direction of the inclusion, i.e. $\mathbf{B} = \mathbf{B}'\mathbf{M}'$ for some $\mathbf{M}' \in \mathbb{Z}^{k\times k}$ with rank k. Combining these two facts tells us

$$\mathbf{B} = \mathbf{BMM}'.$$

The matrix $\mathbf{B}$ is non-singular, so the above holds if and only if $\mathbf{MM}' = \mathbf{I}_k$. Since both of $\mathbf{M}$ and $\mathbf{M}'$ are integer valued, their determinants are also integers, and thus $\mathbf{M}, \mathbf{M}' \in \mathrm{GL}_n(\mathbb{Z})$.

For the converse, we argue as above that for any basis $\mathbf{B}$, the lattice generated by $\mathbf{BU}$ for any $\mathbf{U} \in \mathbb{Z}^{k\times k}$ is a sublattice of $\mathcal{L}(\mathbf{B})$, and so $\mathcal{L}(\mathbf{BU}) \subseteq \mathcal{L}(\mathbf{B})$. Now if $\mathbf{U}$ is unimodular, then $\mathbf{U}^{-1} \in \mathbb{Z}^{k\times k}$ and so by the same argument, $\mathcal{L}(\mathbf{B}) = \mathcal{L}(\mathbf{BUU}^{-1}) \subseteq \mathcal{L}(\mathbf{BU})$. Therefore $\mathcal{L}(\mathbf{B}) = \mathcal{L}(\mathbf{BU})$. □

3.2 Covering, Packing, Tiling

Subsets of $\mathbb{R}^n$ are called coverings, packings or tilings based on how well they "fill up" the space between lattice points. A set S is covering if the repeated pattern of S at every lattice point will cover all of $\mathrm{Span}_{\mathbb{R}}(L)$. Similarly, a set S is packing if no repeated instance of S will intersect with another instance of S. Tiling is when both of these are true: every point of $\mathrm{Span}_{\mathbb{R}}(L)$ can be found in exactly one instance of S, when S is repeated at every lattice point. Formally,

DEFINITION 9 (COVERING, PACKING, TILING) *Let $S \subseteq \mathrm{Span}_{\mathbb{R}}(L)$.*

1. *The set S is said to be L-covering if $L + S = \mathrm{Span}_{\mathbb{R}}(L)$.*
2. *S is L-packing if for any pair $\mathbf{t}, \mathbf{u} \in L$, $\mathbf{t} + S \cap \mathbf{u} + S = \emptyset$.*
3. *S is L-tiling if it is both packing and covering.*

All these properties can be though as in term of the union underlying the notation $L + S := \bigcup_{\mathbf{x}\in L} S + \mathbf{x}$: covering means this union is the whole space, and packing means this union is disjoint. Tiling is a synonym of being a fundamental domain for L.

The fundamental parallelepiped $\mathcal{P}(\mathbf{B})$ is a tiling with respect to L, as a corollary of Proposition 5. Since the concept of covering, packing and tiling are so closely linked, we often use an abbreviation of notation that hopefully doesn't lead to any confusion.

PROPOSITION 10 *Let $L \subseteq \mathbb{R}^n$ be a k-dimensional lattice and let $S \subseteq \mathrm{Span}_{\mathbb{R}}(L)$ be a measurable set. If S is L-(covering, packing, tiling), then $\mathrm{vol}(S) \quad (\geq, \leq, =) \quad \mathrm{vol}(\mathcal{P}(\mathbf{B}))$. In particular, every tiling has the same volume.*

PROOF: The proof proceeds with a collage. Let us simplify the notation $P = \mathcal{P}(\mathbf{B})$. Cut S into the disjoint union $S = \bigsqcup_{\mathbf{x}\in L} S \cap (P + \mathbf{x})$. Move each piece back to P by translating it by $-\mathbf{x}$, and glue them back as $T := \bigcup_{\mathbf{x}\in L} ((S - \mathbf{x}) \cap P)$. In particular $T \subseteq P$. By the properties of the Lebesgue measure, $\mathrm{vol}(T) \leq \mathrm{vol}(P)$ and $\mathrm{vol}(T) \leq \mathrm{vol}(S)$.

If S is packing, then the union defining $T = \bigcup_{\mathbf{x}\in L} ((S - \mathbf{x}) \cap P)$ is disjoint because the $(S - \mathbf{x})$ are disjoint. So $\mathrm{vol}(T) = \sum_{\mathbf{x}} \mathrm{vol}((S - \mathbf{x}) \cap P) = \sum_{\mathbf{x}} \mathrm{vol}(S \cap (P + \mathbf{x})) = \mathrm{vol}(S)$ and we conclude that $\mathrm{vol}(S) \leq \mathrm{vol}(P)$.

If S is now covering, we claim that $T = P$, and we conclude that $\mathrm{vol}(S) \geq \mathrm{vol}(P)$. Indeed, $T = \bigcup_{\mathbf{x}\in L} ((S - \mathbf{x}) \cap P) = (\bigcup_{\mathbf{x}\in L}(S - \mathbf{x})) \cap P = \mathrm{Span}_{\mathbb{R}}(L) \cap P = P$. □

3.3 Determinant, Volume and Density

The above leads us to define the determinant of a lattice (sometimes called the volume, or less frequently but more accurately the co-volume).

DEFINITION 11 *The determinant* $\det(L)$ *is defined as* $\sqrt{|\det(\mathbf{B}^t\mathbf{B})|}$ *for any basis* $\mathbf{B}$ *of* L.

It follows from Proposition 8 that this value does not depend on the choice of basis. In that sense it is an *invariant* of the lattice. Note that if L is full rank, then its determinant can be computed more simply as $|\det(\mathbf{B})|$. The determinant can be thought as the inverse of the density of the lattice: tiling the space as $\bigcup_{\mathbf{x}\in L} \mathcal{P}(\mathbf{B}) + \mathbf{x}$, each tile has volume $\mathrm{vol}(\mathcal{P}(\mathbf{B})) = \det(L)$, and contains one lattice point.

This density intuition can be formalized as the following statement, (which applies to any choice of norm).

THEOREM 12 *Let* $L \subset \mathbb{R}^n$ *be a full-rank lattice, and let* $\mathcal{B}$ *denote the closed ball of radius* 1, $\mathcal{B} := \{\mathbf{x} \in \mathbb{R}^n \mid \|\mathbf{x}\| \leq 1\}$. *Then,*

$$\lim_{r\to\infty} \frac{|r\mathcal{B} \cap L|}{r^n \mathrm{vol}(\mathcal{B})} = \frac{1}{\det(L)}.$$

We will prove Theorem 12 using a packing and covering argument.

PROPOSITION 13 *Let* T *be a bounded tiling for a lattice* L, *and* $\mu = \sup_{\mathbf{x}\in T} \|x\|$. *Then, for any radius* $r > \mu$ *it holds that* $\frac{(r-\mu)^n}{\det(L)} \leq \frac{|L\cap r\mathcal{B}|}{\mathrm{vol}(\mathcal{B})} \leq \frac{(r+\mu)^n}{\det(L)}$.

PROOF: We first show the right-hand bound. Let T be a tiling set of L that is contained in $\mu\mathcal{B}$. This is a covering set by definition, so $\mathbb{R}^n = L + T$. Furthermore, $(L \cap r\mathcal{B}) + T \subseteq (r + \mu)\mathcal{B}$ by the triangle inequality. This gives

$$\mathrm{vol}((L \cap r\mathcal{B}) + T) \leq \mathrm{vol}((r + \mu)\mathcal{B}) = (r + \mu)^n \mathrm{vol}(\mathcal{B}).$$

But since T is tiling, and any tiling set has volume $\det(L)$, the left hand side can be written as $\sum_{\mathbf{x}\in L\cap r\mathcal{B}} \mathrm{vol}(T) = |L \cap r\mathcal{B}|\det(L)$. Thus $|L \cap r\mathcal{B}|\det(L) \leq (r + \mu)^n \mathrm{vol}(\mathcal{B})$ and the right hand side follows.

For the left-hand bound, note first that $(r - \mu)\mathcal{B} \subseteq (L \cap r\mathcal{B}) + T$. Then, for the same reasoning as before, we see

$$(r - \mu)^n \mathrm{vol}(\mathcal{B}) \leq |L \cap r\mathcal{B}|\det(L),$$

and the bound follows. □

PROOF OF THEOREM 12: This is an immediate corollary of Proposition 13, by choosing $T = \mathcal{P}(\mathbf{B})$ for any basis $\mathbf{B}$ of L, noting that $\mu \leq \frac{1}{2}\sum \|\mathbf{b}_i\|$ is finite. Divide across by r^n and take the limit. □

4 Minkowski's First Theorem

With all the above prequisite, there is not much left to do to reach Minkowski's first theorem and Minkowski's bound. It is often demonstrated via the so-called Blichfeldt lemma, which at this point is merely the contrapositive of the volume bound for packing (Proposition 10).

LEMMA 14 (BLICHFELDT) *For any lattice L and any measurable set $S \subset \mathrm{Span}_{\mathbb{R}}(L)$ such that $\mathrm{vol}(S) > \det(L)$, there exist distinct $\mathbf{x}, \mathbf{y} \in S$ such that $\mathbf{x} - \mathbf{y} \in L$.*

THEOREM 15 (MINKOWSKI CONVEX BODY THEOREM) *For any lattice L of rank n and any symmetric ($S = -S$)* convex *set $S \subset \mathrm{Span}_{\mathbb{R}}(L)$ such that $\mathrm{vol}(S) > 2^n \cdot \det(L)$, there exists a non-zero lattice vector in S: $|S \cap L| > 1$.*

PROOF: Apply Blichfeldt lemma to the lattice $L' = 2L$, of determinant $\det(L') = 2^n \cdot \det(L)$, and obtain distinct $\mathbf{x}, \mathbf{y} \in S$ such that $\mathbf{x} - \mathbf{y} \in L'$. Note that $\mathbf{z} := \frac{\mathbf{x}-\mathbf{y}}{2} \in L$, and because S is symmetric and convex, $\mathbf{z}$ also belongs to S. Note finally that $\mathbf{z}$ is non-zero since $\mathbf{x}$ and $\mathbf{y}$ are distinct. □

THEOREM 16 (MINKOWSKI BOUND) *The minimal distance of any full rank lattice $L \subset \mathbb{R}^n$ is bounded by below: $\lambda_1(L) \leq 2 \cdot (\frac{\det(L)}{\mathrm{vol}(\mathfrak{B})})^{\frac{1}{n}}$, where $\mathfrak{B}$ is the ball of radius 1: $\mathfrak{B} = \{\mathbf{x} \in \mathbb{R}^n \mid \|\mathbf{x}\| \leq 1\}$.*

In particular, for the Euclidean norm $\|\mathbf{x}\| := \sqrt{\sum x_i^2}$ (a.k.a. the ℓ_2 norm) the unit ball $\mathfrak{B}$ has volume $\mathrm{vol}(\mathfrak{B}) = \frac{\pi^{n/2}}{\Gamma(\frac{n}{2}+1)}$ where $\Gamma(x+1) \sim \sqrt{2\pi x}(x/e)^x$ (Stirling's approximation). Hence, n-dimensional lattice of rank n have a normalized minimal distance of at most:

$$\frac{\lambda_1^{(2)}(L)}{\det(L)^{1/n}} \leq \sqrt{\frac{2n}{\pi e}} + o(\sqrt{n}).$$

Note that because the restriction of an ℓ_2-ball to any subspace of $\mathbb{R}^n$ is still an ℓ_2-ball, this bound is not limited to full-rank lattices.

Lattice Problems and Rounding Algorithms

1 Introduction

In the previous lecture, we established that any lattice L admits a basis $\mathbf{B}$, and furthermore that each basis induces a parallelepiped $\mathcal{P}(\mathbf{B})$ that is a tiling for L. We will first show that this tiling can be performed efficiently by an algorithm. This, in turns allows to a solve computational problems (such as the approximate closest vector problem) up to a certain approximation factor that depends on the basis. This problem is of theoretical interest in Complexity Theory, but also has very practical applications (error correction, quantization, lossy compression).

We will then show that the same basis $\mathbf{B}$ induces another parallelepiped that is tiling for L, namely $\mathcal{P}(\mathbf{B}^\star)$ where $\mathbf{B}^\star$ is the Gram-Schmidt orthogonalization of $\mathbf{B}$, and that this is true despite $\mathbf{B}^\star$ not being a basis for L —an example of this tiling can be viewed in many brick walls, especially in Leiden... An efficient algorithm for the approximate shortest vector problem can also be designed based on this tiling.

The above motivates the rich area of research called basis reduction, which will be studied in the following lectures. Motivating questions include: given that a lattice in 2 or more dimensions can be represented by infinitely many bases, how should you choose one for a specific purpose? How "good" of a basis is guaranteed to exist? Can we find a "good" basis with an algorithm? How costly are those algorithms?

We will finish the lecture with the notion of the Voronoi cell of a lattice, which is in many senses a geometrically optimal tiling of the $\mathbb{R}$-span of the lattice. But it is also computationally costly to deal with. In particular, we will relate its quality for the closest vector problem (CVP) directly to the properties of the lattice.

1.1 Notation

As hinted above, the Euclidean norm (a.k.a. the ℓ_2 norm) will play a special role in this lecture, and in most lectures involving lattice reduction. We will (do our best to[1]) always specify this restriction by using the notation $\|\cdot\|_2$, while $\|\cdot\|$ denotes an arbitrary norm. The associated inner product is denoted $\langle\cdot,\cdot\rangle$. When considering metric properties of a lattice, like the minimal distance or the covering radius, the use of the Euclidean metric will be denote in parentheses in the exponent, e.g. $\lambda_1^{(2)}(L)$, $\mu^{(2)}(L)$.

Algorithms and Complexity for the Mathematician. This course is written for students with a mathematical background, but the topic really lies at the intersection between mathematics and computer science, and in particular complexity theory. We will do our best to provide the necessary notion of complexity theory as we go, in a progressive manner. This section is an overview of the mindset, ideas, and terminology that we will encounter, which will be made more precise later on.

A **computational problem** is the task of an algorithm: given an **input** guaranteed to have a certain property, the algorithm must produce an **output** with prescribed properties. An algorithm is said to be **correct** for a task if it always terminates and fits the requirement stated by the task.

[1] This goes against the habits of the lecturer, so mistakes are likely. Please notify such typos.

When describing an algorithm, it will often be listed together with its **specification**, that is, the problem that it solves. Note that an algorithm can be correct for several tasks, and it will often be the case that we give an algorithm with a tight specification, only to show later that it is correct for a less specific task. For example, we will soon see that there is an algorithm (Algorithm 2) to tile the space $\mathbb{R}^n$ using $\mathcal{P}(\mathbf{B})$ for some basis $\mathbf{B}$ of L. This is a tight specification of the algorithm: the specification dictates a unique valid output for each input. We will prove that this algorithm is correct (Lemma 4) and, as a corollary that it solves the **closest vector problem** up to a certain radius (Corollary 5).

Computational problems and algorithms *are* formal objects, and not some alien object interacting with mathematics. They have several equivalent formal foundations (Church, Turing), though algorithms are often described using "pseudo-code" for convenience and readability. Pseudo-code is no less formal than theorems and proofs as usually written in textbooks and mathematical papers; it is "pseudo" in the sense that it is not machine-readable (in the same way that proofs are not machine-readable unless written in a language like `Coq` or `Lean`).

The **complexity** of an algorithm is a measure of how many operations it performs. This is typically given in number of bit operations, but this is sometimes inconvenient. As an intermediate step, we may just count the number of arithmetic operations in an algorithm: the gap between both methods of counting depends on how large the numbers in the computations are. Take Algorithm 1 as an example.

Algorithm 1: Addition Algorithm

Input : An integer x and an integer y.
Output: An integer z such that $z = x + y$

return $x + y$

The number of arithmetic operations in this algorithm is 1, because the algorithm just adds two numbers and outputs the answer. But the bit complexity is larger, because a computer will add two integers the same way we all learn to in school: add digits in one column, and if the value is greater than ten, *carry the one* to the next columns and continue. In a computer this is all done in base two, so the number of **bit operations** is bounded above by the number of bits required to write the two numbers in binary.

The complexity can be given explicitly in terms of the input, such as the dimension n of the input matrix $\mathbf{B} \in \mathbb{Q}^{n \times n}$. But when we merely say that an algorithm is quadratic, or **polynomial time** without specifying in which variable, it is implicitly meant as a function of the **bitsize of the input**. Explicitly, an algorithm runs in polynomial time if there exists constants C, d such that for any valid input of bitsize K, the algorithm outputs a correct answer after no more than CK^d bit operations. This is more often abbreviated by saying that the running time is $O(K^d)$. For an integer output $x \in \mathbb{Z}$, the bitsize of the input is $\log_2 |x| + O(1)$, while a rational $x/y \in \mathbb{Q}$ can be represented using two integers x and y, and an $n \times k$ matrix can be represented using nk inputs, etc.

Not only can we write theorems about algorithms (correctness, complexity) and quantify over classes of algorithms, but we can also formally relate computational problems to one another. In particular, complexity theory is often interested in showing that a computational problem A is "easier" or "not much harder" than a computational problem B. Or, one can say equivalently that computational problem A is **reducible** to computational problem B. This means that one

can solve an instance of problem A by (cheaply) transforming it to be solvable in one or more instances of problem B. When designing such a **reduction**, the unknown algorithm for solving B is called an **oracle** for B: we only assume that it solves B, but we do not know how it does it.

These type of statements are also central in cryptography. We will want to convincingly argue for or against the security of a cryptosystem, which is the practical (im)possibility of breaking it. A cryptosystem might be complex; it might be interactive between the sender and receiver; or there may even be many parties involved. But by reducing its (formally defined) security to a simple and non-interactive computational problem, we can better focus the effort of attempting attacks (cryptanalysis) and be reassured about its security.[2]

2 Lattice Computational Problems

There are many (many!) lattice problem variants that are of interest, and we should not list them all here in full detail. We focus on the most important ones.

Definition 1 (Exact SVP and α-SVP) *The approximate Shortest Vector Problem with approximation factor $\alpha \geq 1$, denoted α-SVP is defined as follows:*

- *Given as input the basis $\mathbf{B}$ of a lattice L*
- *Output a short non-zero vector $\mathbf{v} \in L \setminus \{\mathbf{0}\}$, satisfying $\|\mathbf{v}\| \leq \alpha \cdot \lambda_1(L)$.*

The Exact version of the problem, denoted SVP, is the special case $\alpha = 1$.

Note that $\lambda_1(L)$ is not necessarily known, which can be inconvenient. In particular we cannot even efficiently verify that a given solution is indeed correct. For this reason, some variations of these problems are sometime considered, such as α-Hermite SVP, or α-HSVP. Here, shortness is defined relative to the determinant: $\|\mathbf{v}\| \leq \alpha \cdot \det(L)^{1/k}$ where k is the rank of L. The factor $\det(L)^{1/k}$ makes the problem invariant by rescaling $L \mapsto s \cdot L$ for $s > 0$. Or it might be sometimes convenient to simply state an absolute bound for the desired vector (AbsSVP).

Note further that, even for $\alpha = 1$, the solution is never unique since $\|-\mathbf{v}\| = \|\mathbf{v}\|$. But even up to flipping the sign, there may be many solutions. Take for example the lattice

$$L = \{(x_1, \ldots x_n) \in \mathbb{R}^n : x_i \in \mathbb{Z} \ \forall \ i\}.$$

This lattice has $2n$ shortest vectors. Meanwhile, in dimension $n = 24$ there is a lattice with 196560 shortest vectors. Therefore we should always speak of *a* shortest vector rather than *the* shortest vector.

Definition 2 (Exact CVP and α-CVP) *The approximate Closest Vector Problem with approximation factor $\alpha \geq 1$, denoted α-CVP is defined as follows:*

- *Given as input the basis $\mathbf{B}$ of a lattice $L \subset \mathbb{R}^n$ and a target $\mathbf{t} \in \mathbb{R}^n$*
- *Output a vector $\mathbf{v} \in L$ satisfying $\|\mathbf{v} - \mathbf{t}\| \leq \alpha \cdot d(\mathbf{t}, L)$*

[2] And yet, the saying is that "Cryptographers seldom sleep well at night". In particular, if $P = NP$, then cryptography as we know it would entirely collapse. In fact, cryptography relies on the stronger and more specific assumption than $P \neq NP$.

where $d(\mathbf{t}, L) := \min_{\mathbf{x}\in L} \|\mathbf{x} - \mathbf{t}\|$ *denotes the distance to the lattice. The Exact version of the problem, denoted CVP, is the special case* $\alpha = 1$.

For the same reasons as for SVP, variants of this problem where the norm is bounded in absolute terms is also convenient (AbsCVP). Lastly, we define a restriction of the Exact CVP problem, where this time we are also given a guarantee that the solution is particularly close to the lattice, and therefore, unique.

Definition 3 (α-BDD) *The Bounded Distance Decoding Problem with approximation factor* $\alpha \leq 1/2$*, denoted* α*-BDD is defined as follows:*

- *Given as input the basis* $\mathbf{B}$ *of a lattice* $L \subset \mathbb{R}^n$ *and a target* $\mathbf{t} \in \mathbb{R}^n$ *such that* $d(\mathbf{t}, L) < \alpha\lambda_1(L)$
- *Output the unique closest vector* $\mathbf{v} \in L$ *satisfying* $\|\mathbf{v} - \mathbf{t}\| < \alpha\lambda_1(T)$

where $d(\mathbf{t}, L) := \min_{\mathbf{x}\in L} \|\mathbf{x} - \mathbf{t}\|$ *denotes the distance to the lattice.*

Once again, an absolute version (AbsBDD) will also be convenient to consider.

Close Vectors and Tiling. Many algorithms of interest for CVP and BDD are related to L-tilings. That is, for a subset $T \subseteq \mathrm{Span}_{\mathbb{R}}(L)$ that is L-tiling, an algorithm is said to *solve T-tiling* if, when given an input basis $\mathbf{B}$ and target $\mathbf{t}$, the algorithm outputs a CVP/BDD vector $\mathbf{v} \in L$ such that the error $\mathbf{e} = \mathbf{v} - \mathbf{t}$ is in T.

In this case, the we can state that the algorithm solves CVP and BDD up to radii that depend on the following geometric properties of T. For any bounded body $T \subset \mathbb{R}^n$, we define its inner radius $\nu(T)$ and its outer radius $\mu(T)$ by

$$\nu(T) := \sup\{r \in \mathbb{R} \mid r \cdot \mathfrak{B} \subset T\}, \quad \mu(T) := \inf\{r \in \mathbb{R} \mid T \subset r \cdot \mathfrak{B}\}. \tag{1}$$

Alternatively, the above may be re-written as:

$$\nu(T) := \inf_{\mathbf{x}\in\mathbb{R}^n\setminus T} \|\mathbf{x}\|, \quad \mu(T) = \sup_{\mathbf{x}\in T} \|\mathbf{x}\|. \tag{2}$$

If a close vector algorithm is associated with a tiling T, then it correctly solves:

- AbsBDD up to a radius $\nu(T)$
- AbsCVP up to a radius $\mu(T)$
- α-CVP up to an approximation factor $\alpha = \mu(T)/\nu(T)$.

Note that the inner and outer radius of tilings are bounded by the metric properties of the lattice: for any tiling T of L, it must be that $\mu(T) \geq \mu(L)$ and $\nu(T) \leq \lambda_1(L)/2$.

3 The Simple Rounding algorithm and the $\mathcal{P}(\mathbf{B})$ Tiling

The Simple Rounding algorithm merely rounds the coordinate of the target vector when it is expressed in terms of the basis $\mathbf{B}$. One can understand the algorithm as a reduction to solving CVP in the trivial lattice $\mathbb{Z}^n$: multiply by $\mathbf{B}^{-1}$ which sends L to $\mathbb{Z}^n$, solve CVP on the lattice $\mathbb{Z}^n$ by coordinate-wise rounding to the nearest integer, and then translate the solution back to L by multiplying it by $\mathbf{B}$.

Algorithm 2: SimpleRounding$(\mathbf{B}, \mathbf{t})$: Simple Rounding Algorithm

Input : A basis $\mathbf{B} \in \mathbb{Q}^{n\times n}$ of a full rank lattice L, a target $\mathbf{t} \in \mathrm{Span}_{\mathbb{R}}(L)$.
Output: $\mathbf{v} \in L$ such that $\mathbf{e} = \mathbf{t} - \mathbf{v} \in \mathcal{P}(\mathbf{B})$

$\mathbf{t}' \leftarrow \mathbf{B}^{-1} \cdot \mathbf{t}$
$\mathbf{v}' \leftarrow (\lfloor t'_i \rceil)_{i\in\{1\dots n\}}$
$\mathbf{v} \leftarrow \mathbf{B} \cdot \mathbf{v}'$
return $\mathbf{v}$

LEMMA 4 *Algorithm SimpleRounding is correct and runs in polynomial time.*

PROOF: Let us start with correctness. Because L is full-rank, its basis B is invertible, so the first line of the algorithm is well-defined. By construction, $\mathbf{v}'$ belongs to $\mathbb{Z}^n$, and because $\mathbf{B}$ is a basis of L, $\mathbf{v} = \mathbf{B}\mathbf{v}'$ belongs to L. Now define $\mathbf{e}' := \mathbf{t}' - \mathbf{v}'$, which by construction belongs to $[-1/2, 1/2)^n$. Note that $\mathbf{e} = \mathbf{B} \cdot \mathbf{e}'$, so by definition of $\mathcal{P}(\mathbf{B}) = \mathbf{B} \cdot [-1/2, 1/2)^n$ we have $\mathbf{e} \in \mathcal{P}(\mathbf{B})$.

For proving polynomial time complexity, the main issue is inversion of the matrix $\mathbf{B}^{-1}$. The Gaussian elimination process requires $O(n^3)$ arithmetic operation, but one must also show that the numerators and denominators of the rational numbers that occur in the intermediate steps remain small enough. □

COROLLARY 5 *For* $\mathbf{B}$ *a basis of* L *, the algorithm* SimpleRounding$(\mathbf{B}, \cdot)$ *solves* $\mu(\mathcal{P}(\mathbf{B}))$*-AbsCVP and* $\nu(\mathcal{P}(\mathbf{B}))$*-AbsBDD.*

Inner and Outer Radii of $\mathcal{P}(\mathbf{B})$. The outer radius $\mu(\mathcal{P}(\mathbf{B}))$ admits a natural upper bound by direct application of the triangle inequality:

$$\mu(\mathcal{P}(\mathbf{B})) \leq \frac{1}{2}\sum_{i=1}^{n} \|\mathbf{b}_i\| . \tag{3}$$

In the particular case of the Euclidean norm, this bound is never reached, that is $\mu^{(2)}(\mathcal{P}(\mathbf{B})) < \frac{1}{2}\sum_{i=1}^{n} \|\mathbf{b}_i\|_2$ because an equality instance of the triangle inequality occurs when vectors are co-linear, but the vectors of a basis can never be collinear. However, one can get arbitrarily close to it, for example

$$\mathbf{B} = \begin{pmatrix} 1 & 1 & 1 & \cdots & 1 \\ 0 & \varepsilon & 0 & \cdots & 0 \\ 0 & 0 & \varepsilon & \cdots & 0 \\ \vdots & & & \ddots & \\ 0 & 0 & 0 & \cdots & \varepsilon \end{pmatrix} \quad \text{gives } \frac{\mu^{(2)}(\mathcal{P}(\mathbf{B}))}{\frac{1}{2}\sum_{i=1}^{n} \|\mathbf{b}_i\|_2} \geq \frac{n}{n\sqrt{1+\varepsilon^2}} \to 1 \text{ as } \varepsilon \to 0, \tag{4}$$

On the contrary, the same example with any $\varepsilon \in (-1, 1)$ shows that this upper bound can be reached for the ℓ_∞ norm.

Similarly, one would like a lower bound on the inner radius $\nu(\mathcal{P}(\mathbf{B}))$. This turns out to be more challenging for general norms. However, it can be exactly determined for the Euclidean norm. Let us start with a characterization of the parallelepiped $\mathcal{P}(\mathbf{B})$ which allows us to describe a parallelepiped by a system of linear inequalities rather than by a spanning basis.

LEMMA 6 *Let* $\mathbf{B} \in \mathrm{GL}_n(\mathbb{R})$ *and* $\mathbf{C} = (\mathbf{B}^{-1})^\top$ *be its inverse-transpose. Then,*

$$\mathcal{P}(\mathbf{B}) = \left\{ \mathbf{x} \in \mathbb{R} \mid \mathbf{c}_i^\top \cdot \mathbf{x} \in [1/2, 1/2) \text{ for all } i \in \{1, \dots, n\} \right\}.$$

PROOF: Recall that $\mathcal{P}(\mathbf{B}) = \mathbf{B} \cdot [-1/2, 1/2)^n$ and rewrite the condition $\mathbf{x} \in \mathcal{P}(\mathbf{B})$ as $\mathbf{B}^{-1} \cdot \mathbf{x} \in [-1/2, 1/2)^n$. Each coordinate y_i of $\mathbf{y} = \mathbf{B}^{-1}\mathbf{x}$ writes $\mathbf{c}_i^\top \cdot \mathbf{x} \in [1/2, 1/2)$ for all $i \in \{1, \dots, n\}$. □
From there, we obtain an explicit

LEMMA 7 *Let* $\mathbf{B} \in \mathrm{GL}_n(\mathbb{R})$ *and* $\mathbf{C} = (\mathbf{B}^{-1})^\top$ *be it's inverse-transpose. Then*

$$\nu^{(2)}(\mathcal{P}(\mathbf{B})) = \min_i \frac{1}{2 \cdot \|\mathbf{c}_i\|_2}.$$

PROOF: Recall that $\nu(T) := \inf_{\mathbf{x} \in \mathbb{R}^n \setminus T} \|\mathbf{x}\|$. Note that for the canonical inner product $\langle \cdot, \cdot \rangle$ of $\mathbb{R}^n$, $\mathbf{c}_i^\top \cdot \mathbf{x} = \langle \mathbf{c}_i, \mathbf{x} \rangle$. By Cauchy-Schwarz and the above characterization, it holds that any $\mathbf{x}$ of norm strictly less than $\min_i \frac{1}{2 \cdot \|\mathbf{c}_i\|_2}$ belongs to $\mathcal{P}(\mathbf{B})$. On the contrary, the vector $\mathbf{x} = \frac{\mathbf{c}_i}{2\|\mathbf{c}_i\|_2^2}$ satisfies $\mathbf{c}_i^\top \cdot \mathbf{x} = 1/2$ and therefore does not belong to $\mathcal{P}(\mathbf{B})$, and this vector has norm $\frac{1}{2 \cdot \|\mathbf{c}_i\|_2}$. Minimizing over i yields the claim. □

Lattice Problems and Rounding Algorithms - Part II

1 Orthogonal Projections

Definition 1 *The Gram-Schmidt Orthogonalization (GSO)* $\mathbf{B}^\star$ *of a non-singular matrix* $\mathbf{B}$ *is defined by* $\mathbf{b}_i^\star = \pi_i(\mathbf{b}_i)$ *where* π_i *denotes the projection orthogonal to the span of* $\mathbf{b}_1, \ldots, \mathbf{b}_{i-1}$.

The projections π_i can be given explicitly by recursion:

$$\pi_i : \mathbf{x} \mapsto \mathbf{x} - \sum_{j<i} \frac{\langle \mathbf{x}, \mathbf{b}_j^\star \rangle}{\|\mathbf{b}_j^\star\|_2^2} \mathbf{b}_j^\star.$$

We recall some key properties of the Gram-Schmidt Orthogonalization:

- $\mathbf{b}_1^\star = \mathbf{b}_1$
- $\mathrm{Span}_{\mathbb{R}}(\mathbf{b}_1, \ldots, \mathbf{b}_j) = \mathrm{Span}_{\mathbb{R}}(\mathbf{b}_1^\star, \ldots, \mathbf{b}_j^\star)$ for all $j \leq n$
- $\mathbf{b}_i^\star \perp \mathbf{b}_j^\star$ for all $i \neq j$
- $\mathbf{b}_i \perp \mathbf{b}_j^\star$ for all $i < j$.

The Gram-Schmidt Orthogonalization can also be written as a matrix decomposition: $\mathbf{B} = \mathbf{B}^\star \cdot \mathbf{T}$ where $\mathbf{T}$ is an upper triangular matrix with unit diagonal and $\mathbf{B}^\star$ has orthogonal row. In particular $\det(\mathbf{T}) = 1$, and therefore if $\mathbf{B}$ is a basis of a lattice L we have

$$\prod_{i=1}^{n} \|\mathbf{b}_i^\star\|_2 = \sqrt{|\det(\mathbf{B}^{\star\top}\mathbf{B}^\star)|} = \sqrt{|\det(\mathbf{B}^\top \mathbf{B})|} = \det(L).$$

Note however that $\mathbf{T}$ is not necessarily integral: $\mathbf{B}^\star$ is not necessarily a basis of L! And yet, we will see that $\mathcal{P}(\mathbf{B}^\star)$ is also a tiling of L.

For any non-zero vector $\mathbf{v}$, we denote by $\pi_{\mathbf{v}}^\perp : \mathbf{x} \mapsto \mathbf{x} - \frac{\langle \mathbf{x}, \mathbf{v} \rangle}{\|\mathbf{v}\|_2^2} \mathbf{v}$ the projection orthogonally to $\mathbf{v}$.

Theorem 2 *Let* L *be a rank* k *lattice, and* $\mathbf{v} \in L$. *Then,* $\pi_{\mathbf{v}}^\perp(L)$ *is a lattice of rank* $k - 1$.

Proof: Because $\pi_{\mathbf{v}}^\perp$ is linear, $L' = \pi_{\mathbf{v}}^\perp(L)$ is a group. To prove that it is discrete, we claim that any $\mathbf{x} \in \pi_{\mathbf{v}}^\perp(L)$ has a pre-image $\mathbf{y} \in L$ such that $|\langle \mathbf{y}, \mathbf{v} \rangle| \leq \frac{1}{2}\|\mathbf{v}\|^2$; i.e. there exists a pre-image of $\mathbf{x}$ in $\mathbf{x} + [-1/2, 1/2) \cdot \mathbf{v}$. Indeed, consider an arbitrary pre-image $\mathbf{y}'$ of $\mathbf{x}$, and set $k = \left\lfloor \frac{\langle \mathbf{y}', \mathbf{v} \rangle}{\|\mathbf{v}\|^2} \right\rceil \in \mathbb{Z}$, and set $\mathbf{y} = \mathbf{y}' - k\mathbf{v}$.

If L' were not discrete, it would contain infinitely many distinct points in the unit ball $\mathfrak{B}$. Each of them can be lifted to a distinct point of L in the body $\mathfrak{B} + [-1/2, 1/2] \cdot \mathbf{v}$, which is a bounded: this would imply that L is not discrete.

Regarding the rank, we note the inclusion $\mathrm{Span}_{\mathbb{R}}(L') \subset \pi_{\mathbf{v}}^\perp(\mathrm{Span}_{\mathbb{R}}(L))$. Since $\mathbf{v} \in \mathrm{Span}_{\mathbb{R}}(L)$, the dimension of $\pi_{\mathbf{v}}^\perp(\mathrm{Span}_{\mathbb{R}}(L))$ is $k - 1$, so the rank of L' is at most $k - 1$. We also note that $\mathrm{Span}_{\mathbb{R}}(L') + \mathbf{v} \cdot \mathbb{R} \supseteq \mathrm{Span}_{\mathbb{R}}(L)$, hence the rank of L' is at least $k - 1$. □

The process by which we chose a particular lift $\mathbf{y}$ of $\mathbf{x}$ in the segment $\mathbf{x} + [-1/2, 1/2) \cdot \mathbf{v}$ will be central in many algorithms to follow, referred to as the **nearest plane method** or as **size-reduction**. Note that in general, such a lift may not be unique. This can happen if $\mathbf{v}$ is not *primitive* with respect to L, that is if $\mathbf{v}$ is an integral multiple of another vector $\mathbf{v} = j \cdot \mathbf{w}$ for some integer $j > 1$. This is in fact, the only exception to uniqueness.

Definition 3 (Primitive Vector) *A non-zero vector* $\mathbf{v}$ *in a lattice* L *is said primitive (with respect to* L*) if* $(\mathbf{v}\cdot\mathbb{R})\cap L = \mathbf{v}\cdot\mathbb{Z}$*. Equivalently, it is primitive if* $\frac{1}{j}\mathbf{v}\notin L$ *for any integer* $j>1$*.*

Lemma 4 *A vector* $\mathbf{v}$ *in a lattice* L *is primitive if and only if* $\mathbf{v}$ *is part of some basis of* L*.*

Proof: Let $\mathbf{B}$ be a basis of L with $\mathbf{b}_1 = \mathbf{v}$. Let j be a positive integer such that $\mathbf{w} = \frac{1}{j}\mathbf{v}$ is in L. Because $\mathbf{w}\in L$, we can write $\mathbf{w} = \mathbf{B}\cdot\mathbf{x}$ for some non-zero integer vector $\mathbf{x}\in\mathbb{Z}^n$. It can also be written as $\mathbf{w} = \mathbf{B}\cdot(1/j, 0, \ldots, 0)$; because $\mathbf{B}$ is non singular this implies $\mathbf{x} = (1/j, 0, \ldots, 0)$, and therefore that $j = 1$.

Reciprocally, let $\mathbf{v}$ be primitive, and let $\mathbf{B}'$ be a basis of $L' = \pi_{\mathbf{v}}(L)$. Let $\mathbf{b}_i$ be a pre-image in L of $\mathbf{b}'_i$ for each $i\in\{1,\ldots,n-1\}$. We claim that setting $\mathbf{b}_n = \mathbf{v}$ makes $\mathbf{B} = (\mathbf{b}_1,\ldots,\mathbf{b}_n)$ into a basis of L. By construction, all the $\mathbf{b}_i$ belong to L so $\mathbf{B}$ generates a sublattice S of L. Furthermore, S and L have the same rank, so any vector $\mathbf{w}$ of L writes as $\mathbf{B}\cdot\mathbf{x}$ for some *real* vector $\mathbf{x}\in\mathbb{R}^n$; we need to prove that $\mathbf{x}$ is in fact integer. Note that $\pi_{\mathbf{v}}(\mathbf{w}) = \sum_{i=1}^{n-1} x_i\mathbf{b}'_i$, and that it belongs to L', hence the x_i's are all integers for $i\in\{1\ldots n-1\}$. Substracting $\sum_{i=1}^{n-1} x_i\mathbf{b}_i$ from $\mathbf{w}$, we get that $\mathbf{B}\cdot(0,0,\ldots,0,x_n)$ belongs to L, that is $x_n\cdot\mathbf{v}\in L$: by primitivity of $\mathbf{v}$, x_n is an integer. Hence $\mathbf{B}$ is indeed a basis of L. □

2 The Nearest Plane Algorithm and the $\mathcal{P}(\mathbf{B}^\star)$ Tiling

Lemma 5 *Let* $\mathbf{v}$ *be a primitive vector of a lattice* L*, define* $L' = \pi^\perp_{\mathbf{v}}(L)$*, and let* T' *be a tiling of* L'*. Then* $T = T' + [-1/2, 1/2)\cdot\mathbf{v}$ *is a tiling of* L*.*

Proof: Let us shorten $\pi^\perp_{\mathbf{v}}$ as π. We start by showing that T is covering for L. Any target $\mathbf{t}\in\mathrm{Span}_{\mathbb{R}}(L)$ as $\mathbf{t} = t_1\mathbf{v} + \mathbf{t}'$ where $\mathbf{t}' = \pi(\mathbf{t})\in\mathrm{Span}_{\mathbb{R}}(L')$. Because T' is covering for L', $\mathbf{t}'$ can be written as $\mathbf{t}' = \mathbf{e}' + \mathbf{w}'$ for some $\mathbf{e}'\in T'$ and $\mathbf{w}'\in L'$. Let $\mathbf{w}''\in L$ be a pre-image of $\mathbf{w}'$ for π, in particular $\mathbf{w}'' = \mathbf{w}' + w_1\mathbf{v}$ for some $w_1\in\mathbb{R}$. Unrolling, we have:

$$\begin{aligned}
\mathbf{t} &= t_1\mathbf{v} + \mathbf{t}'\\
&= t_1\mathbf{v} + \mathbf{e}' + \mathbf{w}'\\
&= (t_1 - w_1)\mathbf{v} + \mathbf{e}' + \mathbf{w}''\\
&= \underbrace{((t_1 - w_1) - \lfloor t_1 - w_1\rceil)\cdot\mathbf{v}}_{\in[-1/2,1/2)\cdot\mathbf{v}} + \underbrace{\mathbf{e}'}_{\in T'} + \underbrace{\lfloor t_1 - w_1\rceil\cdot\mathbf{v} + \mathbf{w}''}_{\in L}
\end{aligned}$$

and we conclude that $T = T' + [-1/2, 1/2)\cdot\mathbf{v}$ is covering for L.

We now prove that T is packing for L. Let $\mathbf{t}\in\mathrm{Span}_{\mathbb{R}}(L)$ and let $\mathbf{e}+\mathbf{f}+\mathbf{w} = \mathbf{e}'+\mathbf{f}'+\mathbf{w}' = \mathbf{t}$ were $\mathbf{e},\mathbf{e}'\in T'$, $\mathbf{f},\mathbf{f}'\in[-1/2,1/2)\cdot\mathbf{v}$ and $\mathbf{w},\mathbf{w}'\in L$. Note that $\pi(\mathbf{t}) = \mathbf{e} + \pi(\mathbf{w}) = \mathbf{e}' + \pi(\mathbf{w}')$ where $\pi(\mathbf{w}),\pi(\mathbf{w}')\in L'$. Since T' is L' packing we have that $\mathbf{e} = \mathbf{e}'$ and that $\pi(\mathbf{w}) = \pi(\mathbf{w}')$. The kernel of π over L is $(\mathbb{R}\cdot\mathbf{v})\cap L$, which by primitivity is exactly $\mathbf{v}\cdot\mathbf{Z}$, so $\mathbf{w}-\mathbf{w}'\in\mathbb{Z}\cdot\mathbf{v}$. Furthermore, $\mathbf{f}-\mathbf{f}' = \mathbf{w}-\mathbf{w}'$, and $\mathbf{f}-\mathbf{f}'\in(-1,1)\cdot\mathbf{v}$; noting that $(-1,1)\cdot\mathbf{v}\cap\mathbb{Z}\mathbf{v} = \{\mathbf{0}\}$ we conclude that $\mathbf{f} = \mathbf{f}'$ and $\mathbf{w} = \mathbf{w}'$. That is, T is indeed packing. □

Corollary 6 *Let* $\mathbf{B}$ *be a basis of* L*; then* $\mathcal{P}(\mathbf{B}^\star)$ *is a tiling of* L*.*

PROOF: We proceed by induction on the dimension. The base case when $n = 1$ is immediate since then $\mathbf{B}^\star = \mathbf{B}$. Denote $\pi = \pi_{\mathbf{b}_1}$ and let $\mathbf{C} = (\pi(\mathbf{b}_2), \ldots, \pi(\mathbf{b}_n))$ be a basis of $L' = \pi(L)$. Our inductive assumption is that $\mathcal{P}(\mathbf{C}^\star)$ is tiling for $\pi(L)$. By Lemma 5, we have that $\mathcal{P}(\mathbf{C}^\star) + [-1/2, 1/2) \cdot \mathbf{b}_1$ is tiling for L. It remains to note that $\mathbf{b}_1^\star = \mathbf{b}_1$ and that $\mathbf{b}_{i+1}^\star = \mathbf{c}_i^\star$ to conclude that $\mathcal{P}(\mathbf{B}^\star) = \mathcal{P}(\mathbf{C}^\star) + [-1/2, 1/2) \cdot \mathbf{b}_1$ is tiling. □

An important remark is that $\mathcal{P}(\mathbf{B}^\star)$ can be characterized as follow:

$$\mathbf{e} \in \mathcal{P}(\mathbf{B}^\star) \Leftrightarrow \langle \mathbf{e}, \mathbf{b}_i^\star \rangle / \|\mathbf{b}_i^\star\|_2^2 \in [-1/2, 1/2) \text{ for all } i. \tag{1}$$

This characterization is a special case of the argument used in the proof of Lemma 7 from the previous lecture, using the property that $\mathbf{B}^\star$ has orthogonal columns, as in the claim below.

CLAIM 7 *If $\mathbf{M} \in \mathrm{GL}_n(\mathbb{R})$ has orthogonal columns, then $\mathbf{M}^{-1} = \mathbf{D}^{-1} \cdot \mathbf{M}^\top$ where $\mathbf{D}$ is a diagonal matrix with $d_{i,i} = \|\mathbf{m}_i\|_2^2$.*

PROOF: Note that $\mathbf{M}$ having orthogonal columns is equivalent to $\mathbf{M}^\top \cdot \mathbf{M}$ being diagonal. □

In this case, defining $\mathbf{C} = (\mathbf{B}^{\star -1})^\top$, we have $\mathbf{C} = (\mathbf{D}^{-1} \cdot \mathbf{B}^{\star \top})^\top = \mathbf{B}^\star \cdot \mathbf{D}^{-\top} = \mathbf{B}^\star \cdot \mathbf{D}^{-1}$ and conclude that $\mathbf{c}_i = \mathbf{b}_i^\star / \|\mathbf{b}_i\|_2^2$.

Algorithm 1: NearestPlane$(\mathbf{B}, \mathbf{t})$: Nearest Plane Algorithm (Babai)

Input : A basis $\mathbf{B} \in \mathbb{Q}^{n \times n}$ of a full rank lattice Λ, a target $\mathbf{t} \in \mathrm{Span}_{\mathbb{R}}(L)$.
Output: $\mathbf{v} \in L$ such that $\mathbf{e} = \mathbf{t} - \mathbf{v} \in \mathcal{P}(\mathbf{B}^\star)$

Compute the GSO $\mathbf{B}^\star$ of $\mathbf{B}$
$\mathbf{v} \leftarrow \mathbf{0}$
$\mathbf{e} \leftarrow \mathbf{t}$
for $i = n$ *down to* 1 **do**
  $k \leftarrow \lfloor \langle \mathbf{e}, \mathbf{b}_i^\star \rangle / \|\mathbf{b}_i^\star\|^2 \rceil$
  $\mathbf{v} \leftarrow \mathbf{v} + k\mathbf{b}_i$
  $\mathbf{e} \leftarrow \mathbf{v} - k\mathbf{b}_i$
end
return $\mathbf{v}$

LEMMA 8 *Algorithm 1 is correct and runs in polynomial time.*

PROOF: For correctness, we consider various invariants of the **for** loop. First, the equation $\mathbf{v} + \mathbf{e} = \mathbf{t}$ is true at initialization and maintained at each iteration. Secondly, $\mathbf{v} = \mathbf{0}$ at initialization so $\mathbf{v} \in L$, and it remains in L during the loop as we only add integer combination of basis vectors.

We now prove that $\mathbf{e} \in \mathcal{P}(\mathbf{B}^\star)$ at the end of the algorithm. By construction of k, and noting that $|\langle \mathbf{b}_i, \mathbf{b}_i^\star \rangle| = \|\mathbf{b}_i^\star\|^2$, it holds that $|\langle \mathbf{e}, \mathbf{b}_i^\star \rangle| / \|\mathbf{b}_i^\star\|^2 \in [-1/2, 1/2)$ at the end of iteration i. Furthermore, the inner product $\langle \mathbf{e}, \mathbf{b}_i^\star \rangle$ is unaffected by the operation $\mathbf{e} \leftarrow \mathbf{v} - k\mathbf{b}_j$ at later stages of the loop $j < i$ because $\mathbf{b}_j \perp \mathbf{b}_i^\star$ (note crucially that the loop goes by **decreasing** indices i).

We conclude that by the end of the algorithm, it holds that $|\langle \mathbf{e}, \mathbf{b}_i^\star \rangle| / \|\mathbf{b}_i^\star\|^2 \in [-1/2, 1/2)$ for all i, which, by the characterization (1) implies $\mathbf{e} \in \mathcal{P}(\mathbf{B}^\star)$.

Regarding polynomial running time, the algorithm, including the GSO process itself, requires $O(n^3)$ operations over $\mathbb{Q}$, but it remains to analyze how large the numerators and denominators at hand are. We refer the interested reader to Micciancio's Lecture notes.[1] □

[1] https://cseweb.ucsd.edu/classes/wi10/cse206a/lec2.pdf

Inner and Outer Radii of $\mathcal{P}(\mathbf{B}^\star)$. Because orthogonal projections and GSO are intimately tied to the Euclidean metric, we will only consider inner and outer radius in the ℓ_2 norm.

For the outer radius we claim that:

$$\mu^{(2)}(\mathcal{P}(\mathbf{B}^\star)) = \frac{1}{2}\sqrt{\sum_{i=1}^{n} \|\mathbf{b}_i^\star\|_2^2}. \tag{2}$$

Indeed, because the $\mathbf{b}_i^\star$ are orthogonal we have Euclidean additivity: $\|\mathbf{B}^\star \cdot \mathbf{x}\|_2^2 = \sum x_i^2 \cdot \|\mathbf{b}_i^\star\|_2^2$. Now since $\mathcal{P}(\mathbf{B}^\star) = \mathbf{B}^\star \cdot [-1/2, 1/2)^n$, the result follows.

At last, using Lemma 7 from the previous lecture and the characterization of orthogonal parallelepiped (1) we can cumpute the inner-radius:

$$\nu^{(2)}(\mathcal{P}(\mathbf{B}^\star)) = \frac{1}{2}\min_{i=1}^{n} \|\mathbf{b}_i^\star\|_2 . \tag{3}$$

Enumeration Algorithms

1 Introduction

In the previous lecture, we defined computational problems for lattices, namely the algorithmic task of finding short non-zero vectors of a lattice (SVP), or a lattice point close to a given target (CVP, BDD). We then provided two algorithms (Simple-Rounding, Nearest-Plane) which, given a basis $\mathbf{B}$ of a lattice L, would tile the space according to the L-tilings $T = \mathcal{P}(\mathbf{B})$ and $T = \mathcal{P}(\mathbf{B}^\star)$ respectively. That is, given a target $\mathbf{t}$, these algorithm can find a lattice point $\mathbf{v} \in L$ such that the error $\mathbf{e} = \mathbf{t} - \mathbf{v}$ belongs to T. By studying the inner radius $\nu(\mathcal{P}(\mathbf{B}))$ and outer radius $\mu(\mathcal{P}(\mathbf{B}))$ we finally concluded with a length guarantees (as a function of $\mathbf{B}$) for which those algorithms solve approximate versions of BDD and CVP.

These algorithm were "fast"; they ran in time polynomial in the size of the input, but they only guaranteed to solve the closest vector problem up to a potentially large approximation factor. Today's lecture is about "slow" algorithms, whose running time can be as large as exponential in the input size, but that can solve SVP, CVP and BDD exactly. Namely, there will be two algorithms directly inspired from Simple-Rounding and Nearest-Plane, but instead of considering only one candidate solution, they will *enumerate* many potential solution and take the shortest/closest one.

The first one (SimpleEnum) is not particularly used in practice, nor very much studied in the literature because it is always slower than the former (FinckePohstEnum). But we find it to be a valuable pedagogical step.

2 Simple Enumeration

The simple enumeration algorithm generalizes simple-rounding, and when the parameter $\ell = 1$, the two algorithms are the same. This generalized algorithm again reduces the question to the lattice $\mathbb{Z}^n$, where finding a close point is as easy as rounding each coordinate. Yet, because this transformation from the lattice L to the lattice $\mathbb{Z}^n$ also distorts the space $\mathbb{R}^n$, we will now consider not only the closest integer (by rounding), we will take the set of all nearby integers.

Algorithm 1: SimpleEnum$(\mathbf{B}, \ell, \mathbf{t})$: Simple Enumeration Algorithm

Input : A basis $\mathbf{B} \in \mathbb{Q}^{n \times n}$ of a full rank lattice Λ, a target $\mathbf{t} \in \mathrm{Span}_{\mathbb{Q}}(L)$, and a side length $\ell \geq 1$.

Output: $\mathbf{c} \in L$ a closest lattice point to $\mathbf{t}$ that lies in $\mathbf{t} + \ell \cdot \mathcal{P}(\mathbf{B})$

$\mathbf{c} = (\infty, \infty, \ldots, \infty)$
$\mathbf{t}' \leftarrow \mathbf{B}^{-1} \cdot \mathbf{t}$
for $\mathbf{v}' \in \mathbb{Z}^n \cap (\mathbf{t}' + [-\ell/2, \ell/2)^n)$ **do**
  $\mathbf{v} = \mathbf{B} \cdot \mathbf{v}'$
  if $\|\mathbf{v} - \mathbf{t}\| < \|\mathbf{c} - \mathbf{t}\|$ **then**
    $\mathbf{c} \leftarrow \mathbf{v}$
  end
end
return $\mathbf{c}$

Note that enumerating integer points in an interval $t + [-\ell/2, \ell/2)$ simply consists of running a for loop from $\lceil t - \ell/2 \rceil$ to $\lceil t + \ell/2 \rceil - 1$. To do so in arbitrary dimension n, one could nest n such for loops, which is inconvenient to write down; technically this would require writing down a different algorithm for each dimension n. Recursive programming techniques allow to solve this issue. For convenience, we will simply write a for loop enumerating integer vectors in a hypercube $\mathbb{Z}^n \cap (\mathbf{t}' + [-\ell/2, \ell/2)^n)$.[1]

LEMMA 1 *Algorithm SimpleEnum is correct, that is, SimpleEnum*$(\mathbf{B}, \ell, \mathbf{t})$ *outputs a vector* $\mathbf{c} \in L$ *such that*

$$\mathbf{c} \in \underset{\mathbf{v} \in L \cap \mathbf{t} + \ell \cdot \mathcal{P}(\mathbf{B})}{\arg\min} \|\mathbf{v} - \mathbf{t}\|.$$

Note that because there may be more than one closest vector, we write $\mathbf{c} \in \arg\min$ and not $\mathbf{c} = \arg\min$.

PROOF: First, except before the first loop, $\mathbf{c}$ is always equal to $\mathbf{B}$ times an integer vector. Because $\ell \geq 1$, the loop is iterated at least once. Hence, the final value of $\mathbf{c}$ is in the lattice L. Next, we argue that the visited vectors $\mathbf{v}$ are exactly those belonging to $L \cap \mathbf{t} + \ell \cdot \mathcal{P}(\mathbf{B})$. This is merely a matter of rewriting

$$\mathbf{B} \cdot (\mathbb{Z}^n \cap (\mathbf{t}' + [-\ell/2, \ell/2)^n)) = (\mathbf{B} \cdot \mathbb{Z}^n) \cap \left(\mathbf{B} \cdot \mathbf{t}' + \ell \cdot \mathbf{B} \cdot [-1/2, 1/2)^n)\right) = L \cap (\mathbf{t} + \ell \cdot \mathcal{P}(\mathbf{B})).$$

□

COROLLARY 2 *Algorithm SimpleEnum*$(\mathbf{B}, \ell, \cdot)$ *solves* α*-BDD up to* $\alpha = \min\{1/2, \ell \cdot \nu(\mathcal{P}(\mathbf{B}))/\lambda_1(L)\}$.

PROOF: By the definition of α-BDD, we are looking for the unique lattice point $\mathbf{v}$ in the ball of radius $r = \alpha \cdot \lambda_1(L) \leq \ell \cdot \nu(\mathcal{P}(\mathbf{B}))$ centered at $\mathbf{t}$. By definition of the inner radius $\nu(\mathcal{P}(\mathbf{B}))$, it holds that the ball of radius $\alpha\lambda_1(L)$ is contained in the parallelepiped $\ell \cdot \mathcal{P}(\mathbf{B})$. □

By a similar reasoning, we can also claim that for large enough ℓ, SimpleEnum solves Exact-CVP.

COROLLARY 3 *If* $\ell \cdot \nu(\mathcal{P}(\mathbf{B})) \geq \mu(L)$ *for a basis* $\mathbf{B}$ *of* L*, then Algorithm SimpleEnum*$(\mathbf{B}, \ell, \cdot)$ *solves Exact-CVP in* L.

2.1 SVP variant

We note that by setting $\mathbf{t} = 0$, we can enumerate lattice vectors close to the origin, and hence find a short vector rather than a close vector. This requires the mild modification of ignoring $\mathbf{v}' = 0$ in the main loop; we we call this variant $\mathsf{SimpleEnum}^{\mathsf{SVP}}(\mathbf{B}, \ell)$. Using once again the same reasoning as for Corollary 3, we can claim

COROLLARY 4 *If* $\ell \cdot \nu(\mathcal{P}(\mathbf{B})) \geq \lambda_1(L)$ *for a basis* $\mathbf{B}$ *of* L*, then Algorithm SimpleEnum*$^{SVP}(\mathbf{B}, \ell)$ *solves Exact-SVP in* L.

[1] This kind abuse of notation should not be used without justification, as one may accidentally hide complexity and difficulty by doing for loops over arbitrary sets.

2.2 Complexity

LEMMA 5 *The complexity of* $\mathsf{SimpleEnum}(\cdot, \ell, \cdot)$ *is of* $O(n^3 + n^2 \cdot \lceil \ell \rceil^n)$ *arithmetic operations. Its bit complexity is* $\lceil \ell \rceil^n$ *up to a polynomial factor in the size of the input.*

PROOF: The $O(n^3)$ term accounts for the cost of matrix inversion at the $\mathbf{t}' \leftarrow \mathbf{B}^{-1} \cdot \mathbf{t}$ step. The remaining term is the cost of the loop. Each iteration costs $O(n^2)$ arithmetic operations, where the most costly step is the matrix-vector product $\mathbf{v} = \mathbf{B} \cdot \mathbf{v}'$. At last, we note that the loop iterates over at most $\lceil \ell \rceil^n$ elements. □

One can further remark that this asymptotic upper bound (mind the big Oh!) is actually rather tight, as we can also lower-bound the number of loops by $\lfloor \ell \rfloor^n$. For integers ℓ the number of loops is therefore always the same independently of $\mathbf{t}$, but for non-integral ℓ it does vary a bit.

COROLLARY 6 *Up to polynomial factor in the size of the input:*

- $\mathsf{SimpleEnum}(\mathbf{B}, \lambda_1(L), \cdot)$ *solves Exact-CVP (and therefore BDD) in time* $\lceil \mu(L)/\nu(\mathcal{P}(\mathbf{B})) \rceil^n$,
- $\mathsf{SimpleEnum}^{SVP}(\mathbf{B}, \mu(L))$ *solves Exact-SVP in time* $\lceil \lambda_1(L)/\nu(\mathcal{P}(\mathbf{B})) \rceil^n$.

Not that running the above requires knowing $\lambda_1(L)$ and $\mu(L)$ in advance. These can be known or approximately known in certain scenario, but what can we do if they are not? A crude upper bound can be given in terms of the input basis: $\lambda_1(L)$ is upper bounded by the length of the smallest vector of the input basis, while $\mu(L)$ is upper bounded by $\frac{1}{2}\sum \|\mathbf{b}_i\|$. But for a basis with arbitrarily long basis vectors, this can give an arbitrarily loose upper-bound on the actual quantity.

In the case of SVP, one can make do without knowledge of $\lambda_1(L)$ without losing much efficiency: one can start trying with $\ell = 1$, and increase ℓ by a factor $(1 + 1/n)$ until a vector of length less than $\ell \cdot \nu(L)$ is found.

3 Fincke-Pohst Enumeration

We now discuss an enumeration algorithm that generalizes the principle of the Nearest-Plane algorithm. It will proceed recursively by projection onto lower-dimensional lattices. It was invented by Fincke and Pohst, and is sometime referred simply as *lattice enumeration*, or qualified as a *Branch and Bound* type of algorithm.

Our presentation might be quite different from the literature in several ways. It is usually described in a much more explicit way using coordinates, being closer to actually implemented machine code. A more fundamental difference is that the version given here is a "breadth-first" approach, while the version used in practice is "depth-first". This difference will be the object of a detailed exercise, but we can already say that the latter is preferable is it consumes much less memory while having the same running time. We nevertheless find the former conceptually simpler.

In more detail, the algorithm enumerates lattice points in a euclidean ball, and is therefore readily fit to solve BDD/CVP/SVP problems in the ℓ_2 norm. But it can also be tweaked to work for arbitrary norm, by over-approximating the associated arbitrary ball by a euclidean ball.

The principle used in each recursive call is that for some projection map $\pi_{\mathbf{v}}^{\perp}$ orthogonal to a vector $\mathbf{v}$, the projection of lattice points whose distance to a target $\mathbf{t}$ is less than r is also at a distance less than r from the projection of $\mathbf{t}$. In equations:

$$\pi_{\mathbf{v}}^{\perp}(L \cap (\mathbf{t} + r\mathfrak{B}_2^n)) \subseteq \pi_{\mathbf{v}}^{\perp}(L) \cap (r\mathfrak{B}_2^{n-1} + \pi_{\mathbf{v}}^{\perp}(\mathbf{t})), \tag{1}$$

where $\mathfrak{B}_2^k$ is the unit euclidean ball in k dimensions (implicitly we are taking the ball to be in k dimensions on a k-dimensional subspace in $\mathbb{R}^n$). We can arbitrarily choose a primitive vector in L and will project the lattice orthogonally from this to get a lattice in $n-1$ dimensions. In Algorithm 2 below, we use a basis vector, but conceptually the algorithm is the same regardless of what primitive vectors are chosen at each step.

For a given vector $\mathbf{v}$ with orthogonal projection map $\pi_{\mathbf{v}}^{\perp}(\cdot)$, the algorithm will enumerate over the lattice $\pi_{\mathbf{v}}^{\perp}(L)$ to find the set $S' := \left\{\mathbf{s}' \in \pi_{\mathbf{v}}^{\perp}(L) : \|\pi_{\mathbf{v}}^{\perp}(\mathbf{t}) - \mathbf{s}'\|_2 \leq r\right\}$, i.e. the points whose distance is at most r from the target $\pi_{\mathbf{v}}^{\perp}(\mathbf{t})$. Then for every $\mathbf{s}' \in S'$, we find all lattice elements in the pre-image $(\pi_{\mathbf{v}}^{\perp})^{-1}(\mathbf{s}')$ whose distance to $\mathbf{t}$ is not bigger than r. Since $\mathbf{v}$ is orthogonal to each element of S', the distance criterion can be checked easily by only taking lattice vectors in $(\pi_{\mathbf{v}}^{\perp})^{-1}(\mathbf{s}')$ whose distance to $\mathbf{s}'$ is not greater than $\sqrt{r^2 - \|\mathbf{s}' - \pi_{\mathbf{b}_1}^{\perp}(\mathbf{t})\|_2^2}$. Finding all such lifts in $(\pi_{\mathbf{v}}^{\perp})^{-1}(\mathbf{s}')$, however, requires us to first calculate one lift of $\mathbf{s}'$ (denoted $\mathbf{x}$ in Algorithm 2) and then add integer multiples of $\mathbf{v}$ to it.

Algorithm 2: FinckePohstEnum$(\mathbf{B}, \mathbf{t}, r)$: Fincke-Pohst Enumeration Algorithm

Input : A basis $\mathbf{B} = (\mathbf{b}_1, \dots \mathbf{b}_n) \in \mathbb{Q}^{d \times n}$ of a lattice Λ, a target $\mathbf{t} \in \mathrm{Span}_{\mathbb{Q}}(L)$, and a radius $r \geq 0$.

Output: A list containing all vectors $\mathbf{x} \in L$ satisfying $\|\mathbf{x} - \mathbf{t}\|_2 \leq r$.

if $n = 1$ **then**
  return $\left\{z \cdot \mathbf{b}_1 \mid z \in \mathbb{Z}, \left\| \left(\frac{\langle \mathbf{0} - \mathbf{t}, \mathbf{b}_1 \rangle}{\|\mathbf{b}_1\|^2} + z \right) \mathbf{b}_1 \right\|_2 \leq r \right\}$
end
$S \leftarrow \{\}$
$\mathbf{B}' \leftarrow \left(\pi_{\mathbf{b}_1}^{\perp}(\mathbf{b}_2), \dots, \pi_{\mathbf{b}_1}^{\perp}(\mathbf{b}_n) \right)$
$S' \leftarrow$ FinckePohstEnum$(\mathbf{B}', \pi_{\mathbf{b}_1}^{\perp}(\mathbf{t}), r)$
for $\mathbf{s}' \in S'$ **do**
  $\rho \leftarrow \sqrt{r^2 - \|\mathbf{s}' - \pi_{\mathbf{b}_1}^{\perp}(\mathbf{t})\|_2^2}$
  $\mathbf{y} \leftarrow \mathbf{B}'^{-1} \cdot \mathbf{s}' \quad \in \mathbb{Z}^{n-1}$ // *the coordinates of* $\mathbf{s}'$ *in terms of* $\mathbf{B}'$
  $\mathbf{x} \leftarrow \mathbf{B} \cdot (0, \mathbf{y})^{\top} \quad \in L$ // *an arbitrary lift of* $\mathbf{s}'$ *to* L
  $Z \leftarrow \left\{ z \in \mathbb{Z} \mid \left\| \left(\frac{\langle \mathbf{x} - \mathbf{t}, \mathbf{b}_1 \rangle}{\|\mathbf{b}_1\|^2} + z \right) \mathbf{b}_1 \right\|_2 \leq \rho \right\}$
  $S \leftarrow S \cup \{\mathbf{x} + z\mathbf{b}_1 : z \in \mathbb{Z}\}$
end
return S

Lemma 7 *Algorithm 2 is correct. That is, the algorithm outputs the set* $S = L \cap (\mathbf{t} + r \cdot \mathfrak{B}_2)$.

Proof: The proof essentially follow the explanation given above. The case $n = 1$ is correct by construction. We then proceed by induction, assuming the algorithm is correct for dimension

$n-1$. One can see that $\mathbf{x} \in L$ is indeed a lift of $\mathbf{s}'$, that is $\pi^{\perp}_{\mathbf{b}_1}(\mathbf{x}) = \mathbf{s}'$. Then that $\mathbf{x} + Z \cdot \mathbf{b}_1$ is indeed the set of pre-images of $\mathbf{s}'$ whose norm is less than r, by invoking invoking pythagoras for the orthogonal vectors $\mathbf{b}_1 \perp \mathbf{s}'$. □

Note that Fincke-Pohst does not itself solve CVP or SVP but instead finds a large set of short or close vectors, and it only remains to pick the closest among them as done for CVP in Algorithm 3. The SVP variant is equally obvious.

Algorithm 3: FinckePohstCVP$(\mathbf{B}, \mathbf{t}, r)$: Fincke-Pohst CVP Algorithm

Input : A basis $\mathbf{B} = (\mathbf{b}_1, \dots \mathbf{b}_n) \in \mathbb{Q}^{n \times n}$ of a full rank lattice Λ, a target $\mathbf{t} \in \mathrm{Span}_{\mathbb{Q}}(L)$, and a radius $r \geq \mathrm{dist}(\mathbf{t}, L)$.

Output: A closest vector in L to the target $\mathbf{t}$.

$S' \leftarrow$ FinckePohstEnum$(\mathbf{B}, \mathbf{t}, r)$

return $\arg\min_{\mathbf{v} \in S'} \|\mathbf{v} - \mathbf{t}\|_2$

3.1 Complexity

To study the complexity of this algorithm, it is useful to think of the set of visited vectors in the various projected lattices organized in a *leveled tree*. Recall from the Gram-Schmidt Orthogonalization section of last lecture that we denote $\pi_i = \pi^{\perp}_{\mathbf{b}_1, \dots \mathbf{b}_{i-1}}$.

At the bottom of the tree is the trivial 0-dimensional lattice, with a single node. At level i the nodes are the elements of the set $S_i = \pi_{n-i}(L) \cap r\mathfrak{B}_2$. From a vector $\mathbf{v} \in S_i$ and $\mathbf{v}' \in S_{i+1}$ we put a directed edge from $\mathbf{v}$ to $\mathbf{v}'$ when $\pi_{n-i}(\mathbf{v}') = \mathbf{v}$. Note that by Equation (1), every node has one parent, i.e. any node at level $i+1$ is connected to one (and exactly one) node at level i, and is therefore connected to the root. Note however nodes may have zero, one, or several children.

The algorithm can be understood as a *breadth-first* exploration of this tree. One first constructs the set S_i recursively, and from there, one constructs S_{i+1} by constructing the children of each $\mathbf{v} \in S_i$; this is the set $\mathbf{x} + Z\mathbf{b}_1$ constructed in the loop of Algorithm 2. Constructing each child is algorithmically easy, and has complexity polynomial in the size of the input. The main factor of the complexity of the algorithm is therefore simply the size of the entire tree $\sum_{i=1}^{n} |S_i|$.

Note that we can upper bound the size of $|S_i|$ by using the generalization of Proposition 13 of Lecture 1, restated below for convenience.

Lemma 8 *Let L be a lattice of rank n. Then, for any radius $r \geq 0$ and any shift $\mathbf{t} \in \mathrm{Span}_{\mathbb{R}}(L)$, it holds that* $\frac{|r\mathfrak{B}^n \cap (L+\mathbf{t})|}{\mathrm{vol}(\mathfrak{B}^n)} \leq \frac{(r+2\mu(L))^n}{\det(L)}$.

That is, we have

$$|S_i| \leq \frac{(r + 2\mu(\pi_{n-i}(L)))^i}{\det(\pi_{n-i}(L))) \cdot \mathrm{vol}(\mathfrak{B}^i)}.$$

Note that the covering radius $\mu(L)$ of a lattice L can only decrease under projection, hence we can upper bound the numerator by $(r + 2\mu(L))^n$. Note further that the Gram-Schmidt orthogonalization of the basis $\pi_i(\mathbf{b}_i), \dots, \pi_i(\mathbf{b}_n)$ is simply $\mathbf{b}_i^\star, \dots, \mathbf{b}_n^\star$, hence $\det(\pi_{n-i}(L)) = \prod_{j=n-i}^{n} \|\mathbf{b}_j^\star\|$. At last, we recall that $\mathrm{vol}(\mathfrak{B}_2^i) \sim \frac{1}{\sqrt{\pi i}} \left(\frac{2\pi e}{i}\right)^{i/2}$ from Stirling's approximation. We conclude with the following.

THEOREM 9 *The number of nodes visited by* FinckePohstEnum$(\mathbf{B}, r, \cdot)$ *is at most*

$$\sum_{i=1}^{n} \frac{(r+2\mu(L))^i}{\prod_{j=n-i}^{n} \|\mathbf{b}_j^\star\|_2} \cdot \left(\frac{i}{2\pi e}\right)^{i/2} \cdot \sqrt{\pi \cdot i} \cdot (1+o(1)).$$

In particular, it is at most

$$O(n^{3/2}) \cdot \left(\frac{\sqrt{n} \cdot (r+2\mu(L))}{\sqrt{2\pi e} \cdot \min \|\mathbf{b}_i^\star\|_2}\right)^n.$$

3.2 Improvements

As mentioned at the beginning of the section, the above is not the standard form of Fincke-Pohst. We discuss below various enhancements of this algorithm. Those are clever tricks that matter a lot for practice but are hardly visible on the asymptotics.

1. The very first improvement is to switch from a breadth-first version of the algorithm to a depth-first version so as to avoid storing the large intermediate sets S_i, which results in a memory consumption essentially as large as the running time. See Exercise 3 on sheet 3. In fact, this technique drastically reduces memory consumption to barely more than the size of the input.

2. This variation leads to also unlock another opportunistic speed-up for solving SVP and CVP when the exact length is not known in advanced but only upper-bounded. Indeed, each time a solution below a certain radius r is found, one can dynamically decrease the enumeration radius to the length or distance of that vector.

3. With the trick above, we now would like to see the enumeration radius decrease as early as possible, and we can try to optimize our luck by wisely choosing in which order we visit the children of a given node.

4. On a lower implementation level, one can save a linear $\Theta(n)$ factor by carefully writing and maintaining the visited vector written in basis $\mathbf{B}$ and $\mathbf{B}^\star$ rather than writing them in the canonical basis of $\mathbb{R}^n$.

5. The best implementations also de-recursify the algorithm because such recursive function calls are in practice quite costly.

4 Toward Basis Reduction

In all the algorithms we have seen, the geometric shape of the input basis mattered a lot, either to improve the approximation factor of fast algorithms (SimpleRounding, NearestPlane), or to accelerate slow algorithm solving exact SVP and CVP (SimpleEnum, FinckePohstEnum). Focusing on the NearestPlane and FinckePohstEnum algorithms, we noted that

- to improve the approximation factor for approx-CVP via NearestPlane we want to minimize $\mu^{(2)}(\mathcal{P}(\mathbf{B}^\star)) = \frac{1}{2}\sqrt{\sum \|\mathbf{b}_i^\star\|_2^2} \leq \frac{\sqrt{n}}{2} \cdot \max \|\mathbf{b}_i^\star\|_2$

- to improve the approximation factor for BDD via NearestPlane we want to maximize $\nu^{(2)}(\mathcal{P}(\mathbf{B}^\star)) = \frac{1}{2}\min \|\mathbf{b}_i^\star\|_2$
- to speed up SVP/CVP/BDD via FinckePohstEnum we also want to maximize $\nu^{(2)}(\mathcal{P}(\mathbf{B}^\star)) = \frac{1}{2}\min \|\mathbf{b}_i^\star\|_2$

We also remark that solving SVP is equivalent to finding a basis $\mathbf{B}$ that minimizes $\mathbf{b}_1$ which is also equal to $\mathbf{b}_1^\star$. These objectives may sound opposite: for some tasks, we want the Gram-Schmidt norms to be small, sometimes to be large. But this is forgetting the invariant $\prod \|\mathbf{b}_i^\star\| = \det(L)$: making some $\mathbf{b}_i^\star$ small make the other ones larger. Because of this invariant, all these objective are actually aligned: what we want is for the Gram-Schmidt norm to be all *balanced*, all as close as possible to the root determinant $\det(L)^{1/n}$. This would optimize the min, the max and also the Euclidean $\sqrt{\sum \|\mathbf{b}_i^\star\|_2^2}$, according to the inequality of arithmetic and geometric means: $(\sum x_i)/n \geq (\prod x_i)^{1/n}$.

The case of SimpleRounding and SimpleEnum and the geometric control of $\mathcal{P}(\mathbf{B})$ are also related though a bit more complex, and will be discussed as well in further lectures.

Lagrange and Hermite algorithms

1 Introduction

In the previous lecture, we motivated the task of basis reduction: solving lattice problems via the Nearest-Plane algorithm or the Fincke-Pohst Enumeration algorithm was more successful or faster when the input basis was balanced, or more precisely when the Gram-Schmidt vectors were as closer in length to one another. More formally, we consider the *profile* of a basis as defined below, and keep in mind the rate of change of the associated function $\{1, \dots, n\} \to \mathbb{R}$ given by $i \mapsto \ell_i$.

Definition 1 *The profile of a basis* $\mathbf{B}$ *of a lattice of rank* n *is the sequence* $(\ell_1, \dots, \ell_n)$ *of logarithms of its Gram-Schmidt norms:*

$$\ell_i = \log \|\mathbf{b}_i^\star\|_2 \quad \text{for } i \in \{1 \dots n\}.$$

The task of lattice reduction is to change the basis $\mathbf{B}$ so as to make the resulting profile as *flat* as possible, in the sense that each of the ℓ_i are a similar magnitude.

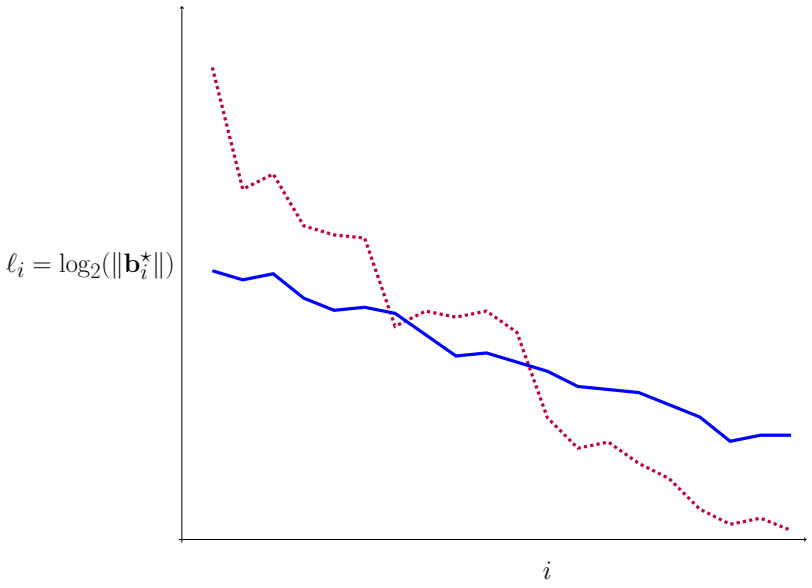

Figure 1: The profile of a basis in dimension 20 before (dashed) and after (plain) a reduction

The volume invariant $\prod \|\mathbf{b}_i^\star\|_2 = \det(L)$ gives us $\sum \ell_i = \log \det(L)$, and thus the area under the plot of $i \mapsto \ell_i$ must remain constant. Also, because the first Gram-Schmidt vector is also a basis vector $\mathbf{b}_1 = \mathbf{b}_1^\star$, a successful reduction of a lattice basis leads to finding a short basis vector. Typically, we start from bases with large vectors $\mathbf{b}_i$. In particular $\mathbf{b}_1$ is large and the later Gram-Schmidt norms are significantly smaller, that is, the profile decreases rapidly.

Note however that there are some lattices that do not admit flat profiles. For example, the lattice generated by $\left(\begin{smallmatrix} \varepsilon & 0 \\ 0 & 1/\varepsilon \end{smallmatrix}\right)$ for $0 < \varepsilon < 1$. For this lattice, there is no basis for which the shortest Gram-Schmidt norm is less than ε. What we can guarantee in general is that the profile does not *decrease* too sharply, but in general, we cannot control sharp *increase* of the profile. Or more

precisely, we can only control them relatively to the so-called successive minima, that we will introduce first.

In this lecture we will see Lagrange reduction, which reduces a 2-dimensional basis. Remarkably, this will always provide a shortest vector of that lattice in polynomial time. Such a Lagrange reduction algorithm will then allow us to generalize reduction notions and algorithms to higher dimensions, using the two dimensional algorithm on projected lattices. These are called *Hermite* and *LLL* reduction. We will prove that the LLL algorithm runs in polynomial time, and provides a solution to α-SVP for and approximation factor $\alpha = 2^{O(n)}$ exponential in the dimension n.

2 Successive Minima

Definition 2 (Successive Minima) *Let $L \subseteq \mathbb{R}^n$ be a rank $k \geq 1$ lattice. For $1 \leq i \leq k$, we define the i^{th} minima of L as*

$$\lambda_i(L) = \inf\{s \geq 0 : \dim(L \cap s\mathfrak{B}^n) \geq i\}.$$

Remark 3 We first note that for $i = 1$, the above definition of $\lambda_1 = \inf\{s \geq 0 : \dim(s\mathfrak{B}^n \cap L) \geq 1\}$ seems somewhat different from the original definition $\lambda_1(L) = \inf_{\mathbf{y} \in L \setminus \{\mathbf{0}\}} \|\mathbf{y}\|$. To see that the definitions are equivalent, note that $\dim(s\mathfrak{B}^n \cap L) \geq 1 \Leftrightarrow \exists \mathbf{y} \in L \setminus \{\mathbf{0}\}$ s.t. $\|\mathbf{y}\| \leq s$. From this, it is direct to see that both definitions yield exactly the same value.

By definition, it is clear that $\lambda_1(L) \leq \lambda_2(L) \leq \cdots \leq \lambda_k(L)$. We now show that successive minima are in fact well-defined, and that there are lattice vectors that attain them.

Lemma 4 *Let $L \subseteq \mathbb{R}^n$ be a $k \geq 1$ dimensional lattice. Then there exists linearly independent vectors $\mathbf{y}_1, \ldots, \mathbf{y}_k \in L$ such that $\|\mathbf{y}_i\| = \lambda_i(L)$. In particular, $\lambda_i(L) < \infty$ for all $i \in [k]$.*

Proof: Let $\mathbf{b}_1, \ldots, \mathbf{b}_k$ denote a basis for L. Let $R = \max_{1 \leq i \leq k} \|\mathbf{b}_i\|$. Clearly $\dim(R\mathfrak{B}^n \cap L) = \dim(L) = k$. Therefore, $\lambda_i(L) \leq R$ for all $i \in [k]$. Hence, if there exists $\mathbf{y} \in L$ such that $\|\mathbf{y}\| = \lambda_i(L)$, for any $i \in [k]$, we must have that $\mathbf{y} \in R\mathfrak{B}^n$.

We recursively choose $\mathbf{y}_1, \ldots, \mathbf{y}_k \in L \setminus \{\mathbf{0}\}$ as follows. Let $V_0 = \{\mathbf{0}\}$, and let $\mathbf{y}_1$ be a shortest vector in $(L \cap R\mathfrak{B}^n) \setminus V_0$. For i, $2 \leq i \leq k$, let $\mathbf{y}_i$ be the shortest vector in $L \cap R\mathfrak{B}^n \setminus V_{i-1}$ where $V_{i-1} = \mathrm{span}(\mathbf{y}_1, \ldots, \mathbf{y}_{i-1})$. We note that $\mathbf{y}_1, \ldots, \mathbf{y}_k$ exist since $L \cap R\mathfrak{B}^n$ is finite (by discreteness of L) and since $\dim(L \cap R\mathfrak{B}^n) = k$.

We claim that $\mathbf{y}_1, \ldots, \mathbf{y}_k$ are linearly independent and that $\|\mathbf{y}_i\| = \lambda_i(L)$, $i \in [k]$. Since each vector is chosen outside the span of the previous vectors, we have that $\mathbf{y}_1, \ldots, \mathbf{y}_k$ are linearly independent. Therefore $\dim(V_i) = \mathrm{span}(\mathbf{y}_1, \ldots, \mathbf{y}_i) = i$ for $i \in \{0, \ldots, k\}$.

Furthermore, by construction, it is clear that $\|\mathbf{y}_1\| \leq \|\mathbf{y}_2\| \leq \cdots \leq \|\mathbf{y}_k\|$. For $i \in [k]$, let $r_i = \|\mathbf{y}_i\|$. From here see that $\dim(r_i\mathfrak{B}^n \cap L) \geq \dim(V_i) = i$. Hence $r_i = \|\mathbf{y}_i\| \geq \lambda_i(L)$ by definition. We now show that $r_i \leq \lambda_i(L)$. For $i \in [k]$, and $0 < \varepsilon \leq r_i$, take $\mathbf{y} \in L \cap (r_i - \varepsilon)\mathfrak{B}^n$. We claim that $\mathbf{y} \in V_{i-1}$. If not, then by our choice of $\mathbf{y}_i$, we must have that $\|\mathbf{y}_i\| = r_i \leq \|\mathbf{y}\| \leq r_i - \varepsilon < r_i$, a clear contradiction. Therefore $\dim(L \cap (r_i - \varepsilon)\mathfrak{B}^n) \leq \dim(V_{i-1})) = i - 1$, and hence $r_i \leq \lambda_i(L)$ as needed. □

Note that the set of vectors constructed in such a way for $k = n$ is *not* necessarily a basis of L, but might instead generate only a a full-rank sublattice of L. An example of when this happens is the lattice $D_n = 2\mathbb{Z}^n \cup (2\mathbb{Z}^n + (1, 1, \ldots, 1))$ for $n \geq 5$: in that case we have $\lambda_1(D_n) = \lambda_2(D_n) = \cdots = \lambda_n(D_n) = 2$, but the lattice generated by the $\mathbf{y}_i$'s is exactly $2\mathbb{Z}^n$.

LEMMA 5 *For any lattice L of rank n we have $\prod_{i=1}^{n} \lambda_i^{(2)}(L) \geq \det(L)$.*

PROOF: Consider the lineraly independant vectors $\mathbf{y}_1, \ldots, \mathbf{y}_n$ as above, and note that they generate a full-rank sublattice L' of L, hence $\det(L) \leq \det(L')$. We also have $\det(L') \leq \prod_{i=1}^{n} \|\mathbf{y}_i\|_2$ and we conclude. □

3 Lagrange Reduction Algorithm

In this section, we will describe Lagrange reduction algorithm, which finds a basis that is as short as possible for a two-dimensional lattice.

DEFINITION 6 (LAGRANGE REDUCTION) *Let L be a 2-dimensional lattice. A basis $\mathbf{B} = (\mathbf{b}_1, \mathbf{b}_2)$ of L is called Lagrange Reduced if:*

- $\mathbf{b}_1$ *is a shortest vector of* L.
- $|\langle \mathbf{b}_1, \mathbf{b}_2 \rangle| \leq \frac{1}{2}\|\mathbf{b}_1\|_2^2$.

THEOREM 7 (WRISTWATCH LEMMA) *Every 2-dimensional lattice admits a Lagrange-Reduced basis.*

The mnemonic name "wristwatch" [1] refers to the Figure 2. Note that this is indeed a notion of reduction as we discussed in the introduction. Indeed, the basis provided by the Wristwatch lemma implies that $\|\mathbf{b}_2\|_2 \geq \|\mathbf{b}_1\|_2 = \|\mathbf{b}_1^\star\|_2$, where

$$\|\mathbf{b}_2\|_2^2 \leq \|\mathbf{b}_2^\star\|_2^2 + 1/4\|\mathbf{b}_1^\star\|_2^2.$$

In particular $\|\mathbf{b}_2^\star\|_2^2 \geq 3/4\|\mathbf{b}_1^\star\|_2^2$, or equivalently $\ell_2 \geq \ell_1 - \log\sqrt{4/3}$: the profile does not decrease too sharply.

The proof of this theorem is 'by algorithm', which means that we give an algorithm that, given any basis, computes a shortest basis as in the Wristwatch lemma. The proof then consists of showing that the algorithm terminates, and after termination indeed gives the correct basis.

Algorithm 1: Lagrange reduction algorithm

Input : A basis $(\mathbf{b}_1, \mathbf{b}_2)$ of a lattice L.
Output: A basis $(\mathbf{b}_1, \mathbf{b}_2)$ as in the Wristwatch lemma.

repeat
  swap $\mathbf{b}_1 \leftrightarrow \mathbf{b}_2$
  $k \leftarrow \lceil \frac{\langle \mathbf{b}_1, \mathbf{b}_2 \rangle}{\|\mathbf{b}_1\|_2^2} \rfloor$
  $\mathbf{b}_2 \leftarrow \mathbf{b}_2 - k\mathbf{b}_1$
until $\|\mathbf{b}_1\|_2 \leq \|\mathbf{b}_2\|_2$

PROOF: We will prove the lemma by showing Algorithm 1 terminates and is correct.

[1] Some readers may recognise the top 'strap' of the wristwatch as a fundamental domain of the action of $\mathrm{SL}_2(\mathbb{Z})$ on the complex upper half-plane.

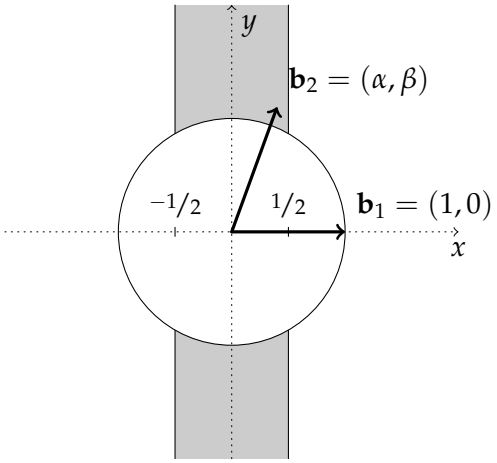

Figure 2: A picture of the wristwatch lemma. The lattice is scaled and rotated such that $\mathbf{b}_1 = (1, 0)$ is of unit length and lies on the x-axis. The second basis vector $\mathbf{b}_2$ must then be in one of the two gray areas.

Algorithm 1 terminates. This can be seen by the fact that $\|\mathbf{b}_1\|_2$ diminishes after every iteration in the repeat-loop. As L has a minimum non-zero length, and $\mathbf{b}_1$ and $\mathbf{b}_2$ are linearly independent, this necessarily means that the algorithm should terminate.

The resulting $\mathbf{b}_1, \mathbf{b}_2$ indeed form a basis of the lattice L. This can be seen by the fact that every operation in the loop (swap, row-addition) is a unimodular transformation on the basis, i.e., multiplication by a matrix in $\mathrm{GL}_2(\mathbb{Z})$. The final pair of vectors is therefore a basis, too.

We indeed have $\langle \mathbf{b}_1, \mathbf{b}_2 \rangle \leq \frac{1}{2}\|\mathbf{b}_1\|_2^2$. To prove this, we assume, without loss of generality (after scaling and rotating) that $\mathbf{b}_1 = (0, 1)$. Write $\mathbf{b}_2 = (\alpha, \beta)$. Then in the last iteration of the algorithm (omitting the swap), we essentially force that $|\beta| \leq 1/2$. As $\langle \mathbf{b}_1, \mathbf{b}_2 \rangle = \beta$, this proves our claim.

The vector $\mathbf{b}_1$ is a shortest (non-zero) lattice vector of L. As any vector in L can be written as $\mathbf{v} = m\mathbf{b}_1 + n\mathbf{b}_2$ with $m, n \in \mathbb{Z}$, our aim is to prove that $\|\mathbf{v}\|_2 = \|m\mathbf{b}_1 + n\mathbf{b}_2\|_2 \geq \|\mathbf{b}_1\|_2$. Again, we write $\mathbf{b}_1 = (0, 1)$ and $\mathbf{b}_2 = (\alpha, \beta)$.

For $n = 0$, this is clearly true, as then then $\mathbf{v} = m\mathbf{b}_1$ is a multiple of $\mathbf{b}_1$, which is always at least as long as $\mathbf{b}_1$ itself. The same holds for $m = 0$. For $n, m \neq 0$, we have

$$\|\mathbf{v}\|_2 = \|(n\alpha, m + n\beta)\|_2 = n^2\alpha^2 + (m + n\beta)^2 = n^2(\alpha^2 + \beta^2) + m^2 + 2mn\beta$$

$$\geq n^2 + m^2 - |mn| \geq \min(n^2, m^2) \geq 1 = \|\mathbf{b}_1\|_2.$$

Where the first inequality comes from the fact that $\alpha^2 + \beta^2 = \|\mathbf{b}_2\|_2^2 \geq 1$ and $|\beta| \leq 1/2$. □

LEMMA 8 *The Lagrange reduction algorithm terminates after* $25 + \max\left\{0, \log_2 \frac{\|\mathbf{b}_1\|_2}{\sqrt{\det L}}\right\}$ *iterations.*

PROOF: Without loss of generality, we may assume that the determinant of the lattice is 1, by scaling. Note that this also means that $\|\mathbf{b}_1^\star\|_2 \|\mathbf{b}_2^\star\|_2 = 1$. We divide the algorithm into two phases; the phase where $\|\mathbf{b}_1\|_2^2 \geq 2$, and the phase where $\|\mathbf{b}_1\|_2^2 < 2$.

- (Phase 1) As $\mathbf{b}_1 = \mathbf{b}_1^\star$, and $\|\mathbf{b}_1\|_2^2 \geq 2$, we must have that $\|\mathbf{b}_2^\star\|_2^2 \leq 1/2 \leq 1/4\|\mathbf{b}_1\|_2^2$. Denote $\mathbf{c}_1$ as being the 'next iteration' $\mathbf{b}_1$. Note that the $\mathbf{c}_1 = \mathbf{b}_2 - k\mathbf{b}_1$ satisfies $|\langle \mathbf{c}_1, \mathbf{b}_1 \rangle| \leq 1/2\|\mathbf{b}_1\|_2^2$

and $\langle \mathbf{c}_1, \mathbf{b}_2^\star \rangle = \langle \mathbf{b}_2, \mathbf{b}_2^\star \rangle = \|\mathbf{b}_2^\star\|_2^2$. It follows that

$$\|\mathbf{c}_1\|_2^2 \leq \frac{1}{4}\|\mathbf{b}_1\|_2^2 + \|\mathbf{b}_2^\star\|_2^2 \leq \frac{1}{2}\|\mathbf{b}_1\|_2^2$$

This means that the square length of $\mathbf{b}_1$ reduces by a factor $1/2$ every iteration. Therefore, the number of iterations in phase 1 is at most $\log_2(\|\mathbf{b}_1\|_2)$.

- (Phase 2) In this phase, $\|\mathbf{b}_1\|_2^2 < 2$. We distinguish the cases $\lambda_2(L)^2 \geq 2$ and $\lambda_2(L)^2 < 2$.

  i) In the first case the algorithm is done, because $\|\mathbf{b}_1\|_2^2 < 2 \leq \lambda_2(L)^2$. Namely, this means that $\mathbf{b}_1$ is a multiple of a shortest vector in L. But, as $(\mathbf{b}_1, \mathbf{b}_2)$ is a basis of L, $\mathbf{b}_1$ must be a shortest vector itself.

  ii) In the second case, we invoke Lemma 5 to obtain $\lambda_1(L) > 1/\sqrt{2}$. We conclude by a packing argument: the set P of points visited in this phase are in a ball of radius $R = \sqrt{2}$, and are separated by distance $\lambda_1(L)$. That is, the open balls of radius $r = \lambda_1(L)/2$ centered at each $p \in P$ are disjoints, furthermore they all are included in the centered ball of radius $R + r$. Therefore $|P| \leq (R+r)^2/r^2 = (1 + R/r)^2 \leq 5^2 = 25$. Because $\|\mathbf{b}_1\|$ is strictly decreasing at each iteration, each point in P can only be visited once.

□

Definition 9 *Let L be a full rank lattice L of dimension n. Then we define $\gamma(L)$ of this lattice as*

$$\gamma(L) := \frac{\lambda_1(L)^2}{\det(L)^{2/n}}$$

Definition 10 (Hermite constant) *The hermite constant γ_n is the supremum of γ over n-dimensional full rank lattices:*

$$\gamma_n := \sup_L \gamma(L)$$

Lemma 11 *The densest sphere packing in dimension 2 is attained by the hexagonal lattice H, which achieves $\gamma(H) = \sqrt{4/3}$.*

Proof: Let $H = \mathcal{L}((0,1), (\sqrt{3/4}, 1/2))$. Verify that this indeed has the required Hermite constant. We now have to show that any 2-dimensional lattice has a Hermite constant at least $\gamma(H)$. Let L be any lattice and let $(\mathbf{b}_1, \mathbf{b}_2)$ be a basis as in the Wristwatch lemma. Without loss of generality (see example sheet 4) we may assume $\mathbf{b}_1 = (0,1)$ and $\mathbf{b}_2 = (\alpha, \beta)$. Then $\lambda_1(L) = 1$, and $\det(L) = \det(\mathbf{B}) = \alpha$. As $|\beta| \leq 1/2$ we must have $1 \leq \alpha^2 + \beta^2 \leq \alpha^2 + \frac{1}{4}$. This directly implies $\alpha \geq \sqrt{3/4}$, which proves the claim. □

4 Hermite's Bound

Theorem 12 (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[2]; 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 13 *Let $L = L(\mathbf{B})$ a lattice generated by the basis $\mathbf{B}$. We will denote by $\mathbf{B}_{i:j}$ the basis $(\pi_i(\mathbf{b}_i), \ldots, \pi_i(\mathbf{b}_j))$, where $\pi_i(\cdot)$ is given as in Gram-Schmidt Orthogonalization, and by $\mathbf{B}^\star_{i:j}$ its Gram-Schmidt orthogonalization.*

Note that because $\pi_{i+a} \circ \pi_i = \pi_{i+a}$ for any $a \geq 0$, it holds that the $a+1$-th column vector of $\mathbf{B}^\star_{i:j}$ is exactly the $(i+a)$-th column vector of $\mathbf{B}^\star$. We can also write the following.

FACT 14 $\mathbf{B}^\star_{i:j} = (\mathbf{B}^\star_{i:k} | \mathbf{B}^\star_{k+1:j})$ *for any* $i \leq k \leq j$.

DEFINITION 15 *Let $L = \mathcal{L}(\mathbf{B})$ a lattice generated by the basis $\mathbf{B}$. We will denote $L_{i:j}$ for the lattice generated by $\mathbf{B}_{i:j} = (\pi_i(\mathbf{b}_i), \ldots, \pi_i(\mathbf{b}_j))$, i.e.,*

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

One easily check the following.

LEMMA 16 $\det(L_{i:j}) = \prod_{k=i}^{j} \|\mathbf{b}^\star_k\|_2$.

Algorithm 2: Hermite reduction algorithm

Input : A basis $\mathbf{B} = (\mathbf{b}_1, \ldots, \mathbf{b}_n)$ of a lattice L.
Output: A basis $\mathbf{B}$ such that $\|\mathbf{b}_1\|_2^2 \leq \gamma_2^{n-1} \cdot \det(L)^{2/n}$.

while $\exists i$ *such that* $\mathbf{B}_{i:i+1} = (\pi_i(\mathbf{b}_i), \pi_i(\mathbf{b}_{i+1}))$ *is not Lagrange reduced* **do**
  Find matrix $\mathbf{U} \in \mathbb{Z}^{2\times 2}$ such that $\mathbf{B}_{i:i+1}\mathbf{U}$ is Lagrange reduced
  $(\mathbf{b}'_i, \mathbf{b}'_{i+1}) \leftarrow (\mathbf{b}_i, \mathbf{b}_{i+1})\mathbf{U}$
  $\mathbf{B} \leftarrow (\mathbf{b}_1, \ldots, \mathbf{b}_{i-1}, \mathbf{b}'_i, \mathbf{b}'_{i+1}, \mathbf{b}_{i+2}, \ldots, \mathbf{b}_n)$
end
return $\mathbf{B}$

PROOF: (of Hermite's theorem)
First, we prove the correctness of Algorithm 2, assuming it terminates. After that, we will show that the algorithm does in fact terminate. 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 $\mathbf{b}^\star_i, \mathbf{b}^\star_{i+1}$ satisfies $\|\mathbf{b}^\star_i\|_2 \leq \gamma_2 \|\mathbf{b}^\star_{i+1}\|_2$. Using inductive reasoning, one can conclude that $\|\mathbf{b}_1\|_2 = \|\mathbf{b}^\star_1\|_2 \leq \gamma_2^{i-1} \|\mathbf{b}^\star_i\|_2$. Multiplying together for all i and using the determinant formula and the triangular number formula, yields

$$\|\mathbf{b}_1\|_2^n \leq \prod_{i=1}^{n} \gamma_2^{i-1} \|\mathbf{b}^\star_i\|_2 = \gamma_2^{\frac{n(n-1)}{2}} \det(L).$$

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

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 $\|\mathbf{b}_i^\star\|_2 \leq \gamma_2 \|\mathbf{b}_{i+1}^\star\|_2$. After Lagrange-reduction, we have $\|\mathbf{b}_i^\star\|_2^2 \leq \gamma_2 \cdot \det(\mathbf{B}_{i:i+1}) = \gamma_2 \|\mathbf{b}_i^\star\|_2 \|\mathbf{b}_{i+1}^\star\|_2$, where the last equality comes from Lemma 16. Dividing out appropriately yields $\|\mathbf{b}_i^\star\|_2 \leq \gamma_2 \|\mathbf{b}_{i+1}^\star\|_2$. So Lagrange-reduction indeed 'resolves' the wrong-way inequality $\|\mathbf{b}_i^\star\|_2 > \gamma_2 \|\mathbf{b}_{i+1}^\star\|_2$.

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 $\|\mathbf{b}_1\|_2 > \gamma_2 \|\mathbf{b}_2^\star\|_2$, we know that $\|\mathbf{b}_1\|_2^2 > \gamma_2 \|\mathbf{b}_2^\star\|_2 \|\mathbf{b}_1\|_2 = \gamma_2 \cdot \det(L_{1:2})$ *before* the Lagrange reduction happens. However, after Lagrange reduction, we have $\|\mathbf{b}_1\|_2 \leq \gamma_2 \cdot \det(L_{1:2})$. So the new $\mathbf{b}_1$ has indeed a strictly smaller norm.
- $\|\mathbf{b}_1\|_2$ has a lower bound. Namely, $\lambda_1(L) \leq \|\mathbf{b}_1\|_2$.

Since L 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 $L_{2:n}$ alone (because no operations on $\mathbf{b}_1$ are done anymore). By the induction hypothesis, it must terminate. Thus, the entire algorithm on L 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 $\gamma_2^{n-1} \cdot \det(L)^{2/n}$. Thus,

$$\gamma(L) = \frac{\lambda_1(L)^2}{\det(L)^{2/n}} \leq \frac{\|\mathbf{b}_1\|_2^2}{\det(L)^{2/n}} \leq \frac{\gamma_2^{n-1} \cdot \det(L)^{2/n}}{\det(L)^{2/n}} = \gamma_2^{n-1}.$$

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

LLL Reduction

1 Introduction

The issue with Hermite's algorithm is that it may be terribly slow: there can be as many as $(I/\lambda_1(L))^n$ many intermediate values for $\mathbf{b}_1$, where I is the initial length $\|\mathbf{b}_1\|$. It can therefore be as large as exponential in the input size. This is without taking into account that there may be many more steps happening where $\mathbf{b}_1$ does not vary.

The idea of the Lenstra-Lenstra-Lovász (LLL) algorithm is to slightly relax the termination condition to make the algorithm (much!) faster. First, we present a weak version of the notion of an 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 are not necessarily bounded; it might still be possible that the numbers involved in the computation are too big to efficiently compute with. In the next section we present the true LLL algorithm, which resolves this problem.

2 The LLL algorithm

Let us start by defininig the targeted reduction notion.

Definition 1 (ε-Weakly LLL reduced) *Let γ_2 be the hermite constant over 2-dimensional full rank lattice. $\mathbf{B}$ is then said to be ε-WLLL reduced if*

$$\|\mathbf{b}_i^\star\|_2 \leq (\gamma_2 + \varepsilon)\|\mathbf{b}_{i+1}^\star\|_2 \quad \textit{for all } i.$$

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

Algorithm 1: ε-Weakly LLL-reduction algorithm

Input : A positive real $\varepsilon > 0$. A basis $\mathbf{B} = (\mathbf{b}_1, \ldots, \mathbf{b}_n) \in \mathbb{Q}^{n\times n}$ of a lattice L.
Output: An ε-WLLL reduced basis $\mathbf{B}$.

while $\exists i$ *such that* $\|\mathbf{b}_i^\star\|_2 > (\gamma_2 + \varepsilon)\|\mathbf{b}_{i+1}^\star\|_2$ **do**
  Find matrix $\mathbf{U} \in \mathbb{Z}^{2\times 2}$ such that $\mathbf{B}_{i:i+1}\mathbf{U}$ is Lagrange reduced
  $(\mathbf{b}_i', \mathbf{b}_{i+1}') \leftarrow (\mathbf{b}_i, \mathbf{b}_{i+1})\mathbf{U}$
  $\mathbf{B} \leftarrow (\mathbf{b}_1, \ldots, \mathbf{b}_{i-1}, \mathbf{b}_i', \mathbf{b}_{i+1}', \mathbf{b}_{i+2}, \ldots, \mathbf{b}_n)$
end
return $\mathbf{B}$

Note that the above algorithm only spends time on doing a Lagrange reduction whenever the norm of $\mathbf{b}_i^\star$ is significantly larger than what it could be. This 'significant amount' is decided by $\varepsilon > 0$. So, in fact, the above algorithm is exactly the Hermite reduction algorithm, but with the difference that it allows some 'slack' in the form of the parameter ε.

Theorem 2 *If $\mathbf{B} \in \mathbb{Z}^{n\times n}$, then Algorithm 1 terminates after poly$(n, \log\|\mathbf{B}\|_\infty, 1/\varepsilon)$ iterations.*

PROOF: Define $P(\mathbf{B}) = \prod_{i=1}^{n} \det(L_{1:i}) = \prod_{i=1}^{n} \|\mathbf{b}_i^\star\|_2^{n-i+1}$, to be the *potential*[1] of the current basis $\mathbf{B}$ where $\pi_1(\cdot)$ is the projection given as in Gram-Schmidt Orthogonalization and $L_{1:i} = \mathcal{L}(\pi_1(\mathbf{b}_1), \ldots, \pi_1(\mathbf{b}_i))$ (note that the notation $L_{1:i}$ hides a dependence in the choice of the basis). We will show that $P(\mathbf{B})$ decreases by a constant factor in each iteration and that $P(\mathbf{B})$ has a lower bound. This combined shows that the algorithm terminates within a polynomially bounded number of iterations.

A single instance of Lagrange on $\mathbf{B}_{i:i+1}$ in an iteration only changes $\det(L_{1:i})$ and keeps all other determinants in the product of the potential fixed. It also maintains that $\mathbf{B} \in \mathbb{Z}^{n\times n}$. Before this iteration $\|\mathbf{b}_i^\star\|_2^2 \geq (\gamma_2 + \varepsilon)\det(L_{i:i+1})$. After the Lagrange reduction we must have $\|\mathbf{b}_i^\star\|_2^2 \leq \gamma_2 \det(L_{i:i+1})$. Therefore, after this Lagrange reduction, $\det(L_{1:i}) = \prod_{k=1}^{i} \|\mathbf{b}_k^\star\|_2$ must at least be diminished by a factor $\sqrt{\frac{\gamma_2}{\gamma_2+\varepsilon}}$. Thus, also the potential $P(\mathbf{B})$ must be reduced by (at least) this factor $f = \sqrt{\frac{\gamma_2}{\gamma_2+\varepsilon}}$.

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

The initial potential $P(\mathbf{B})_{\text{init}}$ (that is, the potential before running the algorithm) is bounded by $\prod_{i=1}^{n} \|\mathbf{b}_i\|_2^{n-i+1} \leq \|\mathbf{B}\|_\infty^{n(n+1)/2}$, where $\|\mathbf{B}\|_\infty := \max_i \|\mathbf{b}_i\|_2$. After N iterations, we know that

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

This means that the algorithm is surely terminated whenever $f^N \cdot \|\mathbf{B}\|_\infty^{n(n+1)/2} \leq 1$. Equivalently, it is 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. □

Polynomial time yet? However, it is not enough to prove that the number of iterations is bounded, in order to show that Algorithm 1 is truly poly-time. Once again, the time spent computing with the (rational) numbers in the algorithm should also be taken into consideration; without more analysis we don't know whether the rational numbers occurring in $\mathbf{B}^\star$ 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 bounded in any way. We will see that bounding the integer coefficients requires a modification of the algorithm, namely the 'true' LLL algorithm.

Lemma 3 below shows that the denominators of the entries in $\mathbf{B}^\star$ are bounded by $\det(\mathcal{L}(\mathbf{B}))^2$, so we are left only with the task of showing that the numerators are also not too large.

LEMMA 3 *If $\mathbf{B} \in \mathbb{Z}^{n\times m}$ with $n \geq m$ has dimension m, then $\mathbf{B}^\star \in \frac{1}{\det(\mathbf{B}^T\mathbf{B})}\mathbb{Z}^{n\times m}$.*

PROOF: We prove this by induction on the number of vectors in our basis (note that the length of the vectors remains the same). Let $\mathbf{B} = (\mathbf{b}_1, \ldots, \mathbf{b}_n)$ be a basis. If $n = 1$, then $\mathbf{B}^\star$ has integer

[1] The word 'potential' is used in the literature, possibly to draw similarities with the 'potential' in classical mechanics, where an object has potential energy at the top of a hill which decreases as it rolls down the hill. In particular, it does not roll up the hill. The idea here is that as the algorithm progresses, there will be less and less *potential* in the basis, as if it were rolling down a hill towards an LLL reduced basis at the bottom.

coefficients and we are done. Write $\mathbf{C} = (\pi_2(\mathbf{b}_2), \ldots, \pi_2(\mathbf{b}_n))$, a projected basis. We make the inductive hypothesis that the statement holds for integer bases of dimension $n-1$. In order to apply this induction hypothesis on $\mathbf{C}$, we require that $\mathbf{C}$ is in $\mathbb{Z}^{n\times(n-1)}$. In general this is not true but we know that for any $\mathbf{v} \in \mathbb{Z}^n$, the projection $\pi_2(\mathbf{v}) = \mathbf{v} - \frac{\langle \mathbf{v}, \mathbf{b}_1 \rangle}{\|\mathbf{b}_1\|_2^2} \cdot \mathbf{b}_1 \in \frac{1}{\|\mathbf{b}_1\|_2^2}\mathbb{Z}^n$. So we can scale up by the integer $d := \|\mathbf{b}_1\|^2$ so that $d\mathbf{C} \in \mathbb{Z}^{n\times(n-1)}$, and we can apply the induction hypothesis:

$$(d\mathbf{C})^\star = d\mathbf{C}^\star \in \frac{1}{\det(\mathbf{C}^T\mathbf{C})}\mathbb{Z}^{n\times(n-1)} \Rightarrow \mathbf{C}^\star \in \frac{1}{d\det(\mathbf{C}^T\mathbf{C})}\mathbb{Z}^{n\times(n-1)}$$

Now use that $\mathbf{C}$ has rank $n-1$, and therefore $\det(\mathbf{C}^T\mathbf{C}) = \prod_{i=2}^n \|\mathbf{b}_i^\star\|_2^2$. Every basis vector of $\mathbf{C}$ is orthogonal to $\mathbf{b}_1$, so $\mathbf{B}^\star = (\mathbf{b}_1 | \mathbf{C}^\star) \in \frac{1}{d\det(C^TC)}\mathbb{Z}^{n\times m}$. Use $d\det(\mathbf{C}^T\mathbf{C}) = \|\mathbf{b}_1\|^2\det(\mathbf{C}^T\mathbf{C}) = \det(\mathbf{B}^T\mathbf{B})$ and we are done. □

2.1 Size-Reduction

Definition 4 *A basis of a lattice* $\mathbf{B}$ *is said to be* size reduced *when*

$$\forall i < j \quad \left|\langle \mathbf{b}_i^\star, \mathbf{b}_j \rangle\right| \leq \frac{1}{2}\|\mathbf{b}_i^\star\|_2^2$$

The above definition is equivalent to saying that the off-diagonal of the upper triangular matrix of the Gram-Schmidt orthogonalization is bounded by 1/2. This notion of reduction allows us to estimate the size of the numerators of a basis. We will want to apply this reduction between every iteration of LLL to control the growth of coefficient, but at the same time, we do not wish to "disturb" the work of LLL on the potential: we want the size reduction to only affect $\mathbf{B}$, and not $\mathbf{B}^\star$.

Algorithm 2: SizeRed($\mathbf{B}$): Size-reduction algorithm

Input : A basis $\mathbf{B} = (\mathbf{b}_1, \ldots, \mathbf{b}_n) \in \mathbb{Q}^{n\times m}$ for $m \leq n$ of a lattice L.
Output: A size-reduced basis $\mathbf{C}$ of the same lattice L.

```
if n = 1 then
    return B
end
B' ← (b_1, ..., b_{n-1})
C' ← SizeRed(B')
v ← NearestPlane(C', π_{C'}(b_n))
Return (C' | b_n − v)
```

Lemma 5 *Algorithm 2 is correct and terminates in polynomial time.*

Proof: Termination in polynomial time is left to the reader. For correctness, we proceed by induction on the rank n of the basis; the statement is vacuously true for $n = 1$.

We start by proving that the output $\mathbf{C}$ is indeed a basis of the same lattice as $\mathbf{B}$. Let us assume by induction that $\mathbf{C}'$ is a basis of the same lattice as $\mathbf{B}'$; that is, $\mathbf{C}' = \mathbf{B}' \cdot \mathbf{U}'$ for some unimodular matrix $\mathbf{U}' \in \mathrm{GL}_n(\mathbb{Z})$. By construction, $\mathbf{v}$ is in the lattice generated by $\mathbf{C}'$, so $\mathbf{v} = \mathbf{C}' \cdot \mathbf{x}$ for some

$\mathbf{x} \in \mathbb{Z}^{n-1}$. One chan check that the returned basis writes as $\mathbf{C} = \left(\begin{smallmatrix} \mathbf{U} & 0 \\ \mathbf{0} & 1 \end{smallmatrix}\right) \cdot \left(\begin{smallmatrix} \mathbf{I} & 0 \\ -\mathbf{x} & 1 \end{smallmatrix}\right) \mathbf{B}$. Note that $\left(\begin{smallmatrix} \mathbf{U} & 0 \\ \mathbf{0} & 1 \end{smallmatrix}\right) \cdot \left(\begin{smallmatrix} \mathbf{I} & 0 \\ -\mathbf{x} & 1 \end{smallmatrix}\right)$ is integer and has determinant 1, it is therefore a unimodular transformation: $\mathbf{C}$ is indeed a basis of L.

Now, regarding the size-reduction property, the induction hypothesis gives $|\langle \mathbf{c}_i^\star, \mathbf{c}_j \rangle| \leq \frac{1}{2}\|\mathbf{c}_i^\star\|_2^2$ for all $i < j < n$, and it remains to prove it for $i < j = n$, where by construction we have $\mathbf{c}_n = \mathbf{b}_n - \mathbf{v}$. By the specification of the NearestPlane algorithm, we have that $|\langle \pi_{\mathbf{C}'}(\mathbf{b}_n) - \mathbf{v}, \mathbf{c}_i^\star \rangle| \leq \frac{1}{2}\|\mathbf{c}_i^\star\|_2^2$ for all i. Furthermore, $\mathbf{c}_i^\star$ belongs to the span of $\mathbf{C}'$, so $\langle \pi_{\mathbf{C}'}(\mathbf{b}_n), \mathbf{c}_i^\star \rangle = \langle \mathbf{b}_n, \mathbf{c}_i^\star \rangle$. □

As mentioned above, an important fact is that size-reducing a basis does not affect its Gram-Schmidt basis and therefore it does not affect the potential.

LEMMA 6 *Let* $\mathbf{B} \in \mathbb{Q}^{n \times m}$, *where* $m \leq n$, *and let* $\mathbf{C} = \mathit{SizeRed}(\mathbf{B})$. *Then,* $\mathbf{C}^\star = \mathbf{B}^\star$. *Furthermore* $P(\mathbf{C}) = P(\mathbf{B})$.

PROOF: We proceed again by induction on the dimension n of the basis. When $n = 1$, we have $\mathbf{c}_1^\star = \mathbf{b}_1^\star$. Now assume that the statement is true for bases with $n-1$ vectors. Let $\mathbf{B}'$, $\mathbf{C}'$, $\mathbf{v}$ be as in Algorithm 2. In particular $\mathbf{B}'^\star = \mathbf{C}'^\star$, and therefore $\pi_{\mathbf{B}'}^\perp = \pi_{\mathbf{C}'}^\perp$. We are finished when we show that

$$\mathbf{c}_n^\star := \pi_{\mathbf{C}'}^\perp(\mathbf{c}) = \mathbf{b}_n^\star.$$

By construction, $\mathbf{v}$ belongs to the lattice with basis $\mathbf{C}'$. In particular $\pi_{\mathbf{C}'}^\perp(\mathbf{v}) = \mathbf{0}$. Hence

$$\mathbf{c}_n^\star = \pi_{\mathbf{C}'}^\perp(\mathbf{c}_n) = \pi_{\mathbf{C}'}^\perp(\mathbf{b}_n - \mathbf{v}) = \pi_{\mathbf{C}'}^\perp(\mathbf{b}_n) = \pi_{\mathbf{B}'}^\perp(\mathbf{b}_n) = \mathbf{b}_n^\star.$$

□

Finally, we state a bound on the size of the vectors and Gram-Schmidt vector of a size-reduced basis as a function of its potential. The following Lemma is stated for integer lattices, but it can be applied to rational lattices by simply scaling the lattice up by the least common multiple of the numerators appearing in the input basis.

LEMMA 7 *A size-reduced basis* $\mathbf{B}$ *of an* integer *lattice* $L \subseteq \mathbb{Z}^n$ *has basis vectors* $\mathbf{b}_i$ *satisfying*

$$\log(\|\mathbf{b}_i\|), \log(\|\mathbf{b}_i^\star\|_2) \leq \mathit{poly}(n) \cdot \log(P(\mathbf{B})),$$

where $P(\mathbf{B}) = \prod_{i=1}^n \det(L_{1:i}) = \prod_{i=1}^n \|\mathbf{b}_i^\star\|_2^{n-i+1}$ *denotes the potential of the basis* $\mathbf{B}$ *as in the proof of Theorem 2.*

PROOF: As $\mathbf{B} \in \mathbb{Z}^{n \times n}$, $\det(L_{1:i}) \geq 1$ for all $i \leq n$, and therefore $\det(L_{1:i}) \leq P(\mathbf{B})$. Writing $\|\mathbf{b}_i^\star\|_2 = \det(L_{1:i})/\det(L_{1:i-1})$ directly leads to $\|\mathbf{b}_i^\star\|_2 \leq P(\mathbf{B})$. As the basis is size-reduced, we can write any basis vector $\mathbf{b}_j = \mathbf{b}_j^\star + \sum_{i=1}^{j-1} c_i \mathbf{b}_i^\star$ with $|c_i| \leq 1/2$. Therefore

$$\|\mathbf{b}_j\| \leq \sum_i |c_i| \|\mathbf{b}_i^\star\|_2 \leq (1 + n/2) P(\mathbf{B}).$$

□

Algorithm 3: LLL-reduction algorithm

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

Size-reduce the basis $\mathbf{B}$
while $\exists i$ *such that* $\|\mathbf{b}_i^\star\|_2 > (\gamma_2 + \varepsilon)\|\mathbf{b}_{i+1}^\star\|_2$ **do**
    Lagrange reduce $\mathbf{B}_{i:i+1}$ as in the WLLL algorithm.
    Size-reduce the new basis $\mathbf{B}$.
end
Return $\mathbf{B}$

2.2 LLL reduction and algorithm

Definition 8 *A basis $\mathbf{B}$ is ε-LLL reduced if it is both ε-WLLL reduced and size reduced.*

Usually we omit mentioning ε and instead we call such a basis LLL reduced. We also call the algorithm that achieves this reduction the LLL algorithm.

As for the WLLL algorithm, the correctness is trivial: the algorithm keeps running until the basis $\mathbf{B}$ is LLL reduced. The number of iterations of the main loop is the same as in WLLL, because the extra size-reduction step does not affect the Gram-Schmidt basis. The size of the manipulated rational is now controlled thanks to size-reduction.

But there is still a missing ingredient before we can conclude that LLL runs in polynomial time: we need to show that the Lagrange reduction algorithm is polynomial time. In the previous lecture, we only proved that the Lagrange reduction algorithm terminates after polynomially many interations in the size of its input 2 by 2 basis, but once again, one should raise the question of how large the coefficient gets during Lagrange reduction.

The lemmas that we have just demonstrated above are also applicable to Lagrange reduction. Namely, one can show that the Lagrange reduction algorithm maintains a size-reduced basis, and that the potential of the 2 by 2 basis on which it operates decreases. We leave the details to Exercise Sheet 5.

Given all the above, we reach our conclusion that LLL is polynomial time.

Theorem 9 *The LLL-basis reduction algorithm (as in Algorithm 3) with as input a basis $\mathbf{B} \subseteq \mathbb{Z}^{n \times n}$ runs in polynomial time.*

2.3 Properties of LLL reduced basis

Following the same argument as for Hermite's bound, one can prove that the first vector of an ε-LLL reduced basis satisfies $\|\mathbf{b}_1\| \leq (\gamma_2 + \varepsilon)^{\frac{n-1}{2}} \cdot \det(L)^{1/n}$. But other useful bounds can be stated as well. Before we get to them, let us state two more general lemmas that are not about LLL-reduced bases.

Lemma 10 *For any basis $\mathbf{B} = (\mathbf{b}_1, \ldots, \mathbf{b}_n)$ of a lattice L, we have*

$$\lambda_1(L) \geq \min_i \|\mathbf{b}_i^\star\|_2$$

PROOF: As the Babai fundamental domain $\mathcal{P}(\mathbf{B}^\star) = \mathbf{B}^\star[-1/2, 1/2)^n$ is tiling, it contains only one lattice point, which is zero. Verify that therefore $\mathbf{B}^\star(-1,1)^n \cap L = \{0\}$. Let $r = \max\{t > 0 \mid t\mathcal{B}_2^n \subseteq \mathbf{B}^\star(-1,1)^n\}$, then clearly $\lambda_1(L) \geq r$. It is easy to see that $r = \min_i \|\mathbf{b}_i^\star\|_2$, and the proof is finished. □

LEMMA 11 *For any lattice L, one has $\lambda_i(L_{k:n}) \leq \lambda_{i+k-1}(L)$.*

PROOF: It is enough to prove that $\lambda_i(L_{k:n}) \leq \lambda_{i+1}(L_{k-1:n})$ for any $k \in \{2, \ldots, n\}$ and $i \in \{1, \ldots, n-1\}$. Here we will prove this inequality for $k = 2$, and let the reader verify that the exact same proof can easily be amended for general $k \in \{2, \ldots, n\}$. So, our aim is to prove that $\lambda_i(L_{2:n}) \leq \lambda_{i+1}(L)$, for all $i \in \{1, \ldots, n-1\}$.

Let $\mathbf{v}_1, \ldots, \mathbf{v}_n \in L$ attain the successive minima of L. Let j be the smallest index such that $\pi_2(\mathbf{v}_j) \in \text{span}(\pi_2(\mathbf{v}_1), \ldots, \pi_2(\mathbf{v}_{j-1}))$, where $\pi_2(\cdot)$ is the projection onto the space orthogonal to vector $\mathbf{b}_1$ as given in Gram-Schmidt Orthogonalization for a basis $\mathbf{B}$. Set

$$\mathbf{w}_i = \begin{cases} \pi_2(\mathbf{v}_i) & \text{if} \quad i < j \\ \pi_2(\mathbf{v}_{i+1}) & \text{if} \quad i \geq j \end{cases}$$

Note that the $n-1$ vectors $\mathbf{w}_i$ are linearly independent elements of $L_{2:n}$. Furthermore, we have $\|\mathbf{w}_i\| \leq \max(\|\mathbf{v}_i\|, \|\mathbf{v}_{i+1}\|) \leq \max(\lambda_i, \lambda_{i+1}) \leq \lambda_{i+1}(L)$. This proves the claim. □

THEOREM 12 *Let $\mathbf{B} = (\mathbf{b}_1, \ldots, \mathbf{b}_n)$ be an ε-LLL reduced basis. Set $L = \mathcal{L}(\mathbf{B})$ and $\alpha = \gamma_2 + \varepsilon$. Then, we have*

1. $\|\mathbf{b}_1\| \leq \alpha^{\frac{n-1}{2}} \cdot \det(L)^{1/n}$ *[Root Hermite-factor Bound]*
2. $\|\mathbf{b}_1\| \leq \alpha^{n-1} \cdot \lambda_1(L)$ *[Approximation factor Bound]*
3. $\|\mathbf{b}_i^\star\|_2 \leq \alpha^{n-i} \cdot \lambda_i(L)$.
4. $\|\mathbf{b}_i\| \leq \alpha^{i-1} \cdot \|\mathbf{b}_i^\star\|_2 \leq \alpha^{n-1} \cdot \lambda_i(L)$ *[Approximation factor alike Bound]*
5. $\prod_{i=1}^n \|\mathbf{b}_i\| \leq \alpha^{\frac{n(n-1)}{2}} \cdot \det(L)$.

Note that in the following proof, for the inequalities (1.) to (3.) no size-reduction is needed. Thus, those inequalities hold in weakly LLL-reduced bases as well. The last two inequalities, however, do depend on size-reduction.
PROOF:

1. The exact same reasoning as in proving the Hermite bound, replacing γ_2 by α, applies here.
2. We have $\|\mathbf{b}_1\| \leq \alpha^{i-1} \cdot \|\mathbf{b}_i^\star\|_2 \leq \alpha^{n-1} \cdot \min_i \|\mathbf{b}_i^\star\|_2 \leq \alpha^{n-1} \cdot \lambda_1(L)$, where the first two inequalities follow from the basis being weakly LLL-reduced and the last inequality uses Lemma 10.
3. Note that $\mathbf{B}$ being Weakly LLL-reduced means that $\mathbf{B}_{i:n}$ is Weakly LLL-reduced, too. Therefore, by part (2.) of this theorem, $\|\mathbf{b}_i^\star\|_2 \leq \alpha^{n-i}\lambda_1(L_{i:n}) \leq \alpha^{n-i}\lambda_i(L)$. The last inequality uses Lemma 11.

4. As the second bound follows from (3.), we only need to prove the first bound. Write $\mathbf{b}_i = \mathbf{b}_i^\star + \sum_{j<i} c_j \mathbf{b}_j^\star$. As the basis is both size-reduced and Weakly LLL-reduced, we know that $|c_j| \leq 1/2$ and $\|\mathbf{b}_j^\star\|_2 \leq \alpha^{i-j}\|\mathbf{b}_i^\star\|_2$ for $j < i$. Therefore

$$\|\mathbf{b}_i\|^2 \leq \|\mathbf{b}_i^\star\|_2^2 + \frac{1}{4}\sum_{j<i}\|\mathbf{b}_j^\star\|_2^2 = \|\mathbf{b}_i^\star\|_2^2\left(1 + \frac{1}{4}\sum_{j<i}\frac{\|\mathbf{b}_j^\star\|_2^2}{\|\mathbf{b}_i^\star\|_2^2}\right) \leq \|\mathbf{b}_i^\star\|_2^2\left(1 + \frac{1}{4}\sum_{j<i}\alpha^{2(i-j)}\right)$$

$$\leq \|\mathbf{b}_i^\star\|_2^2\left(1 + \frac{1}{4}\sum_{k=1}^{i-1}\alpha^{2k}\right) \leq \alpha^{2(i-1)} \cdot \|\mathbf{b}_i^\star\|_2^2.$$

The last inequality follows from the following tedious calculations. As $\alpha^2 \geq \gamma_2^2 \geq 4/3$, we have $3\alpha^2 \geq 4$, so $4(\alpha^2 - 1) \geq \alpha^2$ and therefore $1 \geq \frac{\alpha^2}{4(\alpha^2-1)}$.

$$1 + \frac{1}{4}\sum_{k=1}^{i-1}\alpha^{2k} = 1 + \frac{\alpha^2}{4}\sum_{k=0}^{i-2}\alpha^{2k} = 1 + \frac{\alpha^2}{4}\cdot\frac{\alpha^{2(i-1)} - 1}{\alpha^2 - 1} \leq 1 + (\alpha^{2(i-1)} - 1) = \alpha^{2(i-1)}$$

5. This is left as Exercise in sheet 5.

□

2.4 LLL as pre-processing

All the above bounds are upper bounds on $\|\mathbf{b}_i\|_2$ and $\|\mathbf{b}_i^\star\|_2$, but the control of the slope also allows to state lower bounds on $\|\mathbf{b}_i^\star\|_2$. Those are equally useful to prove the complexity of certain task, by using LLL to pre-process the basis before running another basis-sensitive algorithm.

LEMMA 13 *If $\mathbf{B}$ is an ε-LLL reduced basis of L then $\|\mathbf{b}_i^\star\| \geq \alpha^{-i} \cdot \lambda_1(L)$.*

For example, pre-processing the basis with LLL before calling NearestPlane we obtain the following.

THEOREM 14 *For any $\alpha > \gamma_2$, there exists a polynomial time algorithm solving $(\alpha^{-n}/2)$-BDD in lattices of dimensions n.*

Similarly, pre-processing the basis with LLL before calling FinckePohstEnum we obtain the following.

THEOREM 15 *There exists an algorithm solving exact SVP in time $2^{O(n^2)}$ for lattices of dimension n.*

The above three statements are left as exercises. The third one requires re-using bounds in Exercise 2 of Sheet 3. **The old version of Exercise 2, Sheet 3 had mistakes. Be sure to use the up-to-date version**