Lattice cryptography: from cryptanalysis to New Foundations

Woerden, W.P.J. van

CHAPTER 8

Basis Reduction for Binary Codes

This chapter is an abbreviated version of the joint work 'An Algorithmic Reduction Theory for Binary Codes: LLL and more', with Thomas Debris-Alazard and Léo Ducas, published in IEEE Transactions on Information Theory.

8.1 Introduction

Codes and lattices share many mathematical similarities. A (linear) code $\mathcal{C} \subset \mathbb{F}_q^n$ is defined as a subspace of a vector space over a finite field, typically endowed with the Hamming metric, while a lattice $\mathcal{L} \subset \mathbb{R}^n$ is a discrete subgroup of a Euclidean vector space. They both found similar applications in information theory and computer sciences. For example, both can be used to perform error correction; on digital channels for codes, and on analogue channels for lattices.

Both objects also found applications in cryptography. Cryptosystems can be built relying either on the hardness of finding a close codeword or a close lattice point from a given target. And just as

for lattices, cryptography based on codes appears to be resistant to quantum computing.

The set of techniques for attacking those problems also have similarities, and some algorithms have been transferred in each direction: for example the Blum-Kalai-Wasserman [BKW03] algorithm has been adapted from codes to lattices [Alb+15], while the introduction of locally-sensitive hashing in code cryptanalysis [MO15] shortly followed its introduction in lattice cryptanalysis [Laa15].

It is therefore very natural to question whether all techniques used for codes have also been considered for lattices, and reciprocally. Beyond scientific curiosity, this approach can hint us at how complete each state of the art is, and therefore, how much trust we should put into cryptography based on codes and cryptography based on lattices.

Comparing both states of the art, it appears that there is a major lattice algorithmic technique that has no clear counterpart for codes, namely, *basis reduction*. Recall from Chapter 6 that lattice reduction attempts to find a basis with good geometric properties; in particular its vectors should be rather short and orthogonal to each others.

More specifically, the basis defines, via Gram-Schmidt Orthogonalization, a fundamental domain (or tiling) of the space, and a corresponding decoding algorithm. Decoding with this algorithm is the most favourable when these tiles are close to being square, *i.e.* when the Gram-Schmidt lengths are *balanced*. Basis reduction algorithms such as LLL aim at making the Gram-Schmidt lengths more balanced.

Certainly, the problem of finding short codewords has also been intensively studied in cryptanalysis with the Information Set Decoding (ISD) literature [Pra62; LB88; Ste88; Dum91; MMT11; BJMM12; MO15; BM18], but notions of basis reduction for lattices are more subtle than containing short vectors; as discussed above, a more relevant objective is to balance the Gram-Schmidt norms. There seem to be no analogue notions of Gram-Schmidt norms and basis reduction for codes, or at least they are not explicit nor associated with reduction algorithms. We are also unaware of any study of how such reduced bases would help with decoding tasks.

This observation leads to two questions. Is there an algorithmic reduction theory for codes, analogue to the one of lattices? And if so, can it be useful for decoding tasks?

8.1.1 Contributions

In this chapter we answer both questions positively, and set the foundation of an algorithmic reduction theory for codes. More specifically, we propose as main contributions:

1. the notion of an epipodal matrix $\mathbf{B}^+$ of the basis $\mathbf{B}$ of a binary code $\mathcal{C} \subset \mathbb{F}_2^n$ (depicted in Figure 8.1), playing a role analogue to the Gram-Schmidt Orthogonalisation $\tilde{\mathbf{B}}$ of a lattice basis,

2. a fundamental domain (or tiling) of $\mathcal{C}$ over $\mathbb{F}_2^n$ associated to this epipodal matrix, as an analogue to the rectangle parallelepipedic tiling for lattices,

3. a polynomial time decoding algorithm (SizeRed) effectively reducing points to this fundamental region, analogue to the Nearest Plane algorithm popularised by Babai [Bab86],

4. a relation between the geometric quality of the fundamental domain and the success probability for decoding a random error to the balance of the lengths of the epipodal vectors,

5. an adaptation of the seminal LLL reduction algorithm [LLL82] from lattices to codes, providing in polynomial time a basis with some epipodal length balance guarantees. Interestingly, this LLL algorithm for codes appears to be an algorithmic realisation of the classic bound of Griesmer [Gri60], in the same way that LLL for lattices realizes Hermite's bound.

These contributions establish an initial dictionary between reduction for codes and for lattices, summarised in Table 8.1.

Open Artefacts: Source code (`c++` kernel, with a `python` interface)[1].

8.1.2 Organisation

We introduce some preliminaries on binary linear codes in Section 8.2. In Section 8.3 we introduce the notion of orthopodality for binary vectors, and use this to define (orthopodal) projections and a

[1] Available at `https://github.com/lducas/CodeRed/`.

Table 8.1: A Lattice-Code Dictionary.

	Lattice $\mathcal{L} \subset \mathbb{R}^n$	Code $\mathcal{C} \subset \mathbb{F}_2^n$
Ambient Space	$\mathbb{R}^n$	$\mathbb{F}_2^n$
Metric	Euclidean $\|\mathbf{x}\|^2 = \sum x_i^2$	Hamming $\|\mathbf{x}\| = \#\{i \mid x_i \neq 0\}$
Support (element) Support (lattice/code)	$\mathbb{R} \cdot \mathbf{x}$ $\mathrm{Span}_{\mathbb{R}}(\mathcal{L})$	$\{i \mid x_i \neq 0\}$ $\{i \mid \exists \mathbf{c} \in \mathcal{C} \text{ s.t. } c_i \neq 0\}$
Sparsity Effect. Sparsity	$\det(\mathcal{L})$ $\det(\mathcal{L})$	2^{n-k} $2^{\#\mathrm{S}(\mathcal{C})-k}$
$\mathbf{x} \perp \mathbf{y}$ Projection $\pi_{\mathbf{x}}$	Orthogonality $\langle \mathbf{x}, \mathbf{y} \rangle = 0$ Ortho. Project. onto $\mathbf{x}$ $\mathbf{y} \mapsto \frac{\langle \mathbf{x}, \mathbf{y} \rangle}{\langle \mathbf{x}, \mathbf{x} \rangle} \cdot \mathbf{x}$	Orthopodality $\mathrm{S}(\mathbf{x}) \cap \mathrm{S}(\mathbf{y}) = \emptyset$ Punct. pattern $\mathbf{x}$ $\mathbf{y} \mapsto \mathbf{y} \wedge \mathbf{x}$
Auxiliary matrix Basis profile ℓ	Gram-Schmidt Orth. $\tilde{\mathbf{b}}_i = \mathbf{b}_i - \sum_{j<i} \frac{\langle \mathbf{b}_i, \tilde{\mathbf{b}}_j \rangle}{\langle \tilde{\mathbf{b}}_j, \tilde{\mathbf{b}}_j \rangle} \cdot \tilde{\mathbf{b}}_j$ $\ell_i = \|\tilde{\mathbf{b}}_i\|$	Epipodal matrix $\mathbf{b}_i^+ = \mathbf{b}_i \wedge \overline{(\mathbf{b}_1^+ \vee \cdots \vee \mathbf{b}_{i-1}^+)}$ $\ell_i = \|\mathbf{b}_i^+\|$
Fundamental domain $\mathcal{F}(\mathbf{B})$ Error correct. radius Average deco. dist. Worst deco. dist. Favourable decoding	Parallelepiped $\mathcal{P}(\tilde{\mathbf{B}})$ $\left\{\mathbf{x} \,\middle|\, \forall i \ \lvert\langle \mathbf{x}, \tilde{\mathbf{b}}_i \rangle\rvert \leq \frac{\ell_i^2}{2}\right\}$ $\min_i \ell_i^2/2$ $\sqrt{\frac{1}{12}\sum_i \ell_i^2}$ $\sqrt{\frac{1}{2}\sum_i \ell_i^2}$ balanced ℓ_i's	Prod. of Hamming balls[1] $\left\{\mathbf{x} \,\middle|\, \forall i \ \lvert\mathbf{x} \wedge \mathbf{b}_i^+\rvert \leq \frac{\ell_i}{2}\right\}$ $\min_i \lfloor (\ell_i - 1)/2 \rfloor$ $\approx \frac{n}{2} - \frac{1}{\sqrt{\pi}} \sum_i \sqrt{\lceil \ell_i/2 \rceil}$ $\sum_i \lfloor \ell_i/2 \rfloor$ balanced and odd ℓ_i's
Basis inequality Invariant	$\prod \|\mathbf{b}_i\| \geq \det(\mathcal{L})$ $\prod \|\tilde{\mathbf{b}}_i\| = \det(\mathcal{L})$	$\sum \lvert\mathbf{b}_i\rvert \geq \#\mathrm{S}(\mathcal{C})$ $\sum \lvert\mathbf{b}_i^+\rvert = \#\mathrm{S}(\mathcal{C})$
LLL balance LLL first length Corresponding bound	$\ell_i \leq \sqrt{4/3} \cdot \ell_{i+1}$ $\ell_1 \leq (\frac{4}{3})^{\frac{n-1}{2}} \det(\mathcal{L})^{\frac{1}{n}}$ Hermite's	$1 \leq \ell_i \leq 2 \cdot \ell_{i+1}$ $\ell_1 - \frac{\lceil \log_2 \ell_1 \rceil}{2} \leq \frac{n-k}{2} + 1$ Griesmer's [Gri60]

[1]This is not exactly correct when some epipodal length $\lvert\mathbf{b}_i^+\rvert$ are even. See tie-breaking in Section 8.4.

code analogue to the Gram-Schmidt basis. In Section 8.4 we give a translation of Babai's Nearest Plane algorithm to binary linear codes, and explain how its performance depends on the basis profile. In Section 8.5 we give a translation of the LLL algorithm to binary linear codes, and show how it results in a reasonably balanced basis profile. In the last Section 8.6 we discuss some future research directions.

8.2 Binary linear codes

We give a short introduction on the Hamming metric, binary linear codes, and how they relate to lattices.

Binary vector space

While lattices live in the continuous Euclidean vector space $\mathbb{R}^n$, codes live in the discrete non-euclidean vector space $\mathbb{F}_q^n$ for some prime q. In this work we will only consider the binary case $q = 2$, as it allows us to explain our general ideas, without too many technical difficulties. We identify $\mathbb{F}_2$ with the binary set $\{0, 1\}$, and thus interpret elements of $\mathbb{F}_2^n$ as binary vectors. We will use the standard boolean notations $\overline{\mathbf{x}}$, $\mathbf{x} \oplus \mathbf{y}$, $\mathbf{x} \wedge \mathbf{y}$, $\mathbf{x} \vee \mathbf{y}$, for respectively the bitwise NOT, the bitwise XOR (vector addition over $\mathbb{F}_2^n$), the bitwise AND, and the bitwise OR.[2] In contrast to most code-based literature the vectors $\mathbf{x} \in \mathbb{F}_2^n$ should be interpreted as column vectors, following the overall notation in this thesis.

Hamming metric

The most common metric in the code-based literature is the Hamming metric. The support $\mathrm{S}(\mathbf{x})$ of a vector $\mathbf{x} \in \mathbb{F}_2^n$ is the set of indices of its nonzero coordinates, and its Hamming weight $|\mathbf{x}| \in [\![0, n]\!]$ is the cardinality of its support:

$$\mathrm{S}(\mathbf{x}) := \{i \in [\![1, n]\!] \mid x_i \neq 0\}, \quad |\mathbf{x}| := \#\mathrm{S}(\mathbf{x}).$$

[2] In the code-based cryptography literature, for instance [COT16], the bitwise AND is often interpreted algebraically as a *star-product* or *Schur-product*, denoted $\odot$ or $\star$. We found the boolean notations more adapted to our geometric purposes, especially given that the bitwise OR also plays an important role.

The Hamming distance between two vectors $\mathbf{x}, \mathbf{y} \in \mathbb{F}_2^n$ is likewise given by $|\mathbf{x} \oplus \mathbf{y}| \in [\![0, n]\!]$. We will denote by $\mathcal{S}_w^n$ the Hamming sphere and by $\mathcal{B}_w^n$ the Hamming ball of radius w over $\mathbb{F}_2^n$, namely:

$$\mathcal{S}_w^n := \{\mathbf{x} \in \mathbb{F}_2^n : |\mathbf{x}| = w\}, \ \mathcal{B}_w^n := \{\mathbf{x} \in \mathbb{F}_2^n : |\mathbf{x}| \leq w\}.$$

Binary linear codes

A binary linear code $\mathcal{C}$ of length n and dimension k — for short, an $[n, k]$-code — is a subspace of $\mathbb{F}_2^n$ of dimension k. The ratio $R = k/n$ is called the *rate* of the code. Every linear code can be described either by a set of linearly independent generators (basis representation) or by a system of modular equations (parity-check representation). We will mostly consider the first representation for our purpose.

To build an $[n, k]$-code we may take any set of vectors $\mathbf{b}_1, \ldots, \mathbf{b}_k \in \mathbb{F}_2^n$ which are *linearly independent* and define:

$$\mathcal{C}(\mathbf{b}_1, \ldots, \mathbf{b}_k) := \left\{ \sum_{i=1}^{k} z_i \mathbf{b}_i : z_i \in \mathbb{F}_2 \right\} \qquad \text{(Basis representation)}$$

We say that $\mathbf{b}_1, \ldots, \mathbf{b}_k$ is a basis for the code $\mathcal{C} = \mathcal{C}(\mathbf{b}_1, \ldots, \mathbf{b}_k)$. Alternatively we will call the matrix $\mathbf{B} = [\mathbf{b}_1, \ldots, \mathbf{b}_k] \in \mathbb{F}_2^{n \times k}$ a basis or a generating matrix of the code $\mathcal{C} = \mathcal{C}(\mathbf{B})$.

Properties

An element $\mathbf{c} \in \mathcal{C}$ of a code $\mathcal{C}$ is called a *codeword*. The support of the code is defined as the union of the supports of all its codewords, which also implies the definition of an *effective length* $|\mathcal{C}| \leq n$ of a $[n, k]$-code $\mathcal{C}$:

$$\mathrm{S}(\mathcal{C}) := \bigcup_{\mathbf{c} \in \mathcal{C}} \mathrm{S}(\mathbf{c}), \quad |\mathcal{C}| := \#\mathrm{S}(\mathcal{C}).$$

Indeed, the code length n defined by the ambient space is rather extrinsic information, in particular extending a code $\mathcal{C}$ by padding a 0 to all codewords does not affect the geometry of the code, but it does affect the apparent length n. To avoid unnecessary technicalities we assume in the rest of this chapter that each code has full support, i.e., $|\mathcal{C}| = n$.

Note that in contrast to lattices, the span of a code basis is automatically discrete due to its ambient space $\mathbb{F}_2^n$, while for lattice we need to restrict to the $\mathbb{Z}$-span to obtain the discreteness inside $\mathbb{R}^n$. Similar to lattices, the discreteness allows us to define a first minimum, or minimal distance $d_{\min}(\mathcal{C})$ of a code, as the shortest Hamming weight of nonzero codewords, namely:

$$d_{\min}(\mathcal{C}) := \min\{|\mathbf{c}| : \mathbf{c} \in \mathcal{C} \text{ and } \mathbf{c} \neq \mathbf{0}\}.$$

Hard problems

The problem of finding such a minimum length codeword, the analogue of the Shortest Vector Problem, is known as the Minimum Codeword Problem.

Definition 116 (Minimum Codeword Problem (MCP)). *Given a basis $\mathbf{B}$ of a binary $[n,k]$-code $\mathcal{C}$, compute a nonzero codeword $\mathbf{x} \in \mathcal{C}$ of minimum weight, i.e., such that $|\mathbf{x}| = d_{\min}(\mathcal{C})$.*

Similarly the Closest Vector Problem has a direct analogue in codes, usually named the Nearest Codeword Problem.

Definition 117 (Nearest Codeword Problem (NCP)). *Given a basis $\mathbf{B}$ of a binary $[n,k]$-code $\mathcal{C}$, and a target $\mathbf{t} \in \mathbb{F}_2^n$, compute a codeword $\mathbf{x} \in \mathcal{C}$ of minimum distance to $\mathbf{t}$, i.e., such that $|\mathbf{t} \oplus \mathbf{x}|$ is minimal.*

Alternatively this is called *decoding* the code $\mathcal{C}$.

Systematic form

An usual way in code-based cryptography to decode random codes is to use bases in *systematic form*. A basis $\mathbf{B}$ of an $[n,k]$-code is said to be in *systematic form* or in *reduced row echelon form* if up to a permutation of its rows $\mathbf{B} = (\mathbf{I}_k; \mathbf{B}')$ where $\mathbf{I}_k$ denotes the identity of size $k \times k$ and $\mathbf{B}' \in \mathbb{F}_2^{(n-k)\times k}$. Such bases can be produced from any basis by doing a Gaussian elimination over the columns. Choosing the set of pivots iteratively (possibly at random among available ones), this can be done in time $O(nk^2)$.

8.3 Orthopodality and the epipodal matrix

Let us start by recalling the standard definition of Gram-Schmidt Orthogonalisation (GSO) over Euclidean vector spaces. Given a basis $[\mathbf{b}_1, \ldots, \mathbf{b}_n]$ of the Euclidean space $(\mathbb{R}^n, \|\cdot\|)$ its Gram-Schmidt orthogonalisation $(\tilde{\mathbf{b}}_1; \ldots; \tilde{\mathbf{b}}_n)$ is defined inductively by:

$$\tilde{\mathbf{b}}_i := \pi_i(\mathbf{b}_i) \quad \text{where } \pi_i : \mathbf{x} \mapsto \mathbf{x} - \sum_{j<i} \frac{\langle \mathbf{x}, \tilde{\mathbf{b}}_j \rangle}{\langle \tilde{\mathbf{b}}_j, \tilde{\mathbf{b}}_j \rangle} \cdot \tilde{\mathbf{b}}_j.$$

The map π_i denotes the orthogonal projection onto the orthogonal of the space generated by vectors $\mathbf{b}_1, \ldots, \mathbf{b}_{i-1}$ in $\mathbb{R}^n$. While the GSO of the basis of a lattice *is not* itself a basis of that lattice, it is a central object in the reduction theory of lattices, and in lattice reduction algorithms. This section is dedicated to the construction of an analogue object for bases of binary linear codes.

8.3.1 An orthogonality notion for binary vectors

In the case of Euclidean vector spaces $\mathbb{R}^n$, orthogonality can be defined via the standard inner-product, namely $\mathbf{x} \perp \mathbf{y} : \langle \mathbf{x}, \mathbf{y} \rangle = 0$ and so orthogonal projections onto the line spanned by $\mathbf{x}$ as $\pi_{\mathbf{x}}(\mathbf{y}) := \frac{\langle \mathbf{x}, \mathbf{y} \rangle}{\langle \mathbf{x}, \mathbf{x} \rangle} \mathbf{x}$. Orthogonality also provides Pythagorian additivity:

$$\|\mathbf{x} \oplus \mathbf{y}\|^2 = \|\mathbf{x}\|^2 + \|\mathbf{y}\|^2.$$

The space $\mathbb{F}_2^n$ is also endowed with an inner-product, but it does not lead to a geometrically meaningful notion of orthogonality. For instance for $\mathbf{x}, \mathbf{y} \in \mathbb{F}_2^n$, $\langle \mathbf{x}, \mathbf{y} \rangle = 0 \pmod 2$ does not seem to imply anything similar to Pythagorian additivity. However we note that, by definition of the Hamming weight we do have:

$$|\mathbf{x} \oplus \mathbf{y}| = |\mathbf{x}| + |\mathbf{y}| \iff \mathrm{S}(\mathbf{x}) \cap \mathrm{S}(\mathbf{y}) = \emptyset.$$

In fact, we even have the identity:

$$|\mathbf{x} \oplus \mathbf{y}| = |\mathbf{x}| + |\mathbf{y}| - 2|\mathbf{x} \wedge \mathbf{y}|$$

for $\mathbf{x}, \mathbf{y} \in \mathbb{F}_2^n$, which should be read as an analogue of the Euclidean identity:

$$\|\mathbf{x} - \mathbf{y}\|^2 = \|\mathbf{x}\|^2 + \|\mathbf{y}\|^2 - 2\langle \mathbf{x}, \mathbf{y} \rangle.$$

This suggests to define $\mathbf{x}, \mathbf{y} \in \mathbb{F}_2^n$ to be *orthopodal* if their supports are disjoint, that is, defining the orthogonality relation as:

$$\mathbf{x} \perp \mathbf{y} \iff \mathbf{x} \wedge \mathbf{y} = \mathbf{0}.$$

We can then associate convenient notions of projections *onto* (the support of) $\mathbf{x}$ and *orthopodally to* (the support of) $\mathbf{x}$ as follows:

$$\pi_{\mathbf{x}} : \mathbf{y} \mapsto \mathbf{y} \wedge \mathbf{x}, \quad \pi_{\mathbf{x}}^{\perp} : \mathbf{y} \mapsto \mathbf{y} \wedge \overline{\mathbf{x}} = \pi_{\overline{\mathbf{x}}}(\mathbf{y})$$

Such transformations are certainly not new to codes, and known in the literature as *puncturing* [MS77, Ch.1 §9]. However, we are here especially interested in their geometric virtues. They do satisfy similar properties as their Euclidean analogues: $\pi_{\mathbf{x}}$ is linear, idempotent, it fixes $\mathbf{x}$, it does not increase length (Hamming weight), and together with $\pi_{\mathbf{x}}^{\perp}$ yields an orthogonal decomposition. More formally:

Lemma 118. *For any $\mathbf{x}, \mathbf{y}, \mathbf{z} \in \mathbb{F}_2^n$ it holds that:*

$$\begin{aligned} \pi_{\mathbf{x}}(\mathbf{y} \oplus \mathbf{z}) = \pi_{\mathbf{x}}(\mathbf{y}) \oplus \pi_{\mathbf{x}}(\mathbf{z}), \quad & \pi_{\mathbf{x}}^2(\mathbf{y}) = \pi_{\mathbf{x}}(\mathbf{y}), \\ \pi_{\mathbf{x}}^{\perp}(\mathbf{x}) = \mathbf{0}, \quad & \pi_{\mathbf{x}}(\mathbf{x}) = \mathbf{x}, \\ \pi_{\mathbf{x}}^{\perp}(\mathbf{y}) \perp \mathbf{x}, \quad & |\pi_{\mathbf{x}}(\mathbf{y})| \leq |\mathbf{y}|, \\ \pi_{\mathbf{x}}(\mathbf{y}) \perp \pi_{\mathbf{x}}^{\perp}(\mathbf{y}), \quad & \pi_{\mathbf{x}}(\mathbf{y}) \oplus \pi_{\mathbf{x}}^{\perp}(\mathbf{y}) = \mathbf{y}. \end{aligned}$$

The proof is immediate by boolean algebra. Furthermore, and unlike their Euclidean analogues: they always commute and their compositions can be compactly represented.

Lemma 119. *For any $\mathbf{x}, \mathbf{y} \in \mathbb{F}_2^n$ it holds that:*

$$\pi_{\mathbf{x}} \circ \pi_{\mathbf{y}} = \pi_{\mathbf{y}} \circ \pi_{\mathbf{x}} = \pi_{\mathbf{x} \wedge \mathbf{y}},$$

$$\pi_{\mathbf{x}}^{\perp} \circ \pi_{\mathbf{y}}^{\perp} = \pi_{\mathbf{y}}^{\perp} \circ \pi_{\mathbf{x}}^{\perp} = \pi_{\mathbf{x} \vee \mathbf{y}}^{\perp}.$$

The proof is also immediate by boolean algebra. We therefore extend the notation $\pi_S^{\perp}$ to sets $S \subseteq \mathbb{F}_2^n$ to denote the projection orthopodally to the support of S, namely $\pi_{\mathbf{x}}^{\perp}$ where $\mathbf{x} = \bigvee_{\mathbf{s} \in S} \mathbf{s}$. This compact representation will allow for various algorithmic speed-ups.

8.3.2 Epipodal matrix

We are now fully equipped to define an analogue of the Gram-Schmidt Orthogonalisation process over real matrices to the case of binary matrices. An important remark is that this analogue notion given below does not preserve the $\mathbb{F}_2$-span of partial bases, which may appear as breaking the analogy with the GSO over the reals which precisely preserves $\mathbb{R}$-span of those partial bases. This is in fact not the right analogy for our purpose, noting that the GSO does not preserve the $\mathbb{Z}$-spans of those partial bases. The proper lattice to code translation (Table 8.1) associates $\mathbb{Z}$-spans (*i.e.* lattices) to $\mathbb{F}_2$-spans (*i.e.* codes), and $\mathbb{R}$-spans to *supports*.

Definition 120 (Epipodal matrix). *Let $\mathbf{B} = [\mathbf{b}_1, \ldots, \mathbf{b}_k] \in \mathbb{F}_2^{n \times k}$ be a binary matrix. The i-th projection associated to this matrix is defined as $\pi_i := \pi^{\perp}_{\{\mathbf{b}_1, \ldots, \mathbf{b}_{i-1}\}}$ where π_1 denotes the identity. Equivalently,*

$$\begin{array}{rccc} \pi_i : & \mathbb{F}_2^n & \longrightarrow & \mathbb{F}_2^n \\ & \mathbf{c} & \longmapsto & \mathbf{c} \wedge \overline{(\mathbf{b}_1 \vee \cdots \vee \mathbf{b}_{i-1})}. \end{array}$$

The i-th epipodal vector is then defined as:

$$\mathbf{b}_i^+ := \pi_i(\mathbf{b}_i),$$

and the matrix $\mathbf{B}^+ := [\mathbf{b}_1^+, \ldots, \mathbf{b}_k^+] \in \mathbb{F}_2^{n \times k}$ is called the epipodal matrix of $\mathbf{B}$.

Note that the definition does not need to be restricted to full rank matrices. The i-th epipodal vector should be interpreted as the support increment from the code $\mathcal{C}(\mathbf{b}_1, \ldots, \mathbf{b}_{i-1})$ to $\mathcal{C}(\mathbf{b}_1, \ldots, \mathbf{b}_i)$. The epipodal matrix enjoys the following properties, analogue to the GSO.

Lemma 121 (Properties of Epipodal Matrices). *For any binary matrix $\mathbf{B} = [\mathbf{b}_1, \ldots, \mathbf{b}_k] \in \mathbb{F}_2^{n \times k}$, its epipodal matrix $\mathbf{B}^+ = [\mathbf{b}_1^+, \ldots, \mathbf{b}_k^+]$ satisfies:*

1. *The epipodal vectors are pairwise orthopodal:*

$$\forall i \neq j, \quad \mathbf{b}_i^+ \perp \mathbf{b}_j^+. \tag{8.1}$$

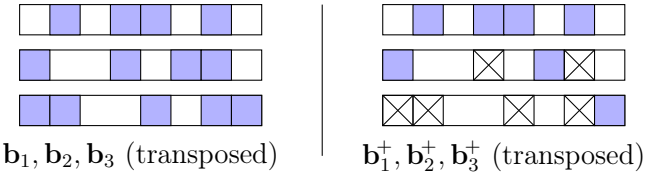

Figure 8.1: The (transposed) basis $\mathbf{B}$ of a $[8, 3]$-code, and its associated epipodal matrix $\mathbf{B}^+$.

2. *For all $i \leq k$, $(\mathbf{b}_1, \ldots, \mathbf{b}_i)$ and $[\mathbf{b}_1^+, \ldots, \mathbf{b}_i^+]$ have the same supports, that is:*

$$\bigcup_{j \leq i} \mathrm{S}(\mathbf{b}_j^+) = \bigcup_{j \leq i} \mathrm{S}(\mathbf{b}_j), \text{ or equivalently, } \bigvee_{j \leq i} \mathbf{b}_j^+ = \bigvee_{j \leq i} \mathbf{b}_j. \quad (8.2)$$

Proof. For any i, we have by definition that $\mathbf{b}_i^+ = \pi_i(\mathbf{b}_i) = \mathbf{b}_j \wedge \overline{(\mathbf{b}_1 \vee \cdots \vee \mathbf{b}_{i-1})}$, so it holds that $\mathrm{S}(\mathbf{b}_i^+) \subset \mathrm{S}(\mathbf{b}_i)$, and that $\mathrm{S}(\mathbf{b}_i^+) \cap \mathrm{S}(\mathbf{b}_j) = \emptyset$ for any $j < i$. Therefore $\mathrm{S}(\mathbf{b}_j^+) \cap \mathrm{S}(\mathbf{b}_i^+) = \emptyset$, that is $\mathbf{b}_i^+ \perp \mathbf{b}_j^+$. For the second item, rewrite $\mathbf{b}_j^+ = \mathbf{b}_j \wedge \bigwedge_{i<j} \overline{\mathbf{b}_i}$ and conclude by induction.

□

Furthermore, one may note that epipodal vectors satisfy a similar induction to the one of the GSO over the reals:

$$\text{GSO: } \tilde{\mathbf{b}}_i = \mathbf{b}_i - \sum_{j<i} \frac{\langle \mathbf{b}_i, \tilde{\mathbf{b}}_j \rangle}{\langle \tilde{\mathbf{b}}_j, \tilde{\mathbf{b}}_j \rangle} \cdot \tilde{\mathbf{b}}_j.$$

$$\text{Epipodal Matrix: } \mathbf{b}_i^+ = \mathbf{b}_i \oplus \sum_{j<i} \mathbf{b}_i \wedge \mathbf{b}_j^+,$$

However following this induction leads to perform $O(k^2)$ vector operations. In the case of the epipodal matrix, the computation can be sped-up to $O(k)$ vector operations using cumulative support vectors $\mathbf{s}_i$:

$$\mathbf{s}_0 = \mathbf{0}, \quad \mathbf{s}_i = \mathbf{s}_{i-1} \vee \mathbf{b}_i, \quad \mathbf{b}_i^+ = \mathbf{b}_i \wedge \overline{\mathbf{s}_{i-1}}.$$

The epipodal matrix of the basis of a code also enjoys an analogue invariant to the GSO for lattices. The GSO $(\tilde{\mathbf{b}}_1, \ldots, \tilde{\mathbf{b}}_k)$ of a lattice $\mathcal{L}$ is not generally a basis of $\mathcal{L}$ but it verifies the following invariant $\prod_{i=1}^{k} \|\tilde{\mathbf{b}}_i\| = \det(\mathcal{L})$.

Corollary 122 (Length Invariant). *For any basis* $[\mathbf{b}_1, \dots, \mathbf{b}_k]$ *of an* $[n, k]$*-code* $\mathcal{C}$*:*

$$\sum_{i=1}^{k} |\mathbf{b}_i^+| = |\mathcal{C}|.$$

Proof. Using equation (8.2) we have $\bigcup_{j \le n} \mathrm{S}(\mathbf{b}_j^+) = \bigcup_{j \le n} \mathrm{S}(\mathbf{b}_j) = \mathrm{S}(\mathcal{C})$, and according to (8.1) this union is disjoint. □

8.4 Size-reduction and its fundamental domain

While the GSO of a lattice basis is not itself a basis of that lattice, it is a central notion to define what a "good" basis is. For example, because of the invariant $\prod \|\tilde{\mathbf{b}}_i\| = \det(\mathcal{L})$, and because $\tilde{\mathbf{b}}_1 = \mathbf{b}_1$, making the first vector of a basis short means that other Gram-Schmidt lengths must grow.

The notion of quality of a basis should more specifically be linked to what we can do algorithmically with it. In the cases of lattices, the GSO of a basis allows to *tile* the space. More formally, if $\mathbf{B}$ is the basis of a full rank lattice $\mathcal{L} \subset \mathbb{R}^n$, one can define a fundamental domain of the translation action of $\mathcal{L}$ over $\mathbb{R}^n$ (*i.e.* a set of representatives of the quotient $\mathbb{R}^n/\mathcal{L}$) by the following rectangle parallelepiped:

$$\mathcal{P}(\tilde{\mathbf{B}}) := \left\{ \sum x_i \tilde{\mathbf{b}}_i \;\middle|\; -\tfrac{1}{2} \le x_i < \tfrac{1}{2} \right\} = \left[-\frac{1}{2}, \frac{1}{2} \right)^k \cdot \tilde{\mathbf{B}}$$

Furthermore, there is a polynomial time algorithm that effectively reduces points $\mathbf{x} \in \mathbb{R}^n$ modulo $\mathcal{L}$ to this parallelepiped, namely *size-reduction* [LLL82], also known as the *Nearest Plane Algorithm* [Bab86]. This parallelepiped has inner radius $r_{\text{in}} = \min \|\tilde{\mathbf{b}}_i\|/2$ and outer square radius $r_{\text{out}}^2 = \frac{1}{4} \sum \|\tilde{\mathbf{b}}_i\|^2$. This means that size-reduction can, in the worst case, find a close lattice vector at distance r_{out}, and correctly decode all errors of length up to r_{in}. One can also establish that the average squared distance of the decoding of a random coset is $\frac{1}{12} \sum \|\tilde{\mathbf{b}}_i\|^2$.

This Section is dedicated to an equivalent size-reduction algorithm for binary codes, and to the study of its associated fundamental domain.

8.4.1 Size-reduction: definition and algorithm

Let us start by defining size-reduction and its associated fundamental domain. A first technical detail is that in the case of codes, it is not given that the epipodal vectors are nonzero, a minor difference with the Gram-Schmidt Orthogonalisation for bases in a real vector space. We restrict our attention to *proper bases*.

Definition 123 (Proper bases). *A basis* $\mathbf{B}$ *is proper if all epipodal vectors* $\mathbf{b}_i^+$ *are nonzero.*

Note for example that bases in systematic form are proper bases: proper bases do exist for all codes, and can be produced from any basis in polynomial time.

A more annoying hiccup is that we may need to handle ties to prevent the size-reduction tiles from overlapping; in the case of lattices these might as well be ignored as the difference between $[-\frac{1}{2}, \frac{1}{2}]^n$ and $[-\frac{1}{2}, \frac{1}{2})^n$ has zero measure. This issue arises when an epipodal vector $\mathbf{p}$ has even weight, if we are reducing some $\mathbf{y} \in \mathbb{F}_2^n$ such that $|\mathbf{y} \wedge \mathbf{p}| = |\mathbf{p}|/2 = |(\mathbf{y} \oplus \mathbf{p}) \wedge \mathbf{p}|$. We (arbitrarily) use the first epipodal coordinate to break such ties:

$$\mathrm{TB}_{\mathbf{p}}(\mathbf{y}) = \begin{cases} 0 & \text{if } |\mathbf{p}| \text{ is odd,} \\ 0 & \text{if } y_j = 0 \text{ where } j = \min(\mathrm{S}(\mathbf{p})), \\ 1/2 & \text{otherwise.} \end{cases}$$

Definition 124 (size-reduction). *Let* $\mathbf{B} = [\mathbf{b}_1, \ldots, \mathbf{b}_k]$ *be a basis of an* $[n, k]$*-code. The Size-Reduced region relative to* $\mathbf{B}$ *is defined as:*

$$\mathcal{F}(\mathbf{B}^+) := \left\{ \mathbf{y} \in \mathbb{F}_2^n \ : \ \forall i \in [\![1, k]\!], \ |\mathbf{y} \wedge \mathbf{b}_i^+| + \mathrm{TB}_{\mathbf{b}_i^+}(\mathbf{y}) \leq \frac{|\mathbf{b}_i^+|}{2} \right\}.$$

Vectors in this region are said to be size-reduced with respect to the basis $\mathbf{B}$*.*

Furthermore, as for lattices, we have an efficient size-reduction algorithm that reduces any target to this region.

Proposition 125. *Algorithm 12 is correct and runs in polynomial time.*

Algorithm 12: SizeRed($\mathbf{B}, \mathbf{y}$) Size-reduce $\mathbf{y}$ with respect to $\mathbf{B}$

Input : A basis $\mathbf{B} = [\mathbf{b}_1, \ldots, \mathbf{b}_k] \in \mathbb{F}_2^{k \times n}$ and a target $\mathbf{y} \in \mathbb{F}_2^n$
Output: $\mathbf{e} \in \mathcal{F}(\mathbf{B}^+)$ such that $\mathbf{e} \oplus \mathbf{y} \in \mathcal{C}(\mathbf{B})$

$\mathbf{e} \leftarrow \mathbf{y}$
for $i = k$ *down to* 1 **do**
  if $|\mathbf{e} \wedge \mathbf{b}_i^+| + \mathrm{TB}_{\mathbf{b}_i^+}(\mathbf{e}) > |\mathbf{b}_i^+|/2$ **then**
    $\mathbf{e} \leftarrow \mathbf{e} \oplus \mathbf{b}_i$
  end
end
return $\mathbf{e}$

Proof. First note that $\mathbf{e} \oplus \mathbf{y} \in \mathcal{C}(\mathbf{B})$ is a loop invariant, as we only add basis vectors $\mathbf{b}_i$ to $\mathbf{e}$. Therefore all we need to show is that $\mathbf{e} \in \mathcal{F}(\mathbf{B}^+)$. Note that the loop at step i enforces $|\mathbf{e} \wedge \mathbf{b}_i^+| + \mathrm{TB}_{\mathbf{b}_i^+}(\mathbf{e}) \leq |\mathbf{b}_i^+|/2$. Furthermore, this constraint is maintained by subsequent loop iterations of index $j < i$ since $\mathbf{b}_j \wedge \mathbf{b}_i^+ = \mathbf{0}$. □

Proposition 126 (Fundamental Domain). *Let $\mathbf{B} := [\mathbf{b}_1, \ldots, \mathbf{b}_k]$ be a proper basis of an $[n, k]$-code $\mathcal{C}$. Then $\mathcal{F}(\mathbf{B}^+)$ is a fundamental domain for $\mathcal{C}$, that is:*

1. *$\mathcal{F}(\mathbf{B}^+)$ is $\mathcal{C}$-packing:*

$$\forall \mathbf{c} \in \mathcal{C} \setminus \{\mathbf{0}\}, \quad (\mathbf{c} + \mathcal{F}(\mathbf{B}^+)) \cap \mathcal{F}(\mathbf{B}^+) = \emptyset,$$

2. *$\mathcal{F}(\mathbf{B}^+)$ is $\mathcal{C}$-covering:*

$$\mathcal{C}(\mathbf{B}) + \mathcal{F}(\mathbf{B}^+) = \mathbb{F}_2^n.$$

Proof. Let us start by proving that $\mathcal{F}(\mathbf{B}^+)$ is $\mathcal{C}(\mathbf{B})$-packing. Let $\mathbf{c} = \sum_i x_i \mathbf{b}_i \in \mathcal{C}(\mathbf{B})$ and $\mathbf{y}_1, \mathbf{y}_2 \in \mathcal{F}(\mathbf{B}^+)$ such that $\mathbf{c} = \mathbf{y}_1 \oplus \mathbf{y}_2$. By definition of the orthopodalization $\mathbf{c} = \sum_i x_i \mathbf{b}_i^+ \oplus \sum_{j<i} x_i \mathbf{b}_i \wedge \mathbf{b}_j^+$ which gives:

$$\forall \ell \in [\![1, k]\!], \quad \mathbf{c} \wedge \mathbf{b}_\ell^+ = x_\ell \mathbf{b}_\ell^+ \oplus \sum_{i>\ell} x_i \mathbf{b}_i \wedge \mathbf{b}_\ell^+.$$

Suppose by contradiction that $\mathbf{c} \neq \mathbf{0}$. Let j be the largest index such that $x_j \neq 0$. Then $\mathbf{c} \wedge \mathbf{b}_j^+ = \mathbf{b}_j^+$. As $\mathbf{c} = \mathbf{y}_1 \oplus \mathbf{y}_2$ where $\mathbf{y}_1, \mathbf{y}_2 \in \mathcal{F}(\mathbf{B}^+)$,

$$|\mathbf{c} \wedge \mathbf{b}_j^+| \leq |\mathbf{y}_1 \wedge \mathbf{b}_j^+| + |\mathbf{y}_2 \wedge \mathbf{b}_j^+| < |\mathbf{b}_j^+|,$$

which is a contradiction. Therefore $\mathbf{c} = \mathbf{0}$ which shows that $\mathcal{F}(\mathbf{B}^+)$ is $\mathcal{C}(\mathbf{B})$-packing. The $\mathcal{C}(\mathbf{B})$-covering property follows from the fact that Algorithm 12 is correct as proven in Proposition 125. □

8.4.2 Decoding performance of size-reduction

Given any fundamental domain $\mathcal{F}$ of an $[n, k]$-code $\mathcal{C}(\mathbf{B})$ and a corresponding reduction algorithm one can consider the decoding performance. Such a reduction algorithm would reduce a target $\mathbf{y} \in \mathbb{F}_2^n$ to the (unique) error $\mathbf{e} \in (\mathbf{y} + \mathcal{C}) \cap \mathcal{F}$, and thereby finding a close codeword $\mathbf{c} = \mathbf{y} \oplus \mathbf{e}$ to $\mathbf{y}$. A uniformly random target in $\mathbb{F}_2^n$ is uniformly distributed over the fundamental domain after reduction and thus the quality of decoding fully dependents on the geometric properties of the fundamental domain. For example, the expected Hamming distance is equal to the expected weight $\mathbb{E}[|\mathcal{U}(\mathcal{F})|]$. And, more precisely, decoding a random target, to a codeword at Hamming distance w, succeeds with probability:

$$p_{\leq w}^{\text{rand}}(\mathcal{F}) = \frac{\#(\mathcal{F} \cap \mathcal{B}_w^n)}{\#\mathcal{F}} = \frac{\#(\mathcal{F} \cap \mathcal{B}_w^n)}{2^{n-k}}.$$

The minimal distance $d := d_{\min}(\mathcal{C})$ of a random code $\mathcal{C}$ is expected to be very close to the Gilbert-Varshamov bound [OS09, §3.2, Definition 1], and a uniformly random target lies almost always at distance $\approx d$ (see [MO15, §2] for a justification). Therefore in this setting random decoding is mostly interesting for $w \geq d$, as otherwise no solution is expected to exist.

Another regime we can consider is unique decoding, where we have the guarantee that our target has a unique codeword at distance at most w. For general codes this is the case for half distance decoding up to weight $w < d/2$. If the error is uniform over $\mathcal{B}_w^n$, we obtain a success probability of

$$p_{\leq w}^{\text{uniq}}(\mathcal{F}) = \frac{\#\mathcal{F} \cap \mathcal{B}_w^n}{\#\mathcal{B}_w^n}.$$

For random codes a target at distance at most $w < d$ (instead of $d/2$) is almost always uniquely decodable and as a result the success probability is also close to the above quantity. For cryptanalytic purposes it is assumed to be identical [OS09, page 3.3].

Let us now consider three such fundamental domains, an optimal one, the one corresponding to an algorithm by Prange [Pra62], and finally the size-reduction domain $\mathcal{F}(\mathbf{B}^+)$.

Maximum likelihood decoding

An optimal decoder, also know as a maximum likelihood decoder, always decodes to a nearest codeword. Implicitly this corresponds to a fundamental domain $\mathcal{F}$ where each coset representative has minimal weight. This implies that the random decoding probability $p^{\text{rand}}_{\leq w}(\mathcal{F})$ hits the probability that a uniformly random target lies at distance at most w to the code, and a perfect unique decoding probability of $p^{\text{uniq}}_{\leq w}(\mathcal{F}) = 1$. For lattices the analogue fundamental domain is unique up to the boundary and is known as the *Voronoi Domain* of a lattice. Unfortunately, reducing a target to this fundamental domain is in general hard for both lattices and codes, and takes exponential time [Pra62; MV13; DLW19].

Prange's fundamental domains

Given a basis $\mathbf{B}$ in systematic form, a more common decoding algorithm, namely the Prange algorithm [Pra62], is to assume an error of $\mathbf{0}$ on the k pivot positions of the systematic form. This induces a fundamental domain, but of geometric shape $\mathbb{F}_2^{n-k} \times \{0\}^k$ instead of $\mathcal{F}(\mathbf{B}^+)$. This leads respectively to an expected error weight of $(n-k)/2$, and random and unique decoding probabilities of:

$$p^{\text{rand}}_{\leq w}(\mathbb{F}_2^{n-k} \times \{0\}^k) = \frac{\#\mathcal{B}_w^{n-k}}{2^{n-k}}, \quad \text{and} \quad p^{\text{uniq}}_{\leq w}(\mathbb{F}_2^{n-k} \times \{0\}^k) = \frac{\#\mathcal{B}_w^{n-k}}{\#\mathcal{B}_w^n}.$$

Size-reduction

In the case of lattices, size-reduction gives a fundamental domain that can be written as a direct sum of segments $\mathcal{P}(\tilde{\mathbf{B}}) = \prod_i [-1/2, 1/2) \cdot \tilde{\mathbf{b}}_i$ where the $\tilde{\mathbf{b}}_i$'s are the GSO of the lattice basis. The expected squared

decoding error $\frac{1}{12}\sum\|\tilde{\mathbf{b}}_i\|^2$ is simply the sum of the expected squared error on each segment, and only depends on the Gram-Schmidt profile $\|\tilde{\mathbf{b}}_1\|,\ldots,\|\tilde{\mathbf{b}}_k\|$.

We proceed similarly for the Size-Reduced region $\mathcal{F}(\mathbf{B}^+)$. The role of the segment is taken over by the *fundamental ball* $\mathcal{E}^p$ of length $p>0$ as:

$$\mathcal{E}^p := \left\{\mathbf{y}\in\mathbb{F}_2^p \ :\ |\mathbf{y}| + \mathrm{TB}_{(1,\ldots,1)}(\mathbf{y}) \le p/2\right\}.$$

It should be thought of as the canonical fundamental domain of the $[p,1]$-code $\mathcal{C}=\mathbb{F}_2\cdot(1,1,\ldots,,1)$ inside $\mathbb{F}_2^p$ (the repetition code of length p). To each proper epipodal vector $\mathbf{b}_i^+$ we assign an *epipodal ball* $\mathcal{E}^{|\mathbf{b}_i^+|}$, and this allows to rewrite the fundamental domain $\mathcal{F}(\mathbf{B}^+)$ as a direct product of balls via the isometry:

$$\mathcal{F}(\mathbf{B}^+) \xrightarrow{\sim} \prod_{i=1}^{k} \mathcal{E}^{|\mathbf{b}_i^+|} : \quad \mathbf{y}\mapsto \left(\mathbf{y}|_{\mathrm{S}(\mathbf{b}_1^+)},\ldots,\mathbf{y}|_{\mathrm{S}(\mathbf{b}_k^+)}\right).$$

The latter object only depends on the epipodal lengths $|\mathbf{b}_1^+|,\ldots,|\mathbf{b}_k^+|$ which we call the *profile* $(\ell_i := |\mathbf{b}_i^+|)_i$ of the basis $\mathbf{B}$.

We can now proceed to analyse the Hamming weight of uniformly random targets in the domain by looking at their weight in each local fundamental ball $\mathcal{E}^{|\mathbf{b}_i^+|}$. Let $W_p := |\mathcal{U}(\mathcal{E}^p)|$ for $p>0$ be the distribution of the Hamming weight of uniformly drawn targets in the fundamental ball $\mathcal{E}^p$. This weight distribution has the following probabilities for integer weight $w\ge 0$:

$$\Pr\left[W_p=w\right] = \begin{cases} 0 & \text{if } w>p/2 \text{ or } w<0,\\ \frac{\binom{p}{p/2}}{2^p} & \text{if } w=p/2,\\ \frac{\binom{p}{w}}{2^{p-1}} & \text{otherwise,}\end{cases}$$

and its expectation is given by:

$$\mathbb{E}\left[W_p\right] = \frac{p}{2} - \left\lceil\frac{p}{2}\right\rceil\cdot\binom{p}{\lfloor\frac{p}{2}\rfloor}\cdot 2^{-p} = \frac{p}{2} - \sqrt{\frac{p}{2\pi}} + \Theta(1/\sqrt{p}) \qquad (8.3)$$

which is bounded by $\frac{p-1}{2}$. This bound is strict for $p\ge 3$, which will give us a gain over the Prange decoder [Pra62]. Analogues to the lattice case the expected Hamming weight of a uniformly random target in $\mathcal{E}^p$ is given by the sum of the local expectations $\sum_{i=1}^{k}\mathbb{E}\left[W_{\ell_i}\right]$.

To be more precise, for a basis $\mathbf{B}$ with profile $\ell = (\ell_1, \dots, \ell_k)$, the weight distribution $W(\mathbf{B}) := |\mathcal{U}(\mathcal{F}(\mathbf{B}^+))|$ of the size-reduction algorithm, simply denoted by $W(\ell)$, is given by a convolution of the local weight distributions $W_{\ell_1}, \dots, W_{\ell_k}$:

$$\Pr[W(\mathbf{B}) = w] = \sum_{\sum_i w_i = w} \left(\prod_{i=1}^{k} \Pr[W_{\ell_i} = w_i] \right).$$

The whole distribution can efficiently (in time polynomial in n) be computed by iterated convolutions, as its support $[\![0, n]\!]$ is discrete and small (see `weights.py`).

8.4.3 Comparing profiles for size-reduction decoding

We have shown that the geometric shape of the fundamental domain $\mathcal{F}(\mathbf{B}^+)$ depends fully on the profile $\ell_1, \dots, \ell_k$. For any profile we have $\mathbb{E}[W(\ell)] = \sum_{i=1}^{k} \mathbb{E}[W_{\ell_i}] \leq \sum_{i=1}^{k} \frac{\ell_i - 1}{2} = (n-k)/2$, with a strict inequality if $|\ell_i| \geq 3$ for any $i \in [\![1, k]\!]$, and thus we improve on the fundamental domain induced by Prange. But what is actually a good profile?

Let us first focus on the expected error weight. By eq. (8.3) the expectation roughly equals

$$\mathbb{E}[W(\ell)] \approx \sum_{i=1}^{k} \frac{\ell_i}{2} - \sqrt{\frac{\ell_i}{2\pi}} = n/2 - \frac{1}{\sqrt{2\pi}} \sum_{i=1}^{k} \sqrt{\ell_i},$$

or more precisely we have the following statement, which follows from the inequalities $\frac{4^k}{\sqrt{\pi(k+1)}} \leq \binom{2k}{k} \leq \frac{4^k}{\sqrt{\pi k}}$.

Lemma 127. *Given a proper basis $\mathbf{B}$ of an $[n, k]$-code with profile $\ell = (\ell_1, \dots, \ell_k)$ we have:*

$$\frac{1}{\sqrt{\pi}} \sum_{i=1}^{k} \sqrt{\frac{\left\lceil \frac{\ell_i}{2} \right\rceil^2}{\left\lceil \frac{\ell_i}{2} \right\rceil + 1}} \leq \frac{n}{2} - \mathbb{E}[W(\mathbf{B})] \leq \frac{1}{\sqrt{\pi}} \sum_{i=1}^{k} \sqrt{\left\lceil \frac{\ell_i}{2} \right\rceil}.$$

Since $x \mapsto \sqrt{x}$ is concave, Lemma 127 (ignoring the rounding) suggest that the expected error is minimised when the ℓ_i are the most

balanced. Again a similar phenomenon is well known in the case of lattices: on random inputs, size-reduction produces vectors with an expected squared length of $\frac{1}{12}\sum \|\tilde{\mathbf{b}}_i\|^2$; under the invariant $\prod \|\tilde{\mathbf{b}}_i\| = \det(\mathcal{L})$ the expectation is minimised for a basis with a balanced profile $\|\tilde{\mathbf{b}}_1\| = \|\tilde{\mathbf{b}}_2\| = \cdots = \|\tilde{\mathbf{b}}_k\|$.

However, the quantity $\mathbb{E}[W(\mathbf{B})]$ discussed above does not necessarily reflect the quality of the basis for all relevant algorithmic tasks. For example, if one wishes to decode errors of weight at most w with a 100% success probability, it is necessary and sufficient that $w < \min_i \ell_i/2$.

We therefore propose the following partial ordering on profiles that is meant to account that a profile is better than another for the mentioned natural decoding tasks; as a counterpart, this is only a partial ordering and two profiles may simply be incomparable.

Definition 128 (Comparing Profiles). *We define a partial ordering $(\mathcal{L}_{n,k}, \preceq)$ on the set of proper profiles $\mathcal{L}_{n,k}$ of $[n,k]$-codes by:*

$$\ell \preceq \ell' \iff \Pr[W(\ell) \leq w] \geq \Pr[W(\ell') \leq w] \text{ for all } w \in [\![0, n]\!].$$

This also defines an equivalence relation $\asymp$ on $\mathcal{L}_{n,k}$. We call a profile ℓ better than ℓ' if $\ell \preceq \ell'$. We call ℓ strictly better than ℓ' and write $\ell \prec \ell'$ if $\ell \preceq \ell'$ and $\ell \not\asymp \ell'$. We call ℓ, ℓ' incomparable and write $\ell \nparallel \ell'$ if $\ell \not\preceq \ell'$ and $\ell \not\succeq \ell'$.

Let us first justify the relevance of this partial ordering for random decoding and unique decoding for an $[n,k]$-code $\mathcal{C}(\mathbf{B})$ with profile ℓ.

Random decoding

The probability to successfully decode a random target up to an error of weight at most w can directly be expressed as $\Pr[W(\ell) \leq w]$. For a better profile we see that this probability will also be higher. The expected distance is equal to $\mathbb{E}[W(\ell)]$. By noting that $\mathbb{E}[W(\ell)] = n - \sum_{w=0}^{n} \Pr[W(\ell) \leq w]$ we see that a better profile also implies a lower expected distance.

Unique decoding

For unique decoding up to weight $w < d_{\min}(\mathcal{C}(\mathbf{B}))/2$ we can also rewrite the success probability in terms of the weight distribution as:

$$\frac{\#\mathcal{F}(\mathbf{B}^+) \cap \mathcal{B}_w^n}{\#\mathcal{B}_w^n} = \frac{2^{n-k} \cdot \Pr[W(\ell) \leq w]}{\sum_{i=0}^{w} \binom{n}{i}}.$$

Again we see that a better profile gives a higher success probability.

The more balanced, the better

Now that we have argued that the ordering of Definition 128 is relevant, let us show that, indeed, balanced profiles are preferable. As we will see, this rule of thumb is in fact imperfect, and only apply strictly to profiles that share the same parities $(\wp_i := \ell_i \bmod 2)_i$.

Lemma 129 (Profile Relations). *The partial ordering $\preceq$ has the following properties:*

(1) If ℓ is a permutation of ℓ' then $\ell \asymp \ell'$.

(2) if $\ell_1 \preceq \ell_1'$ and $\ell_2 \preceq \ell_2'$, then $(\ell_1|\ell_2) \preceq (\ell_1'|\ell_2')$.

(3) If $3 \leq x \leq y+1$, then $(x, y) \prec (x-2, y+2)$.

Proof. (1) follows from the fact that the geometric properties of the size-reduced region fully depend on the values of the profile (and not their ordering). For (2) note that $\ell_i \preceq \ell_i'$ implies the existence of a one-to-one map f_i from the size-reduction domain $\mathcal{F}(\ell_i)$ to $\mathcal{F}(\ell_i')$ that is non-decreasing in weight for $i = 1, 2$. The product map $f_1 \times f_2$ is then a one-to-one map from $\mathcal{F}(\ell_1|\ell_2)$ to $\mathcal{F}(\ell_1'|\ell_2')$ that is non-decreasing in weight, which implies that $(\ell_1|\ell_2) \preceq (\ell_1'|\ell_2')$. For the technical proof of (3) see the full version of this work[3].

□

[3] Available at https://eprint.iacr.org/2020/869.

An odd game of parity

Although for fixed parity a balanced profile is always better than an unbalanced one there are some exceptions to this rule when dropping the parity constraint. For example, looking at Lemma 127 one can notice that, at least for average decoding distances, odd values in a profile are preferable to even values. One can show that a slight unbalance with odd coefficients is preferable for all purpose to a perfect even balance:

$$(x-1, x+1) \preceq (x, x) \quad \text{for all even } x \geq 2.$$

This is an artefact of the need to tie-break certain size-reductions with respect to epipodal vectors of even length.

8.4.4 Basis size-reduction

To complete the analogy with the lattice literature, let us now adapt the notion of size-reduction for a basis. We call a basis size-reduced if each basis vector is size-reduced with respect to all previous basis vectors.

Definition 130. *We call a proper basis* $\mathbf{B} = [\mathbf{b}_1, \dots, \mathbf{b}_k]$ *size-reduced if* $\mathbf{b}_i \in \mathcal{F}([\mathbf{b}_1, \dots, \mathbf{b}_{i-1}]^+)$ *for all* $1 < i \leq k$.

Size-reduction for a basis allows to control the basis vector lengths with the epipodal lengths as follows.

Proposition 131. *Let* $\mathbf{B} = [\mathbf{b}_1, \dots, \mathbf{b}_k]$ *be a size-reduced basis with epipodal lengths* $\ell_i = |\mathbf{b}_i^+|$. *Then, for all* $i \leq n$ *it holds that* $|\mathbf{b}_i| \leq \ell_i + \sum_{j<i} \lfloor \ell_j/2 \rfloor$ *for all* $i \leq k$.

Perhaps surprisingly, this global notion of size-reduction will not be required in the LLL algorithm for codes discussed in the next Section 8.5.2, which is a first deviation from the original LLL algorithm for lattices [LLL82]. However it can still be useful to adapt more powerful reduction algorithms such as deepLLL [SE94; FSW14].

Proposition 132. *Algorithm 13 is correct and runs in polynomial time.*

Algorithm 13: SizeRedBasis($\mathbf{B}, \mathbf{y}$) Size-reduce the basis $\mathbf{B}$

Input : A proper basis $\mathbf{B} = [\mathbf{b}_1, \ldots, \mathbf{b}_k] \in \mathbb{F}_2^{k \times n}$ of a code $\mathcal{C}$
Output: A size-reduced basis of $\mathcal{C}(\mathbf{B})$ with the same epipodal matrix as $\mathbf{B}$.

```
for i = 2 to k do
    b_i ← SizeRed([b_1, ..., b_{i-1}], b_i)
end
return [b_1, ..., b_k]
```

Proof. The polynomial claim immediately follows from Proposition 125. Secondly note that vectors $\mathbf{b}_i$'s form a basis of the code $\mathcal{C}$ given as input, and this is a loop invariant. Indeed, in any step i of the algorithm only codewords from the sub-code $\mathcal{C}(\mathbf{b}_1, \ldots, \mathbf{b}_{i-1})$ are added to $\mathbf{b}_i$. Furthermore, this does not affect $\mathbf{b}_i^+ := \pi_i(\mathbf{b}_i)$. The loop at step i enforces that $\mathbf{b}_i \in \mathcal{F}([\mathbf{b}_1, \ldots, \mathbf{b}_{i-1}]^+)$ and this constraint is maintained as $\mathbf{b}_1, \ldots, \mathbf{b}_i$ are unchanged by all later steps. □

8.5 LLL for binary codes

In the previous section, we have seen that the geometric quality of the fundamental domain $\mathcal{F}(\mathbf{B}^+)$ solely depends upon the epipodal lengths $\ell_i := |\mathbf{b}_i^+|$: the more balanced, the better, both for finding close codewords of random words, and for decoding random errors. This situation is in perfect analogy with the situation in lattices. We therefore turn to the celebrated LLL [LLL82] algorithm for lattice reduction, which aims precisely at balancing the profile $(\ell_i)_i$.

The LLL algorithm can be interpreted as an algorithmic version of the so-called Hermite's bound on the minimal length of an n-dimensional vector [GHKN06]. Again, the analogy between codes an lattices stands: the LLL reduction for codes turns out to be an algorithmic version of Griesmer's bound [Gri60].

Certainly, Griesmer's bound [Gri60] is far from tight in all regimes for the parameters of the code, as it is already the case with Hermite's bound for lattices which is exponentially weaker than Minkowski's bound. Griesmer's bound and Hermite's bound virtues reside in the algorithm underlying their proofs.

8.5.1 Griesmer's bound and LLL reduction

In this section, we revisit the classical Griesmer's bound and its proof from the perspective of reduction theory, that is we will re-interpret its proof in terms of the epipodal matrix and in particular its profile. The proof we propose is admittedly a bit less direct than the original; our purpose is to dissect this classic proof, and extract an analogue to LLL reduction for codes.

Theorem 133 (Griesmer Bound [Gri60]). *For any $[n,k]$-code of minimal distance $d := d_{\min}(\mathcal{C})$, it holds that:*

$$n \geq \sum_{i=0}^{k-1} \left\lceil \frac{d}{2^i} \right\rceil .$$

In particular, if $k-1 \geq \log_2(d)$, it holds that $d - \frac{\lceil \log_2(d) \rceil}{2} \leq \frac{n-k}{2} + 1$.

The latter inequality follows from the first by setting $r = \lceil \log_2(d) \rceil$ as follows:

$$\begin{aligned} n &\geq d \sum_{i=0}^{r-1} \frac{1}{2^i} + \sum_{i=r}^{k-1} 1 \\ &= 2d(1 - \frac{1}{2^r}) + (k-r) \\ &\geq 2d - 2 + k - r. \end{aligned}$$

Definition 134 (Griesmer-reduced basis). *A basis $\mathbf{B} = [\mathbf{b}_1, \ldots, \mathbf{b}_k]$ of an $[n,k]$-code is said to be* Griesmer*-reduced if $\mathbf{b}_i^+$ is a shortest nonzero codeword of the projected subcode $\pi_i(\mathcal{C}(\mathbf{b}_i, \ldots, \mathbf{b}_k))$ for all $i \in [\![1,k]\!]$.*

This definition is a direct analogue of the so-called Hermite-Korkine-Zolotarev (HKZ) reduction for lattice bases. Note that the existence of such a basis is rather trivial by construction: choose $\mathbf{b}_1$ as a shortest nonzero vector and so forth. The only minor difficulty is showing that the projected codes $\pi_i(\mathcal{C}(\mathbf{b}_i, \ldots, \mathbf{b}_k))$ are non-trivial, which can be done by resorting to Singleton's bound. In particular, Griesmer-reduced bases are proper bases.

Lemma 135 ([HP10, Corollary 2.7.2]). *Let $\mathcal{C}$ be an $[n,k]$-code and $\mathbf{c}$ be a codeword of weight $d_{\min}(\mathcal{C})$. Then $\mathcal{C}' := \pi_{\mathbf{c}}^{\perp}(\mathcal{C}) = \mathcal{C} \wedge \overline{\mathbf{c}}$ satisfies:*

1. *$|\mathcal{C}'| = n - d_{\min}(\mathcal{C})$ and its dimension is $k-1$,*
2. *$d_{\min}(\mathcal{C}') \geq \lceil d_{\min}(\mathcal{C})/2 \rceil$.*

Therefore with the first point of this lemma we can prove by induction on k that there exists for any $[n,k]$-code $\mathcal{C}$ a Griesmer-reduced basis. Let $[\mathbf{b}_1, \ldots, \mathbf{b}_k]$ be such a basis and let $\ell_i := |\mathbf{b}_i^+|$. From definition of Griesmer-reduced bases and the previous lemma we deduce that $\ell_{i+1} \geq \lceil \ell_i/2 \rceil$. In other words, the profile $(\ell_i)_i$ is somewhat controlled: it *does not decrease too fast*. To prove Griesmer's bound it remains to chain those inequalities and to sum them up to obtain:

$$n \geq |\mathcal{C}| = \sum_{i=1}^{k} \ell_i \geq \sum_{i=0}^{k-1} \left\lceil \frac{\ell_1}{2^i} \right\rceil = \sum_{i=0}^{k-1} \left\lceil \frac{d_{\min}(\mathcal{C})}{2^i} \right\rceil \tag{8.4}$$

The proof of Lemma 135 proceeds by a *local* minimality argument, namely it looks at the first two vectors $\mathbf{b}_1, \mathbf{b}_2$. It shows that the support of $\mathbf{b}_1$ is at most $2/3$ of the support of $\mathcal{C}(\mathbf{b}_1, \mathbf{b}_2)$:

$$|\mathbf{b}_1| \leq \tfrac{2}{3} \cdot \big|\mathcal{C}(\mathbf{b}_1, \mathbf{b}_2)\big|.$$

The proof is rather elementary as the code $\mathcal{C}(\mathbf{b}_1, \mathbf{b}_2)$ has only 3 nonzero codewords to consider: $\mathbf{b}_1, \mathbf{b}_2$ and $\mathbf{b}_1 \oplus \mathbf{b}_2$. What we should note here is that the notion of Griesmer-reduction is stronger than what is actually used by the proof: indeed, we only need the much weaker property that $\mathbf{b}_1$ is a shortest codeword of the 2-dimensional subcode $\mathcal{C}(\mathbf{b}_1, \mathbf{b}_2)$, and so forth inductively. This relaxation gives us an analogue of the LLL reduction for linear codes.

Definition 136 (LLL reduced basis). *A basis $\mathbf{B} = [\mathbf{b}_1, \ldots, \mathbf{b}_k]$ of an $[n,k]$-code is said to be* LLL*-reduced if it is a proper basis, and if $\mathbf{b}_i^+$ is a shortest nonzero codeword of the projected subcode $\pi_i(\mathcal{C}(\mathbf{b}_i, \mathbf{b}_{i+1}))$ for all $i \in [\![1, k-1]\!]$.*

Note that a Griesmer-reduced basis is an LLL reduced basis, and the same holds for lattices: an HKZ reduced lattice basis is also LLL reduced. Indeed, if $\mathbf{b}_i^+$ is a shortest codeword of $\pi_i(\mathcal{C}(\mathbf{b}_i, \ldots, \mathbf{b}_n))$ it is also a shortest vector of the subcode $\pi_i(\mathcal{C}(\mathbf{b}_i, \mathbf{b}_{i+1}))$.

Having identified this weaker yet sufficient notion of reduction, we can finalize the proof of Griesmer's bound, in a reduction-theoretic fashion.

Lemma 137. *Let* $[\mathbf{b}_1, \dots, \mathbf{b}_k]$ *be an* LLL*-reduced basis, and let* $\ell_i = |\mathbf{b}_i^+|$ *for* $i \leq k$*. Then we have,*

$$\forall i \in [\![1, k]\!], \quad \ell_{i+1} \geq \left\lceil \frac{\ell_i}{2} \right\rceil .$$

Proof. We start by noting that LLL reduced bases are proper bases by definition, hence every projected subcode $\mathcal{C}_i := \pi_i(\mathcal{C}(\mathbf{b}_i, \mathbf{b}_{i+1})) = \mathcal{C}(\mathbf{b}_i^+, \pi_i(\mathbf{b}_{i+1}))$ has dimension 2 and support size $\ell_i + \ell_{i+1}$.

Let us denote by $\mathbf{x} = \mathbf{b}_i^+$, $\mathbf{y} = \pi_i(\mathbf{b}_{i+1})$ and $\mathbf{z} = \mathbf{y} \oplus \mathbf{x}$ the three nonzero codewords of $\mathcal{C}_i$, and remark that $|\mathbf{x}| = \ell_i$, $|\mathbf{z}| = |\mathbf{x}| + |\mathbf{y}| - 2|\mathbf{x} \wedge \mathbf{y}|$ and $|\mathbf{x} \wedge \mathbf{y}| = |\mathbf{y}| - \ell_{i+1}$. This gives $|\mathbf{x}| + |\mathbf{y}| + |\mathbf{z}| = 2(\ell_i + \ell_{i+1})$, [4] and because $\mathbf{x}$ is the shortest codeword among $\mathbf{x}, \mathbf{y}, \mathbf{z}$, we conclude with:

$$\ell_i = |\mathbf{x}| \leq \tfrac{1}{3}\big(|\mathbf{x}| + |\mathbf{y}| + |\mathbf{z}|\big) \leq \tfrac{2}{3}(\ell_i + \ell_{i+1}).$$

□

We can now reformulate Griesmer bound, while making the underlying reduction notion explicit.

Theorem 138 (Griesmer bound, revisited). *Let* $[\mathbf{b}_1, \dots, \mathbf{b}_k]$ *be a basis of a (linear, binary)* $[n, k]$*-code* $\mathcal{C}$ *that is* LLL*-reduced. Then,*

$$n \geq \sum_{i=0}^{k-1} \left\lceil \frac{\ell_1}{2^i} \right\rceil , \tag{8.5}$$

where $\ell_1 := |\mathbf{b}_1| \geq d_{\min}(\mathcal{C})$*. In particular, if* $k - 1 \geq \log_2(d_{\min}(\mathcal{C}))$*, it holds that* $\ell_1 - \frac{\lceil \log_2(\ell_1) \rceil}{2} \leq \frac{n-k}{2} + 1$*. Moreover, every binary linear code admits an* LLL*-reduced basis.*

Proof. The inequalities follow from (8.4), while the existence of an LLL reduced basis follows from the fact that Griesmer-reduced bases are LLL reduced. □

[4] Alternatively, one could have invoked the more general fact that the average weights $2^{-k} \sum |\mathbf{c}|$ over a linear code of dimension k is half of its support size $|\mathcal{C}|/2$.

Tightness

A first remark is that the local bound $\ell_{i+1} \geq \left\lceil \frac{\ell_i}{2} \right\rceil$ is tight; it is reached by the $[3, 2]$-code $\mathcal{C} = \{(000), (101), (110), (011)\}$, and more generally by $[n, 2]$-codes for any $n \geq 3$ following a similar pattern. Griesmer's bound is also reached globally and thus we know inputs that give the worst case of our LLL algorithm (which computes efficiently LLL reduced bases of a code), *i.e.* the largest ℓ_1. For instance there are the simplex codes[5] [MS77, Ch.1, §9] and the Reed-Muller codes of order one [Ree53; Mul54] which are respectively $[2^m - 1, m]$-codes and $[2^m, m + 1]$-codes.

Generalisations

The discussion above shows that one can think of Griesmer's bound as an inequality relating codes of dimension k to codes of dimension 2, in the same way that Hermite related lattices of rank n to lattices of rank 2 via Hermite's inequality on the eponymous constants $\gamma_n \leq \gamma_2^{n-1}$.

This type of reasoning can be generalised to relate other quantities. In the literature on lattices, those are known as Mordell's inequalities [GHKN06; GN08a]. These bounds also have underlying algorithms, namely block-reduction algorithm such as BKZ [Sch87] and Slide [GHKN06]. Translating those bounds and their associated algorithms from lattices to codes appears as a very interesting research direction.

Beyond algorithms based on finding shortest vectors of projected sublattices, we also note that some algorithms consider the dense sublattice problem [DM13; LN14]. According to [Bog01], the analogy should be made with the notion of *higher weight* [Wei91; TV95].

Comparison with other code-based bounds

Before presenting our LLL algorithm let us quickly compare in Table 8.2 Griesmer's bound (and thus the bound reached by the LLL algorithm) to classic bounds from coding theory: Singleton's and Hamming's. One can consult [HP10] for their proofs.

An important remark is that until now, only the Singleton bound was algorithmic while Griesmer's bound was seen as an extension of

[5] The simplex code is defined as the dual of the Hamming code.

Bound	Concrete	Asymptotic	Poly. Algo.
Singleton's	$d \leq n - k + 1$	$\delta \leq 1 - R$	YES
Hamming's	$2^k \sum_{i=0}^{\lfloor \frac{d-1}{2} \rfloor} \binom{n}{i} \leq 2^n$	$R \leq 1 - h\left(\frac{\delta}{2}\right)$	NO
Griesmer's	$d - \frac{\log_2 d}{2} \leq \frac{n-k}{2} + 1$	$\delta \leq \frac{1-R}{2}$	**Now, YES**

Table 8.2: Bounds on $d = d_{\min}(\mathcal{C})$. Asymptotic form is given for a fixed rate $R = k/n \in [0, 1]$, $\delta := d/n$ and $n \to \infty$. In Hamming's constant, $h(x) := -x \log_2(x) - (1-x) \log_2(1-x)$ denotes the so-called binary entropy of x.

it but not algorithmic. For an $[n, k]$-code, Singleton's bound states that $d_{\min}(\mathcal{C}) \leq n - k + 1$. The underlying algorithm of this bound simply consists in putting the basis in systematic form, namely to reduce in row echelon form the basis, to get a short codeword. It may be argued that for random codes this bound is far from tight. Indeed, the systematic form in fact produces codewords of average length $\frac{n-k}{2} + 1$ (this is exactly what Prange algorithm [Pra62] does). While this seems better than Griesmer's bound, the LLL algorithm gives a codeword of length at most $\frac{n-k}{2} + \log_2 n$ but in the *worst-case*.

This concludes the translation of all the notions at hands from lattices to codes. We summarize them as a dictionary in Table 8.1.

8.5.2 An LLL reduction algorithm for codes

In the above subsection, we have defined LLL reduced bases and have shown that they exist by constructing a basis with an even stronger reduction property. However, such a construction requires to solve the shortest codeword problem, a problem known to be NP-hard [Var97], and the best known algorithm have exponential running time in n or in k, at least for constant rates $R = k/n$.

In other words, we have shown existence of LLL reduced bases (*local minimality*) by a *global minimality* argument, which would translate into an algorithm with exponential running time. Instead, we can show their existence by a *descent* argument, and this proof translates to a polynomial time algorithm, the *LLL algorithm for binary codes*.

The strategy to produce LLL reduced bases is very simple, and essentially the same as in the case of lattices [LLL82]: if $\pi_i(\mathbf{b}_i)$ is not a shortest nonzero codeword of $\pi_i(\mathcal{C}(\mathbf{b}_i, \mathbf{b}_{i+1}))$ for some i, then apply a change of basis on $\mathbf{b}_i, \mathbf{b}_{i+1}$ so that it is. Such a transformation may break the same property for nearby indices $i-1$ and $i+1$, however, we will show that, overall, the algorithm still makes progress.

There are two technical complications of the original LLL [LLL82] that can be removed in the case of codes. The first is that we do not need a global size-reduction on the basis; this step of LLL does not affect the Gram-Schmidt vectors themselves, but is needed for numerical stability issues, which do not arise over the finite field $\mathbb{F}_2$. Secondly, we do not need to introduce a small approximation term $\varepsilon > 0$ to prove that the algorithm terminates in polynomial time, thanks to the discreteness of epipodal lengths.

Algorithm 14: LLL($\mathbf{B}$) — LLL reduce the basis $\mathbf{B}$

Input : A proper basis $\mathbf{B} = [\mathbf{b}_1, \dots, \mathbf{b}_k] \in \mathbb{F}_2^{k \times n}$ of a code $\mathcal{C}$
Output: An LLL reduced basis for $\mathcal{C}$

while $\exists i \in [\![0, k-1]\!]$ *s.t.* $\min\left(\left|\pi_i(\mathbf{b}_{i+1})\right|, \left|\mathbf{b}_i^+ \oplus \pi_i(\mathbf{b}_{i+1})\right|\right) < \left|\mathbf{b}_i^+\right|$ **do**
  if $\left|\pi_i(\mathbf{b}_{i+1}) \wedge \mathbf{b}_i^+\right| + \mathrm{TB}_{\mathbf{b}_i^+}\left(\pi_i(\mathbf{b}_{i+1})\right) > \left|\mathbf{b}_i^+\right|/2$ **then**
    $\mathbf{b}_{i+1} \leftarrow \mathbf{b}_{i+1} \oplus \mathbf{b}_i$ `// Local size-reduction`
  end
  $\mathbf{b}_i \leftrightarrow \mathbf{b}_{i+1}$ `// Swap`
end
return B

Theorem 139. *Algorithm 14 is correct and its running time is polynomial; more precisely on input an $[n,k]$-code it performs at most kn vector operations over $\mathbb{F}_2^n$.*

It will be a consequence of the two following lemmata. Let us start with correctness.

Lemma 140 (Correctness). *If the* LLL *algorithm terminates then it outputs an* LLL*-reduced basis for the code spanned by the basis given as input.*

Proof. Note that vectors $\mathbf{b}_i$'s form a basis of the code $\mathcal{C}$ given as input and this is a loop invariant. The exit condition ensures that the basis is indeed LLL reduced, at least if it is proper. So it suffices to show that properness is also a loop invariant.

Assume that the basis is proper ($\ell_j \geq 1$ for all j) as we enter the loop at index i. The local size-reduction step does not affect epipodal lengths. The swap only affects ℓ_i and ℓ_{i+1}, and leaves $\ell_i + \ell_{i+1}$ unchanged. The epipodal length ℓ_i decreases, but remains nonzero since $\mathbf{b}_i^+$ is a shortest-codeword of a 2-dimensional code by construction. The epipodal length ℓ_{i+1} can only increase. □

The key-ingredient to prove that LLL terminates lies in the construction of a potential: a quantity that only decreases during the algorithm, and is lower bounded.

Definition 141 (Potential). *Let* $\mathbf{B} = [\mathbf{b}_1, \ldots, \mathbf{b}_k]$ *be a basis of an* $[n, k]$*-code. The potential of* $\mathbf{B}$*, denoted* $\mathcal{D}_\mathbf{B}$*, is defined as:*

$$\mathcal{D}_\mathbf{B} := \sum_{i=1}^{k} (k - i + 1)\ell_i = \sum_{i=1}^{k} \left(\sum_{j=1}^{i} \ell_j \right) = \sum_{i=1}^{k} \mathcal{D}_{\mathbf{B},i},$$

where $\mathcal{D}_{\mathbf{B},i} := |\mathcal{C}(\mathbf{b}_1, \ldots, \mathbf{b}_i)|$ *and* $\ell_i := |\mathbf{b}_i^+|$.

Each time LLL makes a swap, the potential decreases at least by one as shown in the proof of the following lemma. Furthermore, this quantity is always positive. Therefore the number of iterations is upper-bounded by the initial value of $\mathcal{D}_\mathbf{B}$.

Lemma 142 (Termination). *The number of iterations in Algorithm 14 on input* $\mathbf{B} = [\mathbf{b}_1, \ldots, \mathbf{b}_k]$ *is upper bounded by,*

$$\min \left\{ kn, \frac{k(k+1)}{2} \max_i |\mathbf{b}_i^+| \right\}.$$

Proof. The potential $\mathcal{D}_\mathbf{B}$ only changes during the swap step. Let us show that it decreases by at least one at each swap step. Suppose there is a swap at the index i. Let $\mathbf{b}_i$ and $\mathbf{b}_{i+1}$ be the values of the basis vectors after the **if** loop and just before the swap step. Here, the **if** loop ensures:

$$|\mathbf{b}_i^+ \wedge \pi_i(\mathbf{b}_{i+1})| \leq \frac{|\mathbf{b}_i^+|}{2}. \tag{8.6}$$

Now, whether or not the **if** loop was executed we have that:

$$\min\left(\left|\pi_i(\mathbf{b}_{i+1})\right|, \left|\mathbf{b}_i^+ \oplus \pi_i(\mathbf{b}_{i+1})\right|\right) < \left|\mathbf{b}_i^+\right|. \tag{8.7}$$

Therefore, combining Equations (8.6) and (8.7) with $\left|\mathbf{b}_i^+ \oplus \pi_i(\mathbf{b}_{i+1})\right| = |\mathbf{b}_i^+| + |\pi_i(\mathbf{b}_{i+1})| - 2|\mathbf{b}_i^+ \wedge \pi_i(\mathbf{b}_{i+1})|$ leads to:

$$|\pi_i(\mathbf{b}_{i+1})| < |\mathbf{b}_i^+|. \tag{8.8}$$

Now during the **while** execution only $\mathcal{D}_{\mathbf{B},i}$ is modified. Let $\mathcal{C}_i'$ be the new partial code $\mathcal{C}(\mathbf{b}_1, \ldots, \mathbf{b}_i)$ after the swap, and $\mathcal{D}'_{\mathbf{B},i}$ be the new values of $\mathcal{D}_{\mathbf{B},i}$. We have,

$$\begin{aligned}
\mathcal{D}'_{\mathbf{B},i} - \mathcal{D}_{\mathbf{B},i} &= |\mathcal{C}_i'| - |\mathcal{C}_i| \\
&= \sum_{j=1}^{i-1} |\mathbf{b}_j^+| + |\pi_i(\mathbf{b}_{i+1})| - \sum_{j=1}^{i-1} |\mathbf{b}_j^+| - |\mathbf{b}_i^+| \\
&= |\pi_i(\mathbf{b}_{i+1})| - |\mathbf{b}_i^+| < 0
\end{aligned}$$

where for the inequality we used Equation (8.8). Potentials $\mathcal{D}_{\mathbf{B},i}$ are integers. Therefore at each iteration $\mathcal{D}_{\mathbf{B}}$ decreases by at least one. Let $\mathcal{D}_{\mathbf{B}}^{(0)}$ be the initial value of the potential $\mathcal{D}_{\mathbf{B}}$. We conclude with:

$$\mathcal{D}_{\mathbf{B}}^{(0)} = \sum_{i=1}^{k} (k-i+1)|\mathbf{b}_i^+| \leq \min\left\{kn, \frac{k(k+1)}{2} \max_i |\mathbf{b}_i^+|\right\},$$

where for the first bound we use that $\sum_i |\mathbf{b}_i^+| = |\mathcal{C}| \leq n$.

□

8.6 Perspectives

This work brings codes and lattices closer to each other by enriching the existing dictionary (Table 8.1); we hope that it can enable more transfer of techniques between those two research areas, and list some research directions.

Generalisations

In principle, the definitions, theorems and algorithms of this article should be generalizable to codes over $\mathbb{F}_q$ endowed with the Hamming

metric, with the minor inconvenience that one may no longer conflate words over $\mathbb{F}_q$ with their binary support vector. Some algorithms may see their complexity grow by a factor $\Theta(q)$, meaning that the algorithms remain polynomial-time only for $q = n^{O(1)}$. It is natural to hope that such a generalised LLL would still match Griesmer [Gri60] bound for $q > 2$. However, we expect that the analysis of the fundamental domain of Section 8.4 would become significantly harder to carry out.

Another natural generalisation to aim for would be codes constructed with a different metric, in particular codes endowed with the rank metric [Gab85; Gab95]. In this case codes are subspaces of $\mathbb{F}_{q^m}$ endowed with the rank metric; the weight of a codeword $\mathbf{x} \in \mathbb{F}_{q^m}^n$ is the rank of its matrix representation over $\mathbb{F}_q$ (which is a matrix of size $m \times n$). While the support of a codeword with the Hamming metric is the set of its nonzero coordinates, the support of $\mathbf{x} = (x_1, \cdots, x_n) \in \mathbb{F}_{q^m}^n$ is the $\mathbb{F}_q$-subspace of $\mathbb{F}_{q^m}$ that the x_i's generate, namely $\{\sum_i \lambda_i x_i : \lambda_i \in \mathbb{F}_q\}$. We believe that this work can be generalised in this case, in particular the notion of epipodal matrices. However there are some difficulties to overcome. In particular, projecting one support orthopodally to another one is not canonical.

Cryptanalysis

In the full version of this work we introduce a hybrid (LeeBrickellBabai) of the Lee-Brickell information set decoding algorithm [LB88], and the size-reduction algorithm (after LLL). We show heuristically that for common decoding parameters this leads to a small polynomial (in n) speed-up over Lee-Brickell. This contribution is meant to show that this algorithmic reduction theory for codes is compatible with existing techniques, and can, in principle bring improvements. By itself, this hybrid LeeBrickellBabai algorithm with LLL preprocessing is only tenuously faster than the original algorithm. Time-memory trade-offs such as [Ste88; Dum91; MMT11; BJMM12; MO15; BM18] admittedly provide much more substantial speed-ups in theory, and currently hold the records in practice [ALL19].

However, the lattice literature has much stronger reduction algorithms to offer than LLL [Sch87; GHKN06; GN08a; LN14; DM13]; this work opens their adaptation to codes as a new research area, together

with the study of their cryptanalytic implications. Furthermore, it is not implausible that reduction techniques may be compatible with memory intensive techniques [Ste88; Dum91; MO15]; this is the case in the lattice cryptanalysis literature [Duc18].

Further algorithmic translations

Still based around the same fundamental domain, a central algorithm for lattices is the Branch-and-Bound enumeration algorithm of Finkle and Pohst [FP85], which has been the object of numerous variations for heuristic speed-ups [GNR10]. While the hybrid algorithm LeeBrickellBabai may be read as an analogue of the random sampling algorithm of Schnorr [Sch03; AN17], a more general study of enumeration techniques for codes would be interesting.

Both for codes and lattices, there are other natural fundamental domains than the size-reduction studied in this paper. For lattices we have the (non-rectangle) parallelepiped $\mathcal{P}(\mathbf{B})$ provided with the so-called "simple rounding" algorithm $\mathbf{x} \mapsto \mathbf{x} - \lfloor \mathbf{x} \cdot \mathbf{B}^{-1} \rceil \cdot \mathbf{B}$ [Bab86]. For codes we have a domain of the form $\mathbb{F}_2^{n-k} \times \{0\}^k$ for each information set $\mathcal{I} \subset [\![1, n]\!]$ of size k given by Prange's algorithm [Pra62]. It is tempting to think they could be in correspondence in a unified theory for codes and lattices.

Another fundamental domain of interest is the Voronoi domain, which is naturally defined for both codes and lattices. In the case of lattices there are algorithms associated with it known as iterative slicers. Rather than operating with a basis, the provable versions of this algorithm operates with the (exponentially large) set of *Voronoi-relevant* vectors [MV13], while heuristic variants can work with a (smaller, but still exponential) set of short vectors (see chapter 5). We are not aware of similar approaches in the code literature.

Cryptographic design

Some of the developed notions could have application in cryptographic constructions as well, in particular for trapdoor sampling [GPV08; DST19]. Indeed, the Gaussian sampling algorithm of [GPV08] is merely a careful randomisation of the size-reduction algorithm, and the variant of Peikert [Pei10] is a randomisation of the "simple rounding" algorithm discussed above. It requires knowing a basis with a

good profile as a trapdoor. While the construction of a sampleable trapdoor function has finally been realised [DST19], the method and underlying problem used are rather ad-hoc. We note in particular that the underlying generalized $(U, U+V)$-codes admit bases with a peculiar profile, which may explain their fitness for trapdoor sampling. The algorithmic reduction theory proposed in this work appears as the natural point of view to approach and improve trapdoor sampling for codes.

Bounds

Beyond cryptography, this reduction theory may be of interest to establish new bounds for codes. In particular we emphasize the notion of *higher weight* [Wei91; TV95] as an analogue of the notion of the density of sub-lattices [Bog01]; the latter are subject to the so-called Rankin-bound, generalizing Hermite's bound on the minimal distance of a lattice.

Duality

Also on a theoretical level, one intriguing question is how this reduction theory interacts with the notion of duality for codes. In particular, for lattices, the dual of an LLL reduced basis of the primal lattice is (essentially) an LLL reduced basis of the dual lattice. One could wonder whether this also holds for codes; however, while there is a notion of a dual code, their doesn't seem to be a 1-to-1 correspondence between primal and dual bases. The notion of orthopodality does allow to introduce (non-unique) dual bases for codes, with similar geometric properties as the (reversed) dual basis. For lattices this leads to a correpondence $\|\tilde{\mathbf{b}}_i\| = \|\tilde{\mathbf{d}}_i\|^{-1}$; for codes the similar concept seems to be the correspondence between the $[|\mathbf{b}_i^+|, 1]$ repetition code generated by $\mathbf{b}_i^+$, and the $[|\mathbf{b}_i^+|, |\mathbf{b}_i^+| - 1]$ even code generated by some $\mathbf{d}_{j_1}^+, \ldots, \mathbf{d}_{j_{|\mathbf{b}_i^+|-1}}^+$. The construction allows for some choices, such as which basis to use for the even code. Under some light conditions this gives for $\ell_i \geq 2$ a primal-dual profile correspondence of $(\ell_i) \leftrightarrow (2, 1, \ldots, 1)$ with $\ell_i - 2$ ones. We leave a further investigation of the concept of a dual basis, and self-duality of the presented LLL algorithm to future work.

Another remark is that it may also be natural to consider Branch-and-Bound enumeration algorithms working with a reduced basis of the dual code rather than a basis of the primal code, at least if self-duality can not be established. This may be advantageous in certain regimes.