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 B by (cheaply) transforming it to be solvable in one or more instances of problem A. When designing such a **reduction**, the unknown algorithm for solving A is called an **oracle** for A: we only assume that it solves A, 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 Λ, a target $\mathbf{t} \in \mathrm{Span}_{\mathbb{R}}(L)$.
Output: $\mathbf{v} \in L$ such that $\mathbf{e} = \mathbf{t} - \mathbf{v} \in \mathcal{P}(\mathbf{B})$

$\mathbf{t}' \leftarrow \mathbf{B}^{-1} \cdot \mathbf{t}$
$\mathbf{v}' \leftarrow (\lfloor t'_i \rceil)_{i\in\{1\dots n\}}$
$\mathbf{v} \leftarrow \mathbf{B} \cdot \mathbf{v}'$
return 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. This can be found on exercise sheet 2. □

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,*

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

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 $i \in \{1, \ldots, n\}$. $\square$
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. $\square$

4 Orthogonal Projections

Definition 8 *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 9 *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 $\mathrm{Span}_{\mathbb{R}}(L') \subset \pi_{\mathbf{v}}(\mathrm{Span}_{\mathbb{R}}(L))$. Since $\mathbf{v} \in \mathrm{Span}_{\mathbb{R}}(L)$, the dimension of $\pi_{\mathbf{v}}(\mathrm{Span}_{\mathbb{R}}(L))$ 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 10 (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 11 *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. □

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

LEMMA 12 *Let* $\mathbf{v}$ *be a primitive vector of a lattice* L*, define* $L' = \pi_{\mathbf{v}}^{\perp}(L)$*, 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 $\pi_{\mathbf{v}}^{\perp}$ 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 13 *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 12, 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:

$$\mathbf{e} \in \mathcal{P}(\mathbf{B}^\star) \Leftrightarrow \langle \mathbf{e}, \mathbf{b}_i^\star \rangle / \|\mathbf{b}_i^\star\|_2^2 \in [-1/2, 1/2) \text{ for all } i. \tag{5}$$

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

Claim 14 *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 $\mathbf{c}_i = \mathbf{b}_i^\star / \|\mathbf{b}_i\|_2^2$.

Lemma 15 *Algorithm 3 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).

Algorithm 3: $\mathsf{NearestPlane}(\mathbf{B}, \mathbf{t})$: 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 v

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 (5) 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.[3] □

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:

$$\mu^{(2)}(\mathcal{P}(\mathbf{B}^\star)) = \frac{1}{2}\sqrt{\sum_{i=1}^{n} \|\mathbf{b}_i^\star\|_2^2}. \tag{6}$$

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 and the characterization of orthogonal parallelepiped (5) we can cumpute the inner-radius:

$$\nu^{(2)}(\mathcal{P}(\mathbf{B}^\star)) = \frac{1}{2} \min_{i=1}^{n} \|\mathbf{b}_i^\star\|_2 \,. \tag{7}$$

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