UNIVERSITY
OF WOLLONGONG
AUSTRALIA

Usability of structured lattices for a post-quantum cryptography: practical computations, and a study of some real Kummer extensions

Andrea Lesavourey

This thesis is presented as part of the requirements for the conferral of the degree:

Doctor of Philosophy in Computer Science

Supervisors:
Pr W. Susilo & Dr T. Plantard

The University of Wollongong
School of Computing and Information Technology

This research has been conducted with the support of an Australian Government Research Training Program Scholarship.

Declaration

I, *Andrea Lesavourey*, declare that this thesis is submitted in partial fulfilment of the requirements for the conferral of the degree *Doctor of Philosophy in Computer Science*, from the University of Wollongong, is wholly my own work unless otherwise referenced or acknowledged. This document has not been submitted for qualifications at any other academic institution.

Andrea Lesavourey

July 27, 2021

Abstract

Lattice-based cryptography is an excellent candidate for post-quantum cryptography, i.e. cryptosystems which are resistant to attacks run on quantum computers. For efficiency reason, most of the constructions explored nowadays are based on structured lattices, such as module lattices or ideal lattices. The security of most constructions can be related to the hardness of retrieving a short element in such lattices, and one does not know yet to what extent these additional structures weaken the cryptosystems. A related problem – which is an extension of a classical problem in computational number theory – called the Short Principal Ideal Problem (or SPIP), consists of finding a short generator of a principal ideal. Its assumed hardness has been used to build some cryptographic schemes. However it has been shown to be solvable in quantum polynomial time over cyclotomic fields, through an attack which uses the Log-unit lattice of the field considered. Later, practical results showed that multiquadratic fields were also weak to this strategy.

The main general question that we study in this thesis is

To what extent can structured lattices be used to build a post-quantum cryptography?

Such a question encompass two dimensions: *practicability* and *security*. To study this general question, one can follow several directions.

1. Study algebraically structured lattices such as ideal lattices, especially in terms of security.

2. In case algebraically structured lattices reveal themselves to be problematic, study lattices based on structures which cannot be linked to algebraic or arithmetical constructions such as number fields.

3. Improve computations over number fields, to help following the previous direction especially in a practical point of view.

We follow these main ideas, and this thesis is as follows.

We study the possibility of constructing an encryption scheme based on matrices called diagonally dominant matrices. The structure of these matrices is based on standard linear algebra properties. It is not linked to an underlying algebraic construction such as a polynomial ring or a number field.

We follow the first direction by studying the SPIP over some real Kummer extensions, and generalise the work done over multiquadratic fields. We show that these fields have a structure which allowsus to compute their unit group and retrieve a generator of a principal ideal efficiently. Our implementation of these algorithms allows us to evaluate in practice the possibility of solving the SPIP through the Log-unit lattice. We are also able to study the geometrical properties of the situation and compare it to the one over cyclotomic fields. In particular we are able to exhibit a subfamily of Kummer fields over which solving the SPIP is more difficult than over cyclotomic or multiquadratic fields. Our work also highlights the need of considering large degree number fields to be able to draw meaningful conclusions from practical results.

To study Kummer fields, we need to design and implement efficient algorithms. It was particularly necessary to study number fields of degree as large as possible. To this end, we develop practical algorithms to compute two important tasks over number fields: computing norms of ideals relative to an extension, and computing roots of polynomials. In both cases, we study certified algorithms running in polynomial time, and heuristic algorithms allowingfaster computations. We compare the efficiency of our implementations with the methods implemented in the softwares MAGMA or PARI/GP, and show that we obtain speed-ups for both tasks.

Acknowledgments

First of all, I would like to thank my supervisors Willy Susilo et Thomas Plantard for giving me the opportunity to do this research and for having always been there during the – almost – four years it lasted. Your coments and advice have been of great help. I have learned a lot by working with you during this PhD. Obviously, I also thank Australia and the University of Wollongong (UoW) which were great and vibrant environments.

I also wish to mention all the people who worked in UoW and were part of the doctoral process in any way. Additionally, I wish to warmly thank the referees Fabien Laguillaumie and Nicole Sutherland who agreed to revise the manuscript. Their comments helped me to improve the quality of the final thesis.

I want to thank everyone I met at UoW, especially those I shared room 6.214 with. I think about Arnaud and Vincent, Huy, Dung, Prianka and Partha. It was a pleasure to meet you, and I very much appreciated the fruitful (or not) scientific (or not) discussions that we had during my stay in Australia. I hope we will meet again sooner or later.

I also want to mention all the people I met outside of university who helped me go through these four years ; from Wollongong – Maggy and her doggos, Kris and Janne – or Sydney – everyone the French community (there are too many to do the full list here).

Other people I would like to shed some light on are all those who have led me to research or through academic steps at some point in my life. I particularly wish to thank all the teaching staff at University of Bordeaux, with whom I enjoyed studying mathematics. I have particularly fine memories of lectures from (not in order of any kind) Qing Liu, Marie-Line Chabanol, Karim Belabas and many others. A special thank you to Marie-Line Chabanol who directed me towards the Master Crypto in Bordeaux, Gilles Zémor who showed me an internship offer, and Christophe Nègre and Thomas Plantard who kindly agreed to work with me. These steps started the journey I am still pursuing today.

More broadly, I deeply thank my parents and my family for always being supportive – to the point of lending me money so I can go to Australia – and never telling me how stupid it is to try to do some research.

Many thanks to Nolwenn from the bottom of my heart, for coming to Australia, leaving friends and family to follow someone she just met. You were always there for me, and life would not be the same without you.

To end these thanks, I would like to mention all the people who have fought through the ages to make everyone's life better. Their struggles have led to more egalitarian societies, for example socialist France which provided relatively free education (the emphasis is on the relatively), health care, etc... Once again, thank you to all the workers around the world whose activities allow intellectuals like us to spend our days doing nothing. One can only hope that one day their work and their lives will be given due recognition and that the exploitation we all suffer from will end.

Cheers mates!

Contents

Chapter 1

Introduction

1.1 Cryptography: history and context

History of cryptology

The word *cryptology* etymologically means *"knowledge of secret"*, and designates the science dealing with secure data communication. It can be divided in two subcategories. First is *cryptography* which is the study on how to hide information, how to build processes – called *cryptosystems* or *schemes* – allowing data to be exchanged in a secure way. The *cryptanalysis* is the part of cryptology which deals with analysing said constructions to determine up to which point they are secure.

Cryptology has a long history. Indeed, one of the most famous cryptosystems is *Caesar's cipher* used by Julius Caesar which is a shift cipher where each letter in a text is shifted by a fixed number of ranks in the alphabet. More recently one can cite the example of *Enigma*, the machine used by the German army during WWII to encrypt data. The underlying scheme was notoriously broken by a team assembled by the British military force which included a father of computer science, Alan Turing.

Despite this long interest in secure communication, cryptology truly became a science during the twentieth century with its mathematical conceptualisation introduced by Shannon in 1975. Nowadays, it lives a massive boom with the numerical revolution. Indeed, we use remote communication and online transactions daily. A few examples of such tasks are using our mobile phone, secure e-mails, or contactless payment. As individuals, we need these exchanges to be done in a secure way, i.e. such that nobody can eavesdrop or steal our identity.

The advances in computer science and numerical technologies allows us to communicate among each other way faster than humans used to, but it also brings some downsides. One of them is the possibility of malicious entities collecting data without individual consent. Thus a developed cryptography publicly available is needed in order to allow individuals to hide and protect data as much as possible.

Modern cryptography

Modern cryptography is usually separated into two types. The first is called *secret key cryptography* or *symmetric cryptography*, and corresponds to the cryptography which has been used historically. In secret key cryptosystems, two people wishing to communicate safely have to agree on a *secret key* which will allow them to both *encrypt* and *decrypt* the information they will exchange. They have to be the only ones to know the key, which is then secret to the outside world, especially to malicious entities. However, to agree on a key, they need to meet or go through a trusted third-party. These constraints are not compatible with our everyday use of cryptography.

The second type is called *public key cryptography* or *asymmetric cryptography*, and is relatively recent since it was introduced in 1976 by W. Diffie and M. Hellman [36]. With such schemes, if one entity wishes to be able to *receive* secure communications, it generates *a secret and a public key* and publicly reveals the public key. This way, anyone can encrypt data using it, and only the secret key holder is able to decrypt. Note that such a cryptosystem allows two parties to securely exchange a key for later use in a secret key cryptosystem. Generally speaking, the security of public key protocols rely on the supposition that some underlying mathematical problem is hard to solve. The most common problems are integer factorisation as in the RSA protocol [88] and discrete logarithm [36]. Despite extensive research, the best classical algorithms solving these problems are subexponential in the size of the entry. They are also widely used because of their simplicity and efficiency. These qualities are essential for an everyday usage of cryptography.

Post-quantum cryptography

However the security of public-key cryptography is now under threat from a relatively new tool, namely quantum computers. They follow different laws of computation than classical computers – that are called *quantum computing* – and they can solve efficiently some mathematical problems that we still do not know how to solve with a classical computer. In particular, P. Shor proved in its breakthrough paper [95] that a quantum computer can solve in polynomial time the factorisation and the discrete logarithm problems. Until recently we could still consider schemes based on these problems to be safe since we are not yet successful in building such a quantum computer. However the progress made in the last decades led the U.S. National Security Agency (NSA) to declare in 2015 that it considered quantum computing as an upcoming threat. It also called for a change in the orientation of research to focus on developing a cryptography which could be resistant to quantum computers, and that is commonly called *post-quantum cryptography* [10]. Subsequently the U.S. National Institute of Standards and Technology (NIST) announced in 2016 a call for standardisation for post-quantum cryptography [1], which is at the time of writing in its Round 3.

Lattice-based cryptography

Several techniques have been considered to build a post-quantum cryptography upon. Among them are error-correcting codes, multivariate polynomials, hash-functions and Euclidean lattices [10]. The last one is among the most popular options. To give an idea, five of seven of the final candidates for Round 3 of the NIST process are based on lattices.

A Euclidean lattice is a *discrete subgroup of* $\mathbb{R}^n$. It can always be described as the integral linear combinations of a set of linearly independent vectors $(b_1, \ldots, b_r)$, which is called a *basis* of the lattice. The classical problems on lattices are the *Shortest Vector Problem* (SVP) which consists of finding a non zero lattice vector with minimal norm and the *Closest Vector Problem* for which, given a vector t of $\mathbb{R}^n$, one has to find the vector closest to t in the lattice. These problems are known to be NP-hard over random instances [2, 39]. Usually in cryptography, one is more interested in their approximate versions. Hence the following definition of the approximation problems. The γ*-approximate Shortest Vector Problem* (SVP$_\gamma$) consists of finding a lattice vector whose norm is smaller than γ times the minimal norm of a lattice vector. The γ*-approximate Closest Vector Problem* (CVP$_\gamma$) consists of finding a lattice vector which is at a distance to t smaller than γ times the distance of t to the lattice. The complexities of SVP$_\gamma$ and CVP$_\gamma$ decrease when γ is increasing, going from NP-hard for constant γ to P for γ exponential in the rank of the lattice.

The complexities mentioned are true over random lattices, which is mostly not the case of lattices used in cryptography. One typically wants a trapdoor to exist in order to be able to decrypt. In cryptosystems based on lattices, the private key is generally a “good” basis, i.e. composed by relatively short vectors which are almost orthogonal to each other. The public key is then a “bad” basis, for example the Hermite Normal Form (HNF) representing the lattice. The private key then allows to decrypt efficiently and the security of the scheme relies on the assumed hardness to retrieve a good basis from the bad one.

The drawback of general lattices is their efficiency, which is an important parameter for public key cryptography. They are represented by matrices therefore both the storage and the computation cost are expensive. In order to cope with this, one can use special lattices with an extra algebraic structure. Several cryptosystems based on different structures have been proposed and studied over the years. One can cite the work of C. Gentry [45] for a fully homomorphic scheme, of N. Smart and F. Vercauteren [97], the NTRU cryptosystem [54], and the more recent Ring-Learning With Errors (RLWE) [66] or Module-Learning With Errors (MLWE) [61, 23]. The two first schemes consider lattices which are also principal ideals of a number field. Their security relies on the supposed hardness of retrieving a *short* generator of said ideal. This problem is called the *Short Principal Ideal Problem* (SPIP).The RLWE and RSIS primitives are based on *ideal lattices* and their security is linked to the SVP problem restricted to such lattices, i.e. the *Ideal Shortest Vector Problem* (ISVP). Finally the NTRU, MLWE and MSIS primitives rely on module lattices. The hardness of the two last can be linked to the one of the SVP restricted to them namely the *Module Shortest Vector Problem* (MSVP). Module lattices are lattices represented by block matrices such that each block is a basis matrix of an ideal lattice.

1.2 Goal and organisation of the research and thesis

General questions and directions

The main question that we consider regarding lattices and post-quantum cryptography is the following.

"Determine to what extent structured lattices can be used to build a post-quantum cryptography."

The first direction that one can follow to answer this question would be the following.

"Study algebraically structured lattices such as ideal lattices or module lattices, especially in terms of security.".

Then one can consider a second direction.

"Consider less structured lattices – which cannot be linked to an underlying algebraic structure such as a number field – and explore the possibility of building efficient schemes.".

Main objectives of the thesis

Regarding the first direction – which is the one mostly followed by the community – even though most of propositions discussed nowadays such as the candidates to the NIST process use the more complex structures (module lattices) it is still important to study the Short Principal Ideal Problem. First the SPIP is a classical problem in computational number theory [31], and is interesting in itself. Secondly, even though it is simpler than the ISVP, these two problems are linked. In particular, the SPIP is an intermediate task in some algorithms to solve the ISVP [9, 33, 79]. Then the techniques used to analyse this problem could be extended to analyse the ISVP. One could determine if some of the structures are weaker than others. Finally, as we mentioned, some cryptographic constructions rely directly on the supposed hardness of the SPIP.

A generic way of solving the SPIP is done in two steps. First find a generator of the ideal, then reduce this generator to recover a short generator. The first step can be done in quantum polynomial time [13], while the best classical algorithms run in subexponential time [31]. Moreover one can rewrite the second step as a reduction phase with respect to a lattice depending only on the chosen number field K, namely the *Log-unit lattice of* K. This lattice can also be computed in quantum polynomial time [38]. Thus, from a post-quantum perspective, the only interrogation is whether the second step can be carried out, i.e. if a given lattice problem can be solved. This strategy has been mentioned by Campbell et al. in [26]. Cramer et al. showed it can be efficiently and successfully performed over cyclotomic fields [34], then Bauch et al. studied multiquadratic fields and exhibited efficient *classical* (as opposed to quantum) algorithms allowing them to retrieve a generator of principal ideal with high probability [6].

Therefore, we focused on studying the SPIP for the reasons mentioned above, and more precisely we fixed the following goal.

G1: *"Determine to what extent the choice of number field influences the success of a recovering a short generator from another generator".*

Moreover we believe that since it can be difficult to analyse number theoretical problems, it is important to be able to do efficient computations. This way, we could study how some algorithms – which can be computed in quantum polynomial time – behave in practice. Finally, we aimed to study high dimensional number fields, i.e. dimensions of interest in cryptography. Such dimensions are out of reach of generic algorithms computing the main number theoretical objects that are needed for the study of the SPIP. That is why we focus on our second goal.

G2: *"Design and implement efficient algorithms allowing the study of high degree number fields.".*

Real Kummer extensions For all these reasons we extended the work done over multiquadratic fields to Kummer extensions of exponent p, where p is a prime integer. In particular we designed algorithms to compute the unit group of such number fields, retrieve a generator of a principal ideal and shorten it. We implemented these procedures in MAGMA [22] for real Kummer extensions of $\mathbb{Q}$ of exponent p, i.e. generated by p-th roots of integers, and real Kummer extensions of $\mathbb{Q}$ with two exponents – generated by p-th and q-th roots of integers where p and q are prime integers – in order to break the structure and discover if one can still solve the SPIP with a good probability. Moreover our implementation allows us to study the Log-unit lattice of these fields and classify them with respect to their security level. In particular we were able to exhibit a class of fields – real Kummer fields of dimension p^2 – over which the SPIP seems to be more difficult to solve than number fields already studied.

Faster computations in number fields Then as mentioned, part of the work is to improve computations over number fields to be able to handle high dimensions. During our study of Kummer extensions we had to design a procedure to compute p-th roots in these fields. It is inspired by an application of the LLL algorithm mentioned in the original paper [63]. We then extended our method to the computation of polynomial roots in number fields and compared it to the performance of the generic algebraic algorithm of K. Belabas [7]. Comparisons are made over different types of polynomials and number fields. Our implementation is in PARI/GP [76]. Another important task needed for our work on Kummer extensions is the computation of relative norms of ideals. We implemented two simple algorithms, one of which is inspired by algorithms of Cohen [31]. We compare their performances to the procedure implemented in MAGMA.

Finally, we mentioned that it is unknown to what extent extra algebraic structures can be used to solve problems over lattices, and that it is still important to consider more general lattices. This leads to our third goal.

G3: *"Study lattices without underlying algebraic structure to build efficient encryption schemes."*.

Diagonally dominant matrices An example of scheme based on lattices without a link to algebraic structures such as number fields is DRS [83]. It is a signature scheme which uses diagonally dominant matrices. We studied such matrices and showed that they can be used to construct an encryption scheme. In particular, we design reduction algorithms which can be more efficient than using LLL if properly implemented.

Organisation of the thesis

The rest of the thesis is organised as follows.

- Chapter 2 is dedicated to the description of the background and preliminaries.
- Chapter 3 deals with our work on diagonally dominant matrices and the extension of the DRS signature scheme to DRE encryption scheme.
- We describe in Chapter 4 the improvements we developed for computational tasks in number fields.
- We then develop our work on Kummer extensions in Chapter 5. It corresponds to the paper [64].

List of Symbols

Algorithmic

$\mathcal{M}(n)$	Cost of a multiplication of n-bits integers
$\widetilde{O}(f(n))$	$O(f(n)\lvert\log f(n)\rvert^c)$
$L_N(\alpha, c)$	$\exp\left((c+o(1))\log(N)^\alpha(\log\log N)^{1-\alpha}\right)$

Groups

$\langle S\rangle$	Subgroup generated by S
$H_1 \ltimes H_2$	Semi-direct product of H_1 and H_2
$o(g)$	Order of group element g

Matrices, vector spaces and lattices

$\lambda_1^{(p)}(\mathcal{L})$	Norm of a shortest non-zero vector of lattice $\mathcal{L}$, relative to l_p
$\mathcal{B}^\vee, \mathcal{L}^\vee$	Dual basis and lattice
$\mathrm{span}(S)$	Minimal vector space containing $S \subset \mathbb{R}^n$
$\mu^{(p)}(\mathcal{L})$	Covering radius of lattice $\mathcal{L}$, with respect to l_p
π_i	Projection operator onto $\langle b_1, \ldots, b_{i-1}\rangle_{\mathbb{R}}^\perp$
$B(x, R)$	Sphere of radius R centered in x
M_i	i-th row of matrix M

Number theory

$\mathrm{Hom}(L/K, \mathbb{C})$	Set of complex field embeddings of L which are congruent to the identity when restricted to K
$\mathrm{Hom}(L/K, \Omega)$	Set of K-linear field embeddings of $L \hookrightarrow \Omega$
$\mathrm{Inv}(S), L^S$	Subextension of L/K fixed by $S \subseteq \mathrm{Hom}(L/K, \mathbb{C})$

Ω	Algebraic closure of a number field extension L/K
$\mathfrak{D}(L/K)$	Relative different of L/K
$\mathfrak{d}(L/K)$	Relative discriminant of L/K
$\tilde{L}$	Galois closure of L
$\mathrm{Tr}_{L/K}, \mathrm{N}_{L/K}$	Trace and Norm maps, relative to extension L/K
σ_K	Minkowski's embedding of K
$\sigma_{L/K}$	Minkowski's embedding relative to L/K
$D_K(b_1, \ldots, b_n)$	Discriminant of $(b_1, \ldots, b_n)$ with respect to L/K
$D_K, D(K)$	Absolute discriminant of K

Rings

(a)	Ideal generated by the ring element a
A^*	$A \setminus \{0\}$
a^σ	Image of a under action of σ
$A^\times$	Group of invertible elements of the ring A
$Z_B(f)$	Roots of $f(X)$ in B

Others

$(\cdot \mid \cdot)$	Inner product
$\cong$	Isomorphism
$\delta_{(\mathscr{P}(n))}$	Indicator function of proposition $\mathscr{P}(\cdot)$, i.e. is equal to 1 if $\mathscr{P}(n)$ is true and 0 otherwise
$\langle b_1, \ldots, b_n \rangle_A$	$Ab_1 + \ldots, Ab_n$
$[\![a, b]\!]$	Interval in the integers, i.e. $\{x \in \mathbb{Z} \mid a \leqslant x \leqslant b\}$
$\mathcal{F}(m)$	Fundamental parallelepiped of $\mathbb{R}^n$ defined by $m \in \mathbb{R}^n$ – for example $[0, m[^n$
$\mathcal{P}(m)$	Set of prime numbers dividing $\prod_{i=1}^n m_i$, where $m \in \mathbb{Z}^n$
$\otimes$	Tensor product

Chapter 2

Background and preliminaries

2.1 Notations and recalls

First let us mention that vectors are considered to be *row-vectors* throughout this thesis.

- For any $p \in \mathbb{N}^* \cup \{\infty\}$, we denote classically the l_p norm on $\mathbb{R}^n$ to be $l_p(x) = \sqrt[p]{\sum_{i=1}^{n} |x_i|^p}$ if p is an integer and $l_p(x) = \max\{|x_i| \mid i \in [\![1, n]\!]\}$ if $p = \infty$. We also denote l_p by $\| \cdot \|_p$.

- Given $(a, b) \in \mathbb{Z}^2$ we will denote by $[\![a, b]\!]$ the corresponding segment of $\mathbb{Z}$. Concerning intervals, we follow the convention (common in french literature) which uses only square brackets. As an example, $[a, b[$ is the same as $\{x \in \mathbb{R} \mid a \leqslant x < b\}$.

- Given a ring A and a matrix M in $\mathrm{M}_{n,m}(A)$, M_i will designate the i-th row of M. The matrix will be generally defined as $M = [m_{i,j}]_{\substack{i \in [\![1,n]\!] \\ j \in [\![1,m]\!]}}$. When it is not the case, the elements of M will be denoted by $M_{i,j}$.

- Given a morphism $\sigma : A \mapsto B$, and x an object which can be identified with a vector of A^n, we will write x^σ for the image of x under the action of σ. In particular if $f(X) = f_0 + \cdots + f_n X^n \in A[X]$ then f^σ is the polynomial of $B[X]$ equal to $\sigma(f_0) + \cdots + \sigma(f_n) X^n$.

- Given A, B two rings and $f(X) \in A[X]$, we will denote by $Z_B(f)$ the set of roots of $f(X)$ which belong to B, i.e.

$$Z_B(f) = \{x \in B \mid f(x) = 0\}.$$

- Given A a ring and when it makes sense, we will write $\langle b_1, \ldots, b_n \rangle_A$ for $Ab_1 + \ldots, Ab_n$. When there is no ambiguity, we will simply use $\langle b_1, \ldots, b_n \rangle$
- Given $S \subset \mathbb{R}^n$, we will write $\mathrm{span}(S)$ for the minimal sub-vector space of $\mathbb{R}^n$ containing S.
- Let G be a group and $S \subseteq G$. We denote by $\langle S \rangle$ the subgroup of G generated by S.
- Given a function $f : \mathbb{N} \to \mathbb{R}_+$, one defines $\tilde{O}(f(n))$ as $O(f(n)|\log f(n)|^c)$ for some $c > 0$.
- Complexities are often expressed by mean of the L-notation. Given a variable N and two constants α and c with $\alpha \in [0,1]$ and $c > 0$, $L_N(\alpha, c)$ is defined by

$$\exp\left((c + o(1)) \log(N)^{\alpha} (\log\log N)^{1-\alpha}\right).$$

Gram-Schmidt orthogonalisation

An important and classical computation in linear algebra is the Gram-Schmidt orthogonalisation (GSO). It allows transforming a free family of a vector space (with a scalar product) into in an orthogonal family. The naive algorithm can be found in Algorithm 1.

Algorithm 1 GSO

```
Require: A free family $\mathcal{B} = \{b_1, .., b_r\}$ of $\mathbb{R}^n$
Ensure: A family $\tilde{\mathcal{B}} = \{\tilde{b}_1, .., \tilde{b}_n\}$ with $\tilde{b}_1 = b_1$ and $(\tilde{b}_i \mid \tilde{b}_j) = 0$ for $i \neq j$
1: $\tilde{\mathcal{B}} \leftarrow \mathcal{B}$
2: for $i = 1$ to $r$ do
3:     $\tilde{b}_i \leftarrow b_i$
4:     for $j = 1$ to $i - 1$ do
5:         $\tilde{b}_i \leftarrow \tilde{b}_i - \frac{(b_i \mid \tilde{b}_j)}{(\tilde{b}_j \mid \tilde{b}_j)} \tilde{b}_j$
6:     end for
7: end for
8: return $\tilde{\mathcal{B}}$
```

Let $\mathcal{B}$ be a family represented by a matrix B, $\tilde{\mathcal{B}}$ the orthogonal family returned by the GSO process and $\tilde{B}$ its representative matrix. For any $(i,j) \in [\![1, r]\!] \times [\![1, n]\!]$ one usually denotes by $\mu_{i,j}$ the coefficient $\frac{(b_i|\tilde{b}_j)}{(\tilde{b}_j|\tilde{b}_j)}$ occurring in Algorithm1. First remark that the process depends on the order of $\mathcal{B}$. Then we have the relation

$$B = \begin{bmatrix} 1 & 0 & 0 & \dots & 0 \\ \mu_{2,1} & 1 & 0 & & 0 \\ \mu_{3,1} & \mu_{3,2} & 1 & \ddots & \vdots \\ \vdots & \vdots & \ddots & \ddots & 0 \\ \mu_{r,1} & \mu_{r,2} & \dots & \mu_{r,r-1} & 1 \end{bmatrix} \tilde{B}. \tag{2.1}$$

We will designate by G the unnamed matrix above. We also consider that G can be returned by Algorithm 1. Moreover we will also denote by π_i the orthogonal projection onto $\langle b_1, \dots, b_{i-1}\rangle_{\mathbb{R}}^{\perp} = \langle \tilde{b}_1, \dots, \tilde{b}_{i-1}\rangle_{\mathbb{R}}^{\perp}$, the orthogonal supplement of the sub-vector space spanned by the $(i-1)$-th first elements of the basis.

Definition 2.1 (GSO)**.** Given a basis $\mathcal{B}$, we will call the basis $\widetilde{B}$ outputted by Algorithm 1 *the GSO* of $\mathcal{B}$.

2.2 Lattices

We refer the reader interested in more in-depth presentations on Euclidean lattices to [69, 32].

2.2.1 First meeting with lattices

Definition 2.2. A *Euclidean lattice* is a discrete subgroup of $\mathbb{R}^n$ where n is a positive integer. We say a lattice is an *integral* lattice when it is a subgroup of $\mathbb{Z}^n$ or *rational* when it is in $\mathbb{Q}^n$. A *basis* of a lattice $\mathcal{L}$ is a basis of $\mathcal{L}$ as a $\mathbb{Z}$-module. The cardinal of said basis is called the *rank* of the lattice.

Notation. Given a matrix B we will write $\mathcal{L}(B)$ the lattice generated by its row vectors.

Definition 2.3 (Span of a lattice)**.** Let $\mathcal{L}$ be a lattice generated by a basis $\mathcal{B} = (b_1, \dots, b_r)$. We call *span of* $\mathcal{L}$ and write $\operatorname{span}(\mathcal{L})$ the vector space generated by $\mathcal{B}$, i.e. $\langle \mathcal{B}\rangle_{\mathbb{R}} = \mathbb{R}b_1 + \cdots + \mathbb{R}b_r$.

As for vector spaces, a lattice has an infinite number of bases, at least when its rank r is greater than 2. More precisely, consider $r \leqslant n$ two integers, together with B and B' two matrices in $\mathrm{M}_{r,n}(\mathbb{R})$ which row vectors are independent. Then one has

$$\mathcal{L}(B) = \mathcal{L}(B') \iff \exists U \in \mathrm{GL}_r(\mathbb{R}) \mid B' = UB.$$

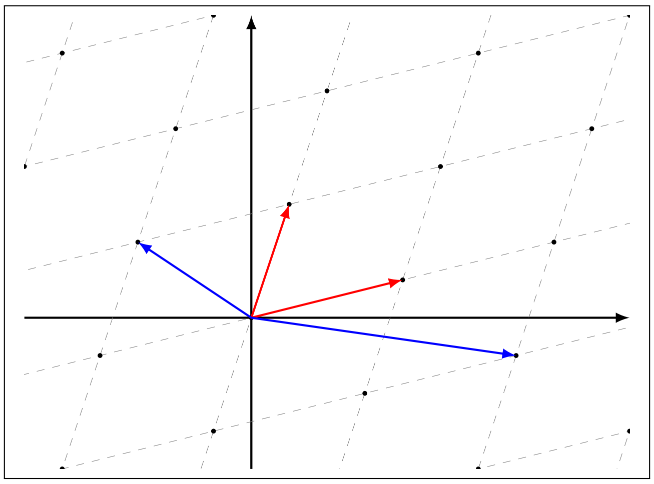


Figure 2.1: Two bases of the same lattice

This is illustrated in Figure 2.1, where two bases of the same lattice of $\mathbb{R}^2$ are plotted.

There are invariants independent from the choice of the basis. They are mainly geometrical parameters of the lattice.

Lemma 2.1. *Consider a matrix B, $U \in \mathrm{GL}_r(\mathbb{R})$ and $B' = UB$. Then the following equality holds:* $\det(BB^\mathsf{T}) = \det(B'B'^\mathsf{T})$.

Definition 2.4. Consider a Euclidean lattice $\mathcal{L}$ with basis matrix B. Then the *determinant* of $\mathcal{L}$, denoted by $\det(\mathcal{L})$, is the value $\sqrt{\det(BB^\mathsf{T})}$.

Notation. The determinant of a lattice $\mathcal{L}$ is also called its *volume* because it is the volume of the fundamental domain defined by the vectors of one of its bases. Thus it is also written $\mathrm{vol}(\mathcal{L})$.

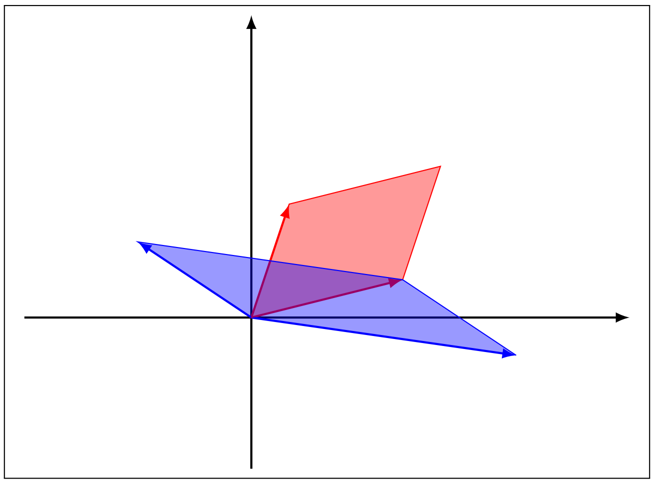


Figure 2.2: Two bases, same volume

One can see in Figure 2.2 that the two bases define two different fundamental domains, which however have the same volume. These two geometrical situations induce two different hardness of solving problems over the same lattice.

Dual lattice

Attached to any lattice is another lattice called the dual lattice.

Definition 2.5 (Dual basis)**.** 1. Consider a free family $\mathcal{B} = (b_1, \dots b_r) \subseteq \mathbb{R}^n$. The dual family of $\mathcal{B}$ denoted by $\mathcal{B}^\vee$ is the family $(b_1^\vee, \dots, b_r^\vee) \subseteq \mathbb{R}^n$ defined by

$$\forall (i,j) \in [\![1, r]\!]^2, (b_i^\vee \mid b_j) = \delta_{i,j}. \tag{2.2}$$

2. Given a lattice $\mathcal{L} = \mathcal{L}(B)$, the *dual lattice of* $\mathcal{L}$ is the lattice generated by $\mathcal{B}^\vee$. It is denoted by $\mathcal{L}^\vee$.

Remark 1. Given a lattice $\mathcal{L}$, it follows directly from Definition 2.5 that the dual lattice verifies $\mathcal{L}^\vee = \{x \in \mathbb{R}^n \mid \forall y \in \mathcal{L}, (x \mid y) \in \mathbb{Z}\}$. This characterisation can be used as an alternative definition.

The dual lattice has special properties linked to the lattice.

Proposition 2.1. *Let $\mathcal{L}$ be a lattice and B the matrix of a basis of $\mathcal{L}$. The following are true.*

1. *The matrix $(BB^\mathsf{T})^{-1}B$ is the matrix of a basis of $\mathcal{L}^\vee$. When $\mathcal{L}$ is full-rank, this matrix becomes $B^{-\mathsf{T}} = (B^\mathsf{T})^{-1}$.*

2. $\det(\mathcal{L}^\vee) = \frac{1}{\det(\mathcal{L})}$.

Hermite Normal Form

Despite an infinite number of bases, there is a canonical way of representing *rational* lattices. The presentation will be done over *integral* lattices, but the results from the last type can be used on the former. Given a rational lattice $\mathcal{L}$, simply remark that there is a minimal $d \in \mathbb{Z}$ – called the denominator of $\mathcal{L}$ – such that $d\mathcal{L}$ is integral.

Definition 2.6 (Hermite Normal Form [31])**.** Consider $H = [h_{i,j}]_{(i,j)} \in \mathrm{M}_{m,n}(\mathbb{Q})$. We say that H is in *Hermite Normal Form* (HNF) if there is $r \in [\![1, n]\!]$ and a strictly increasing map $f : [\![1, r]\!] \to [\![1, n]\!]$ which satisfy the following conditions.

1. For all $i_0 \in [\![1, r]\!]$,

    - $h_{i_0, f(i_0)} > 0$;

- for all $i \in [\![1, m]\!] \setminus \{i_0\}$, $h_{i,f(i_0)} = 0$ if $i > i_0$ and $0 \leqslant h_{i,f(i_0)} < h_{i_0,f(i_0)}$ otherwise.

2. The last $m - r$ rows are equal to 0.

Remark 2. One important case is when $m = n = r$, so for all $i \in [\![1, n]\!], f(i) = i$. Then H in HNF is an upper triangular matrix with non zero coefficients on the diagonal.

One can show that any integral lattice basis is equivalent to a unique matrix in Hermite Normal Form.

Theorem 2.1 ([31])**.** *Consider $M \in \mathrm{M}_{m,n}(\mathbb{Z})$. Then there are $U \in \mathrm{GL}_m(\mathbb{Z})$ and a unique H in HNF such that $H = UM$.*

Definition 2.7. Given a lattice $\mathcal{L}$ we will call *Hermite Normal Form* of $\mathcal{L}$ or HNF of $\mathcal{L}$ the matrix H in HNF equivalent to a basis B of $\mathcal{L}$. It is denoted by $\mathrm{HNF}(\mathcal{L})$.

The HNF of a basis is particularly useful for several computations on lattices. One can use it to compare them (inclusion), solve linear systems, test if an element is in a lattice [31] for example. Moreover it can be computed in polynomial time [58], and is still an active area of research [80, 65].

Because of these properties, the HNF of a full rank lattice is a good candidate for a public key in lattice based cryptography [68].

Geometrical properties, size and volumes

As we will see later, important properties and problems over lattices are geometrical ones. They essentially are about norms of vectors and volumes. Let us state some results in order to set the overall context.

Definition 2.8 (Minima)**.** Consider $\mathcal{L}$ a lattice of rank r. Then for any $i \in [\![1, r]\!]$, its *$i-th$ minimum* is

$$\inf \left\{ R \in \mathbb{R}_+ \mid \dim \operatorname{span}(\mathcal{L} \cap \bar{B}(0, R)) \geqslant i \right\}$$

and is denoted by $\lambda_i(\mathcal{L})$.

Remark 3. First, the first minimum $\lambda_1(\mathcal{L})$ is the norm of a vector in $\mathcal{L} \setminus \{0\}$ with minimal norm. Then the definition depends on the choice for a norm. We mostly use the standard Euclidean norm. If another l_p norm is considered, the i-th minimum will be denoted by $\lambda_i^{(p)}(\mathcal{L})$.

We can give a minimal bound of λ_1 using the Gram-Schmidt orthogonalisation of a basis we have at hand.

Theorem 2.2. *Let $B = (b_i)_{i \in [\![1,r]\!]}$ be a basis of a lattice $\mathcal{L}$. Then $\lambda_1(\mathcal{L}) \geqslant \min\{\|\tilde{b_i}\| \mid i \in [\![1,r]\!]\}$.*

Then one can analyse further the geometry to obtain approximate values on λ_1.

Theorem 2.3 (Minkowski's convex body theorem [70])**.** *Let $\mathcal{L}$ be lattice of $\mathbb{R}^n$ of rank r, and S a convex subset of $\mathbb{R}^n$ such that $\mathrm{vol}(S) > 2^r \cdot \det \mathcal{L}$. Then there is $x \in \mathcal{L} \cap S \setminus \{0\}$.*

Now considering that an approximate value of the volume of a hyperball $B_r(0,R)$ is $V_r(R) \sim \frac{1}{\sqrt{r\pi}}(\frac{2\pi e}{r})^{r/2}R^n$, one obtains the following approximation of λ_1.

Heuristic 2.1 (Gaussian heuristic)**.** *Given a lattice $\mathcal{L}$ of rank r, an approximate value for $\lambda_1(\mathcal{L})$ is*

$$\lambda_1(\mathcal{L})_{\text{gauss}} = \sqrt{\frac{r}{2\pi e}} \times \sqrt[r]{\mathrm{vol}(\mathcal{L})}. \tag{2.3}$$

Definition 2.9 (Hermite's factors)**.** The r-th Hermite's factor is

$$\gamma_r = \max\left\{ \left(\frac{\lambda_1(\mathcal{L})}{\mathrm{vol}(\mathcal{L})}\right)^2 \mid \mathcal{L} \text{ is a lattice of rank } r \right\}$$

Another important value attached to a lattice is its covering radius.

Definition 2.10 (Covering radius)**.** Let $\mathcal{L}$ be a lattice of rank r. Then its covering radius $\mu(\mathcal{L})$ is defined as follows.

$$\mu(\mathcal{L}) = \max\{d(x,\mathcal{L}) \mid x \in \mathrm{span}(\mathcal{L})\}$$

Alternatively one can define the covering radius as the minimal value R such that the ambient space is fully covered by balls of radius R and centered in lattice points.

Remark 4. As it is the case for lattice minima, the covering radius depends on the choice of norm. We will therefore denote the covering radius by $\mu^{(p)}$ when we consider the norm l_p.

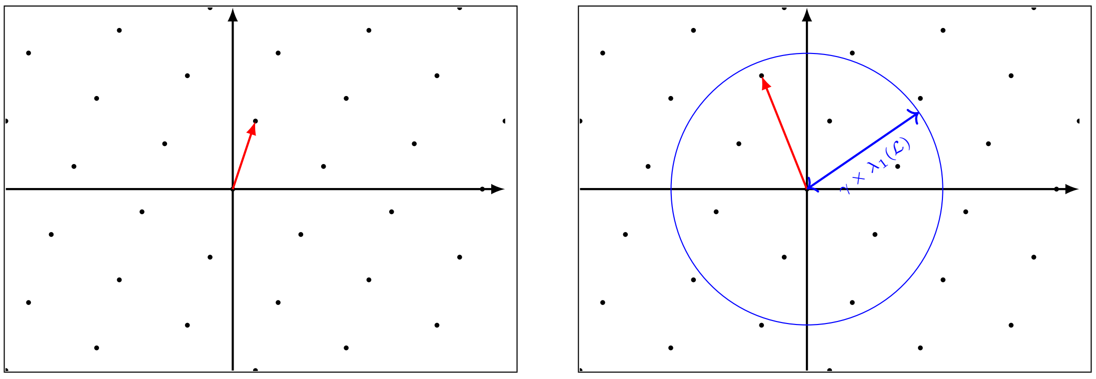
2.2.2 Hard problems on lattices

Let us now describe the major computational problems over lattices. We have seen that questions arising when studying lattices concern short vectors. It is then coherent that the first problem concerns the shortest one.

Definition 2.11 (**SVP**: Shortest Vector Problem)**.** Given a basis B of a lattice $\mathcal{L}$ of rank r, find $u \in \mathcal{L} \setminus \{0\}$ such that $\|u\| = \lambda_1(\mathcal{L})$.

The problem is NP-hard [2, 39] over *general* lattices. Usually in cryptography, one does not need to find the shortest vector but a close approximation. How close that approximation can be in practice is one of the central points of research in lattice-based cryptography. Hence the following definition of the approximate version of the problem.

Definition 2.12 ($\mathbf{SVP}_\gamma$: γ-approximate Shortest Vector Problem)**.** Given a basis B of a lattice $\mathcal{L}$ of rank r and an approximation factor γ, find $u \in \mathcal{L} \setminus \{0\}$ such that $\|u\| \leqslant \gamma \times \lambda_1(\mathcal{L})$.



(a) SVP **(b)** SVP_γ

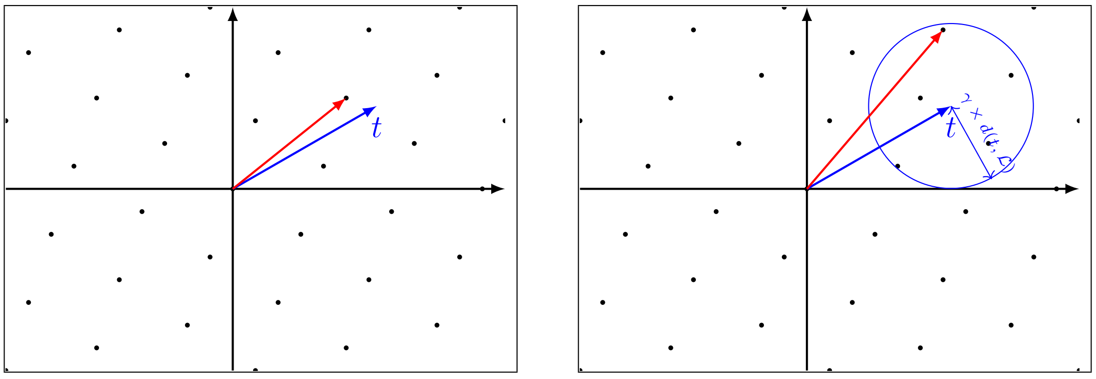
Figure 2.3: SVP and SVP_γ

The second important problem on lattices is the Closest Vector Problem.

Definition 2.13 (**CVP**: Closest Vector Problem)**.** Given a basis B of a lattice $\mathcal{L}$ of rank r and $t \in \mathbb{R}^n$, find $u \in \mathcal{L}$ such that $\forall v \in \mathcal{L}$, $\|t - u\| \leqslant \|t - v\|$.

The CVP is also known to be NP-hard [39]. As there is an approximate version of SVP, the same exists for CVP.

Definition 2.14 ($\mathbf{CVP}_\gamma$: γ-Approximate Closest Vector Problem)**.** Given a basis B of a lattice $\mathcal{L}$ of rank r, an approximation factor γ and $t \in \mathbb{R}^n$, find $u \in \mathcal{L}$ such that $\forall v \in \mathcal{L}$, $\|t - u\| \leqslant \gamma \|t - v\|$.



(a) CVP **(b)** CVP_γ

Figure 2.4: CVP and CVP_γ

The last problems that we will be interested in are the Bounded Distance Decoding with it approximate version, as well as the Guaranteed Distance Decoding.

Definition 2.15 (**BDD**: Bounded Distance Decoding)**.** Given a basis B of a lattice $\mathcal{L}$ and a point x such that $d(x,\mathcal{L}) < \lambda_1(B)/2$, find the lattice vector $v \in \mathcal{L}$ closest to x.

Definition 2.16 (**BDD$_\gamma$**: γ-Approximate Bounded Distance Decoding)**.** Given a basis B of a lattice $\mathcal{L}$, a point x and a approximation factor γ ensuring $d(x,\mathcal{L}) < \gamma\lambda_1(B)$ find the lattice vector $v \in \mathcal{L}$ closest to x.

In practice, one considers BDD$_\gamma$ for $\gamma < \frac{1}{2}$. This ensures that there is only one lattice vector v satisfying $d(x,v) < \gamma\lambda_1(B)$.

One can remark that the BDD is a version of the CVP with the knowledge that the target is close to the lattice.These problems depend on two parameters. First the basis given as an input and second the approximation factor γ for the approximate versions. The complexity is decreasing when γ increases, except for BDD$_\gamma$ which is in fact harder. Moreover a better basis, i.e. with short vectors and relatively orthogonal one to each other allows the problems to be solved faster or up to a better approximation factor.

A last problem close to the BDD is the Guaranteed Distance Decoding.

Definition 2.17 (**GDD$_\gamma$**: γ-Guaranteed Distance Decoding)**.** Given a basis B of a lattice $\mathcal{L}$, *any* vector v in span$(\mathcal{L})$ and an approximation factor γ, find $w \in \mathcal{L}$ such that $\|w - v\| \leqslant \gamma\lambda_1(\mathcal{L})$.

Remark the differences between the BDD and the GDD. The vector given as input of the latest can be any vector of the ambient space, not just the ones particularly close to $\mathcal{L}$.

2.2.3 Algorithms for lattices

We already saw that lattice problems are geometrical in nature, and that some of a lattice's properties can be linked to the GSO of a given basis. Moreover remark that if $\mathcal{L} = \mathcal{L}(\mathcal{B})$ with $\mathcal{B}$ being an orthogonal basis, then the problems can be easily solved. Finally, intuitively if a basis $\mathcal{B}$ is composed by vectors globally more orthogonal to each other than the vectors of another basis matrix $\mathcal{B}'$ are, then the vectors of $\mathcal{B}$ will be globally shorter than the vectors of $\mathcal{B}'$. This is due to the fact that the volume of the fundamental domains defined by both bases are equal to vol$(\mathcal{L})$.

Therefore, given a basis it is natural to orthogonalise it as much as possible to solve problems on lattices. Since it likely results in a basis with shorter vectors, we will call such a process a *reduction process* or *reduction algorithm.*

First let us define a weak notion of basis reduction due to Hermite [51].

Definition 2.18 (Size-reduce)**.** A basis $\mathcal{B} = (b_1, \dots, b_r)$ of a lattice is said to be *size-reduced* if its GSO satisfies the following condition:

$$\forall i \in [\![1, r]\!], \forall 1 \leqslant j < i, |\mu_{i,j}| \leqslant \frac{1}{2}.$$

Geometrically speaking, if one recalls that $\mu_{i,j} = \frac{(b_i | \tilde{b}_j)}{\|\tilde{b}_j\|^2}$, a size-reduced basis is such that the projection of b_i onto $\langle b_1, \dots, b_{i-1}\rangle_{\mathbb{R}}$ is in the domain

$$\left[\frac{1}{2}, \frac{1}{2}\right]^{i-1} \times \left(\tilde{b}_j\right)_{j \in [\![1, i-1]\!]}$$

One can find the simple algorithm computing a size-reduced basis in Algorithm 2, which is essentially an approximation of the GSO algorithm (Alg. 1).

Algorithm 2 SizeReduce

```
Require: A free family $\mathcal{B} = \{b_1, .., b_k\}$ of $\mathcal{L}$ such that $|\mu_{i,j}| \leqslant 1/2$ for all $i \neq j$ , the
    matrix $G$ containing all $\mu_{i,j}$, an element $b \notin \mathcal{L}$.
Ensure: An element $b_{k+1}$ such that $|\mu'_{k+1,i}| \leqslant 1/2$ for $i \neq k+1$
1: $b_{k+1} \leftarrow b$
2: $\tilde{\mathcal{B}}, G \leftarrow \texttt{GSO}(\mathcal{B})$
3: for $i = k$ to 1 do
4:     $b_{k+1} \leftarrow b_{k+1} - \left\lfloor \frac{(b_{k+1} \mid \tilde{b}_i)}{\|\tilde{b}_i\|^2} \right\rceil b_j$
5:     Update $G$
6: end for
7: return $\mathcal{B} \cup \{b_{k+1}\}$, $G$
```

Theorem 2.4 (Complexity of size reduction)**.** *Consider $\mathcal{B} = (b_1, \dots, b_r) \subseteq \mathbb{R}^n$ and $b \in \mathbb{R}^n$, and denote by $M = \max\{\log_2 \|x\|_2 \mid x \in \mathcal{B} \cup \{b\}\}$. Then one can compute the output of Algorithm 2 in bit-complexity*

$$O\left(rn\mathcal{M}(rM)\right). \tag{2.4}$$

Remark 5. When considering a full-rank lattice, i.e. $r = n$, and a classical complexity for the multiplication – for example $\mathcal{M}(n) = \widetilde{O}(n)$ – Equation (2.4) becomes $\widetilde{O}(n^3 M)$.

A first reduction algorithm: LLL [63]

A fairly natural way of reducing a basis would be to follow Algorithm 1, and replace all coefficients $\mu_{i,j}$ by their closest integers. This way one could obtain a basis close to the GSO of the starting matrix, i.e. which is size-reduced. In order to obtain a polynomial time algorithm outputting a basis which is proven to be reduced for varying dimension, one has to introduce a new reduction condition. This is the result of the ground breaking work by A. K. Lenstra, H. W. Lenstra and L. Lovász in [63]. They define a new notion of *reduced basis*, that is then called *LLL-reduced.*

Definition 2.19. Consider a lattice $\mathcal{L}$ defined by a basis $\mathcal{B} = (b_1, \dots, b_r)$ and $\delta \in]\frac{1}{4}, 1[$. Then $\mathcal{B}$ is called LLL-reduced with parameter δ (or δ-LLL reduced) if it satisfies the following conditions.

1. It is *size-reduced*: $\forall i \in [\![1, r]\!], \forall 1 \leqslant j < i, |\mu_{i,j}| \leqslant \frac{1}{2}$.
2. It satisfies the *Lovász conditions*:

$$\forall i \in [\![2, r]\!], \delta\|\tilde{b}_{i-1}\| \leqslant \|\tilde{b}_i + \mu_{i,i-1}\tilde{b}_{i-1}\|^2 = \|\tilde{b}_i\|^2 + \mu_{i,i-1}^2\|\tilde{b}_{i-1}\|^2.$$

Then the LLL algorithm shown in Algorithm 3 essentially consistsof applying `SizeReduce` to new vector basis incrementally, verify if Lovász condition is true and continue if so. Otherwise we swap the two last vectors and reduce again.

Algorithm 3 LLL

Require: $\mathcal{B}$, a basis of $\mathcal{L}$ of rank r, and a constant $\delta \in]\frac{1}{4}, 1[$.
Ensure: $\mathcal{B}'$, a $\delta-$LLL reduced basis of $\mathcal{L}$.

```
1: $i \leftarrow 2$
2: $\mathcal{B}' \leftarrow \{b_1\}$
3: while $i \leqslant r$ do
4:     $\mathcal{B}' \leftarrow (b'_1, \dots, b'_{i-1})$
5:     $\mathcal{B}', G \leftarrow$ SizeReduce($\mathcal{B}', b_i$)
6:     if Lovász condition is satisfied for $\delta, i$ then
7:         $i \leftarrow i + 1$
8:     else
9:         Swap($b'_i, b'_{i-1}$)
10:        $i \leftarrow \max\{2, i - 1\}$
11:    end if
12: end while
13: return $\mathcal{B}$
```

Remark that each time the family is modified, one needs to update its GSO. It is done in `SizeReduce` where the whole GSO is computed. However since at a given step i, only b_i and eventually b_{i-1} (because of the swap) are modified only a few coefficients need to be updated.

Theorem 2.5. *Consider $\mathcal{L}$ a lattice of rank r, $\mathcal{B} = (b_1, \dots, b_r)$ a δ-LLL reduced basis of $\mathcal{L}$ for $\delta = 3/4$, and $\tilde{\mathcal{B}}$ the GSO of $\mathcal{B}$. Then the following properties are true.*

1. $\det(\mathcal{L}) \leqslant \prod_{i=1}^{r} \|b_i\| \leqslant 2^{r(r-1)/4} \det(\mathcal{L})$
2. $\forall i \in [\![1, r]\!], \forall 1 \leqslant j \leqslant i, \|b_j\| \leqslant 2^{(i-1)/2} \|\tilde{b}_i\|$
3. $\|b_1\| \leqslant 2^{(r-1)/4} \sqrt[r]{\det(\mathcal{L})}$
4. *For any* $x \in \mathcal{L} \setminus \{0\}$, $\|b_1\| \leqslant 2^{(r-1)/2} \|x\|$.
5. *For any free family* $(x_1, \dots, x_k) \in \mathcal{L}^k$, $\forall j \in [\![1, k]\!]$, $\|b_j\| \leqslant 2^{(r-1)/2} \max\{\|x_i\| \mid i \in [\![1, k]\!]\}$.

Theorem 2.5 shows that a LLL reduced basis has good properties. It provides upper bounds related to the norms of the basis vectors. In particular, one can remark that the norm of shortest basis vectors cannot be too large compared to $\lambda_1(\mathcal{L})$. Indeed, the fourth point shows that the LLL algorithm solves in deterministic polynomial time SVP$_\gamma$, for $\gamma = 2^{(r-1)/2}$. Many improvements and versions were developed since the original version of LLL. Among many others, one can consider the use of floating-point arithmetic in order to hasten computations [73], the modification of the `Swap` operation called LLL with deep insertions [92, 42], or modifications which allow considering generating families which are not a basis of the lattice [84].

Theorem 2.6 (Complexity of LLL [74])**.** *Consider $\mathcal{L} = \mathcal{L}(\mathcal{B}) \subseteq \mathbb{R}^n$ a lattice of rank r, and $M = \max\{\log_2 \|b\|_2 \mid b \in \mathcal{B}\}$. Then for input $\mathcal{B}$, one can compute a LLL-reduced basis of $\mathcal{L}$ in time complexity*

$$O\left(r^2 n (r + M) M \mathcal{M}(r)\right). \tag{2.5}$$

Remark 6. When considering a full-rank lattice, i.e. $r = n$, and a classical complexity for the multiplication – for example $\mathcal{M}(n) = \widetilde{O}(n)$ – Equation (2.5) becomes $\widetilde{O}\left((n^5 + n^4 M) M\right)$.

Enumerating short vectors

In order to solve the exact SVP or CVP, one can use *enumeration techniques.* It was first suggested by Pohst [85] in 1981, and other versions were then developed by

Kannan or Fincke and Pohst [56, 41] for example. We will present how to enumerate all short vectors to solve the SVP. Then the ideas can be extended to solve the CVP.

In order to find a shortest vector in a lattice, one could go through all lattice vectors and recall which one is the shortest. Obviously, we need bound the space of vectors that we will enumerate. Let us explain how one can do this. First consider a lattice $\mathcal{L}$ given by a basis $\mathcal{B} = (b_1, \dots, b_r)$. We will again use the GSO of $\mathcal{B}$ in order to bound the norm of vectors we are interested in. Recall from Equation (2.1) one has

$$\forall i \in [\![1, r]\!], b_i = \tilde{b}_i + \sum_{j=1}^{i-1} \mu_{i,j}\tilde{b}_j.$$

Thus by inverting these relations, given $v = \sum_{i=1}^{r} v_i b_i \in \mathcal{L}$, one obtains

$$v = \sum_{i=1}^{r} \left(v_i + \sum_{j=i+1}^{r} v_j \mu_{j,i} \right) \tilde{b}_i.$$

Therefore, if we denote by $\tilde{v}_i$ the i-th coefficient of v expressed in $\tilde{\mathcal{B}}$, it is easy to see that one has

$$\forall k \in [\![1, r]\!], \|\pi_k(v)\|^2 = \sum_{i=k}^{r} |\tilde{v}_i|^2 \left\|\tilde{b}_i\right\|^2 = \sum_{i=k}^{r} \left(v_i + \sum_{j=i+1}^{r} v_j \mu_{j,i} \right)^2 \left\|\tilde{b}_i\right\|^2. \quad (2.6)$$

Now if we fix a bound R, we will be able to enumerate vectors v such that $\|v\| \leqslant R$ using Equation (2.6). Indeed one has

$$\|v\|^2 \leqslant R^2 \implies R^2 \geqslant \|\pi_1(v)\|^2 \geqslant \|\pi_2(v)\|^2 \geqslant \cdots \geqslant \|\pi_d(v)\|^2$$

which gives the equations

$$\forall k \in [\![1, r]\!], \sum_{i=k}^{r} \left(v_i + \sum_{j=i+1}^{r} v_j \mu_{j,i} \right)^2 \left\|\tilde{b}_i\right\|^2 \leqslant R^2. \quad (2.7)$$

Then the algorithm works as follows. Equation (2.7) with $k = r$ gives $v_r^2 \leqslant \frac{R^2}{\left\|\tilde{b}_r\right\|^2}$. The algorithm enumerates through all possible values for v_r. Then for a fixed value v_r one has

$$(v_{r-1} + v_r \mu_{r,r-1})^2 \left\|\tilde{b}_{r-1}\right\|^2 + v_r^2 \left\|\tilde{b}_r\right\|^2 \leqslant R^2$$

which gives an interval where v_{r-1} lies:

$$-\frac{\sqrt{R^2 - v_r^2 \left\|\tilde{b}_r\right\|^2}}{\left\|\tilde{b}_{r-1}\right\|} - v_r\mu_{r,r-1} \leqslant v_{r-1} \leqslant \frac{\sqrt{R^2 - v_r^2 \left\|\tilde{b}_r\right\|^2}}{\left\|\tilde{b}_{r-1}\right\|} - v_r\mu_{r,r-1}.$$

Thus the algorithm can enumerate v_{r-1} in this interval, and using all conditions in Equation (2.7) allows finding bounds for all coefficients of v which depend only on the norms of the GSO vectors, the Gram-Schmidt coefficients $\mu_{i,j}$ and the radius R. This process can also be seen as a search through a tree, where the nodes are vectors, levels correspond to the spaces $\pi_{r-i+1}(\mathcal{L})$ and the children of a node v at level i are the vectors in the next space $\pi_{r-i}(\mathcal{L})$ which are projected onto v when applying π_{r-i+1}. At each level the number of nodes are given by the bounds obtained by Equation (2.7).

Theorem 2.7. *Let $\mathcal{L}$ be a lattice given by a basis $\mathcal{B} = (b_1, \ldots, b_r)$, $R \in \mathbb{R}_+$ and $v \in \mathcal{L}$ such that $\|v\| \leqslant R$. Define also the following quantities,*

$$\forall i \in [\![1, r-1]\!], C_i := \frac{\sqrt{R^2 - \sum_{j=i+1}^{r} \left(v_j + \sum_{k=j+1}^{r} \mu_{k,j} v_k\right)^2 \left\|\tilde{b}_j\right\|^2}}{\left\|\tilde{b}_i\right\|}.$$

Then the coefficients of v satisfy the following inequalities,

$$\forall i \in [\![1, r-1]\!], -C_i - \sum_{j=i+1}^{r} v_j\mu_{j,i} \leqslant v_i \leqslant C_i - \sum_{j=i+1}^{r} v_j\mu_{j,i}.$$

The enumeration technique can be found in Algorithm 4.

Algorithm 4 The Enumeration Algorithm for SVP

```
Require: $\mathcal{B} = (b_1, \ldots, b_r)$ a basis of a lattice $\mathcal{L}$
Ensure: $v = \sum_{i=1}^{r} v_i b_i$ such that $v = \lambda_1(\mathcal{L})$
1: while There is still unexplored nodes do
2:     if current node has $\|(0, ..., 0, v_i, .., v_n)\| < R$ then
3:         Go down in the tree, explore all possibilities for $v_{i-1}$
4:     else
5:         Remove the node, go up a level, search another node
6:     end if
7: end while
8: return $B$
```

The complexity of an enumeration algorithm like Algorithm 4 is exponential. To achieve better running time, one usually reduces the basis first, by LLL for example. Indeed one obtains a bound R on $\lambda_1(\mathcal{L})$ by considering the norm of the shortest vector of the basis.

In order to speed-up the computations, one can use a modification called *pruning*, first suggested by Schnorr and Hörner [93]. It consistsof excluding parts of the tree where the probability of finding the shortest vector is very low. Thus enumerating over the remaining nodes gives a heuristic algorithm which is not guaranteed to find a shortest vector. However the vector returned will still be reasonably short.

HKZ reduction

As we saw, a LLL-reduced basis can be obtained in polynomial time and have relatively good properties. However, since the approximation factor obtained with such basis is exponential in the rank, one can consider that LLL is somehow a *weak reduction.* In order to obtained strongly reduced bases, one needs a better condition than Lovász condition.

Definition 2.20 (Hermite-Korkine-Zolotarev reduction [60])**.** Consider $\mathcal{B}$ a basis of a lattice of rank r. Then $\mathcal{B}$ is *Hermite-Korkine-Zolotarev reduced* or *HKZ reduced* if it satisfies the following conditions.

1. It is size-reduced.
2. For all $i \in [\![1, r]\!]$, $\left\|\tilde{b}_i\right\| = \lambda_1(\pi_i(\mathcal{L}))$.

As LLL reduced bases, HKZ reduced bases enjoy nice geometrical properties. In particular they allow good approximations of the successive minima to be obtained.

Theorem 2.8. *Let $\mathcal{B} = (b_1, \ldots, b_r)$ a HKZ reduced basis of a lattice $\mathcal{L}$. One has*

$$\forall i \in [\![1, r]\!], \frac{4}{i+3} \leqslant \left(\frac{\|b_i\|}{\lambda_i(\mathcal{L})}\right)^2 \leqslant \frac{i+3}{4}.$$

Remark 7. HKZ-reduced bases are LLL-reduced.

Using an oracle $\mathcal{O}$ which solves SVP, we can obtain Algorithm 5 – called the KZ algorithm – which computes a HKZ-reduced basis of a lattice.

Since Algorithm 5 calls an oracle to solve SVP instances, its asymptotic complexity is exponential. Thus it cannot be used for high dimensions. However it can be called in other processes to reduce blocks of a given basis, allowing the algorithms to achieve acceptable trade-off between complexity and quality of the reduction. Finally Algorithm 5 is a simplification of proper algorithms computing HKZ-reduced bases. Indeed we expressed $\mathcal{O}$ as a black-box, and call it on the lattice to obtain a shortest vector. A better way (and more complex to describe) is as Kannan describes it [57]. It is an algorithm which projects $(b_2, \ldots, b_n)$ onto $\pi_2(\mathcal{L})$, LLL reduces

Algorithm 5 KZ reduction algorithm

```
Require: $\mathcal{B} = \{b_1, ...b_r\}$ a basis of $\mathcal{L}$ of rank $r$
Ensure: $\mathcal{B}'$ a HKZ-reduced basis of $\mathcal{L}$
1: $b'_1 \leftarrow \mathcal{O}(\mathcal{B})$
2: $\mathcal{B}' \leftarrow \{b'_1\}$
3: for $i = 2$ to $r$ do
4:     Extend $\mathcal{B}'$ into a basis $\mathcal{B}''$ of $\mathcal{L}$
5:     $b''_i \leftarrow \mathcal{O}(\pi_i(B''))$
6:     Lift $b''_i$ to $b'_i \in \mathcal{L}$
7:     $\mathcal{B}' \leftarrow \texttt{SizeReduce}(\mathcal{B}', b'_i)$
8: end for
9: return $\mathcal{B}'$
```

this lattice and recursively calls itself, with some enumeration process. This first version has been improved afterwards by Helfrich [50], and its complexity analysed by Hanrot and Stelhé [49] who give a worst-case complexity of $2^{O(d)}d^{\frac{d}{2}}$.

A reduction by blocks: BKZ

The Block-Korkine-Zolotarev (BKZ) algorithm, was first proposed by Schnorr and Euchner [94]. It uses the KZ reduction algorithm as a subroutine in blocks. Suppose the oracle for SVP runs up to dimension k, and the lattice has $r > k$, then the BKZ algorithm is described in 6. In general, one denotes by BKZ$_k$ to specify the block size used in the algorithm.

Algorithm 6 BKZ basis reduction algorithm

```
Require: $\mathcal{B} = \{b_1, ...b_r\}$ a basis of $\mathcal{L}$, and a SVP oracle $\mathcal{O}$ up to $k \leqslant r$
Ensure: $\mathcal{B}'$ a reduced basis of $\mathcal{L}$ such that $\|\tilde{b}_i\| = \lambda_1(\pi_i(\mathcal{L}))$
1: while Changes occur do
2:     for $i = 1$ to $r - k + 1$ do
3:         HKZ reduce the block $\pi_i(\{b_i, ..., b_{i+k-1}\})$ then lift it in $\mathcal{B}$
4:         Use LLL on $B = \{b_1, ..., b_r\}$
5:     end for
6: end while
7: return $B$
```

The fact that the basis is locally HKZ reduced allows for good trade-offs between running time and the quality of the basis obtained, which increases with the block size used.

Theorem 2.9. *Consider $\mathcal{B}$ a basis of a lattice $\mathcal{L}$, and a SVP oracle $\mathcal{O}$ up to dimension $k \leqslant r$. Then for input $\mathcal{B}$ and $\mathcal{O}$, Algorithm 6 outputs a basis $(b_1, \ldots, b_r)$ satisfying the following properties:*

1. $\frac{\|b_1\|}{\lambda_1(\mathcal{L})} \leqslant (k^{\frac{1+\ln k}{2k-2}})^{r-1}$;
2. $\frac{\|b_1\|}{\sqrt[r]{\mathrm{vol}(\mathcal{L})}} \leqslant \sqrt{\gamma_k}(k^{\frac{1+\ln k}{2k-2}})^{d-1}$.

Despite the quality of the basis obtained by BKZ, one does not know a polynomial bound for the running time of Algorithm 6. Like LLL algorithm, BKZ behaves better in practice than theoretically. Again, several improvements of BKZ have been developed, mostly concerning the SVP oracle. For example, Chen and Nguyen [28] used improved enumerations to obtain faster computations.

Solving the CVP and BDD

We will present some techniques used to solve the BDD, as it is a problem of particular interest in our research. There are three main processes used, in a variety of applications and domains. Two are due to L. Babaï and the third has been developed by R. Kannan.

Babaï's rounding technique: The first method due to Babaï is simple to describe and implement. Given a lattice $\mathcal{L}$ of $\mathbb{R}^n$ given by a basis $\mathcal{B} = (b_1, \ldots, b_r)$ and a target vector $t \in \mathbb{R}^n$, it consistsof rounding to the nearest integer the coefficients of t in the $\mathcal{B}$.

Algorithm 7 Babai's Rounding Off Algorithm - `BabaiRounding`

```
Require: $\mathcal{B} = (b_1, \ldots, b_r)$ a basis of $\mathcal{L}$ and $t \in \langle b_1, \ldots, b_r \rangle_{\mathbb{R}}$
Ensure: $v \in \mathcal{L}$ a vector close to $t$
1: Compute $(t_1, \ldots, t_r)$ such that $t = \sum_{i=1}^r t_i b_i$
2: return $v = \sum_{i=1}^r \lfloor t_i \rceil b_i$.
```

Remark 8. If $\mathcal{L} = \mathcal{L}(B)$ is a full-rank lattice of $\mathbb{R}^n$ then B is an invertible square matrix, and Step 1 of Algorithm 7 can be expressed as tB^{-1}. Thus it is common to write the output of the algorithm as $\lfloor tB^{-1} \rceil B$.

One can apply Babaï's rounding using any basis of the lattice. However, as often with lattice problems and related algorithms, the quality of the solution will depend on the quality of the basis given as input.

Theorem 2.10 ([5])**.** *Consider a rank r lattice $\mathcal{L}$, given by a $\delta -$ reduced basis $\mathcal{B}$, with $\delta = 3/4$. Then for input t and $\mathcal{B}$, Algorithm 7 outputs $v \in \mathcal{L}$ such that*

$$\forall x \in \mathcal{L}, \|t - v\| \leqslant \left(1 + 2r\left(\frac{9}{2}\right)^{r/2}\right)\|t - u\|.$$

Alternatively, Algorithm 7 solves CVP$_\gamma$ for $\gamma = 1 + 2r\left(\frac{9}{2}\right)^{r/2}$.

One can remark that the output of Algorithm 7 belongs to the fundamental parallelepiped defined by the basis $\mathcal{B}$ containing t. Thus the quality of the decoding depends on the quality of the basis defining this parallelepiped, as shown in Proposition 2.2.

Proposition 2.2. *Consider $\mathcal{B}$ a basis of a lattice $\mathcal{L}$, and $\mathcal{F}(\mathcal{B})$ the fundamental parallelepiped defined by $\mathcal{B}$. Then for input $\mathcal{B}$, Babaï's rounding algorithm solves*

1. *CVP$_\gamma$ with γ being the radius of the smallest enclosing sphere of $\mathcal{F}(\mathcal{B})$;*
2. *BDD$_\gamma$ with γ being the radius of the biggest enclosed sphere of $\mathcal{F}(\mathcal{B})$.*

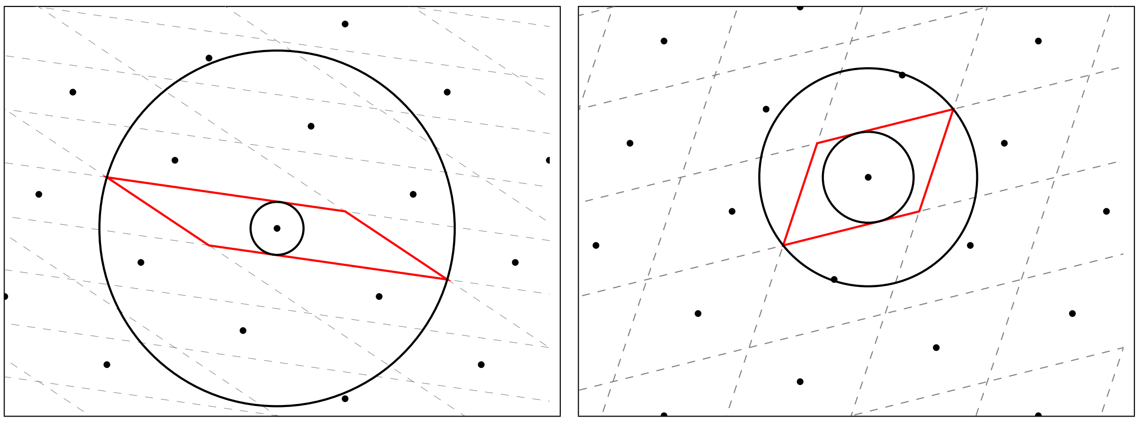
The radii of the largest enclosed and smallest enclosing spheres of a parallelepiped can be expressed with the help of the defining basis. Let us denote by $R_1(\mathcal{B})$ and $R_2(\mathcal{B})$ the mentioned radii. Then one has

$$R_1(\mathcal{B}) = \max\{\|\frac{1}{2}\sum_{i=1}^{r} x_i b_i\| \mid (x_i)_{i\in[\![1,r]\!]} \in \{-1,1\}^r\}$$

and

$$R_2(\mathcal{B}) = \min\{\frac{1}{2}\,\|b_i^\vee\| \mid i \in [\![1,r]\!]\}.$$

In particular, a target $t = v + e$ with $v \in \mathcal{L}$ will be reduced to v if $(e \mid b_i^\vee) \leqslant \frac{1}{2}$ for all $i \in [\![1,r]\!]$.



(a) A bad basis **(b)** A better basis

Figure 2.5: Rounding situation with two bases of the same lattice

Babaï's nearest plane algorithm: The second algorithm is also due to Babaï [5], and is called the *nearest plane algorithm.* It outputs similar although different results from the ones output by Algorithm 7. It is an inductive technique, described in Algorithm 8. As one can see, it is essentially a process of size-reduction (Alg. 2)

Again the output will depend on the quality of the basis given as input. We can prove that the output is not too far from the target given the basis is LLL-reduced.

Algorithm 8 Babaï's Nearest Plane Algorithm

```
Require: $t \in \mathbb{R}^n$, $\mathcal{B} = (b_1, \ldots, b_r)$ a basis of a lattice $\mathcal{L}$, $\tilde{\mathcal{B}}$ the GSO of $B$
Ensure: $v \in \mathcal{L}$ a close vector of $t$
1: $v \leftarrow t$
2: for $i = r$ down to 1 do
3:     $v \leftarrow v - \left\lfloor \frac{(v|\tilde{b}_i)}{\|\tilde{b}_i\|^2} \right\rceil b_i$        ▷ Make $t$ more orthogonal to $b_i$
4: end for
5: return $v - t$
```

Theorem 2.11. *Consider a rank r lattice $\mathcal{L}$, given by a $\delta -$ reduced basis $\mathcal{B}$, with $\delta = 1/4 + 1/\sqrt{2}$. Then for input t and $\mathcal{B}$, Algorithm 8 outputs $v \in \mathcal{L}$ such that*

$$\forall x \in \mathcal{L}, \|t - v\| \leqslant \frac{2^{r/4}}{\sqrt{\sqrt{2} - 1}} \|t - x\| < 1.6 \times 2^{r/4} \|t - x\| .$$

In other words, Algorithm 8 solves CVP$_\gamma$ for $\gamma = \frac{2^{r/4}}{\sqrt{\sqrt{2}-1}}$.

The output of Babaï's nearest plane has better quality than the one of the rounding technique. In fact one can prove it lies in a parallelepiped defined by the GSO of the basis given as input and centered on the target t.

Proposition 2.3. *Consider a rank r lattice $\mathcal{L}$, given by a basis $\mathcal{B}$. Then for input t and $\mathcal{B}$, Algorithm 8 outputs $v \in \mathcal{L}$ lying in $t + \mathcal{F}(\tilde{\mathcal{B}})$. Thus it solves BDD$_\gamma$ for $\gamma \leqslant \frac{1}{2} \min\{\left\|\tilde{b}_i\right\| \mid i \in [\![1, r]\!]\}$.*

Kannan embedding technique: The last process that we will mention is due to Kannan [57]. It transforms a CVP instance over a lattice into a SVP instance over an upper-lattice. Let us consider $\mathcal{B} = (b_1, \ldots, b_r)$ a basis of a lattice $\mathcal{L} \subset \mathbb{R}^n$ and t a vector in span$(\mathcal{L})$. Now denote by v a vector of $\mathcal{L}$ such that $\|t - v\| = d(t, \mathcal{L})$. Then one can remark that $e = t - v$ is short. The embedding technique consistsof building a lattice containing e. For this let us fix a constant $M \in \mathbb{R}_+$ and consider the matrix

$$\begin{bmatrix} b_1 & 0 \\ \vdots & \vdots \\ b_n & 0 \\ t & M \end{bmatrix}.$$

It defines a lattice $\mathcal{L}'$ of $\mathbb{R}^{n+1}$ which is an upper-lattice of $\mathcal{L}$, or more precisely of its embedding in $\mathbb{R}^{n+1}$ under the map $(x_1, \ldots, x_n) \in \mathbb{R}^n \hookrightarrow (x_1, \ldots, x_n, 0) \in \mathbb{R}^{n+1}$. Moreover the error vector $(t, M) - (v, 0) = (e, M)$ is in $\mathcal{L}'$. Therefore, depending on its size compared to the shortest vectors of $\mathcal{L}$, solving the SVP on $\mathcal{L}'$ might allow us to retrieve this vector and find v. The technique is summed up in Algorithm 9.

Algorithm 9 Kannan's embedding technique

```
Require: $B = (b_1, \dots, b_r)$ a basis of a lattice $\mathcal{L} \subset \mathbb{R}^n$, $t \in \mathrm{span}(\mathcal{L})$ and $M \in \mathbb{R}_+$
Ensure: $v \in \mathcal{L}$ a close vector of $t$
1: $\mathcal{B}' \leftarrow (b_i + 0 \times e_{n+1})_{i \in [\![1,r]\!]}$
2: $\mathcal{B}' \leftarrow \mathcal{B} \cup (t + Me_{n+1})$
3: $e' \leftarrow \mathrm{SVP}(\mathcal{B}')$                    ▷ SVP solver on $\mathcal{L}(\mathcal{B}')$
4: $e \leftarrow e' - Me_{n+1}$
5: return $t - e$
```

Theorem 2.12. *Consider a lattice $\mathcal{L}$ of $\mathbb{R}^n$ generated by a basis $\mathcal{B} = (b_1, \dots, b_r)$, $t \in \langle b_1, \dots, b_r \rangle_{\mathbb{R}}$, v a vector of $\mathcal{L}$ such that $d(t, \mathcal{L}) = \|t - v\|$, and $M \in \mathbb{R}_+$. Moreover let $\mathcal{L}'$ be the lattice of $\mathbb{R}^{n+1}$ generated by $\{(b, 0) \mid b \in \mathcal{B}\} \cup \{(t, M)\}$. Then the following holds*

$$\left(\|t - v\| < \frac{\lambda_1(\mathcal{L})}{2}\right) \wedge (M = \|t - v\|) \implies \|(t - v, M)\| = \lambda_1(\mathcal{L}')$$

and Algorithm 9 outputs v for input $\mathcal{B}, t$ and M.

Remark that Algorithm 9 uses a SVP solver and that the value of M in Theorem 2.12 is the norm of the error vector. In practice and for large dimensions, one can only solve SVP_γ for some γ potentially exponential in the rank of the lattice, by using an algorithm like LLL. Moreover the norm of the error vector $t - v$ is not always smaller than $\lambda_1(\mathcal{L})/2$. In these cases, one cannot certify that Kannan's embedding technique solves the CVP. Thus one has a heuristic method depending on the parameters in input.

Remark 9. Since the idea described is to reduce CVP to an instance of SVP, one usually fixes M to be small. However one can twist it as follows. Fix $M = \max\{\|b\|, b \in \mathcal{B}\}$ and use LLL instead of a SVP solver. Then the output of this modified Algorithm 9 is the same as Babaï's nearest plane algorithm.

2.3 Number theory

We refer the reader to [8, 30, 31, 72, 90] for anything related to number fields and computational number theory.

2.3.1 Number fields

Definition 2.21. A *number field* K is a field which is a finite extension of $\mathbb{Q}$, i.e. a finite dimensional $\mathbb{Q}$-vector space.

Notation. Given an extension L/K of number fields, we will call the dimension of L over K the *degree* of L/K. It will denoted by $[L : K]$.

Proposition 2.4. *Let L/K be an extension of number fields. Then there is an* irreducible polynomial $P(X) \in K[X]$ *such that*

$$L \cong \frac{K[X]}{(P(X))}.$$

and $[L:K] = \deg P(X)$. Moreover $P(X)$ has $[L:K]$ distinct roots in an algebraic closure $\overline{K}$ of K containing L. These roots define $[L:K]$ distinct $K-$isomorphisms of L into $\overline{K}$. If α is such a root, then the corresponding isomorphism σ_α is the following,

$$\begin{array}{cccc} \sigma_\alpha : & \dfrac{K[X]}{(P(X))} & \longrightarrow & K[\alpha] \subset \overline{K} \\ & \displaystyle\sum_{i=0}^{[L:K]-1} c_i X^i & \longmapsto & \displaystyle\sum_{i=0}^{[L:K]-1} c_i \alpha^i. \end{array}$$

Remark 10. As a matter of fact Proposition 2.4 is true for any finite field extension (in characteristic 0). If we consider number fields, their algebraic closure is the set of algebraic numbers $\overline{\mathbb{Q}} \subset \mathbb{C}$.

Notation. Given a number field extension L/K we will denote by Ω one of its algebraic closure. Then $\operatorname{Hom}(L/K, \Omega)$ will be the set of the $[L:K]$ distinct K-isomorphisms of L into Ω. The same way, one can denote by $\operatorname{Hom}(L, \Omega)$ the set of field embeddings of L into Ω. Similarly we will denote by $\operatorname{Hom}(L, \mathbb{C})$ the set of $[L:\mathbb{Q}]$ field embeddings of L into $\mathbb{C}$. One can then define the set $\operatorname{Hom}(L/K, \mathbb{C})$ to be the set of K-linear field embeddings of L into $\mathbb{C}$. Be aware that one needs to specify an embedding of K into $\mathbb{C}$ for this to be properly defined. It is usual for the two approaches of field embeddings described above – algebraic or complex – to be identified, as it the case in [30] for example. We will do the same, and the context will help determine which objects are considered. We will therefore mainly talk about "complex embeddings" and use the notations $\operatorname{Hom}(L/K, \mathbb{C})$ and $\operatorname{Hom}(L, \mathbb{C})$, even when considering morphisms from a number field into an algebraic closure.

Remark 11. Given any object f for which it makes sense, the result of the action of $\sigma \in \operatorname{Hom}(L/K, \mathbb{C})$ on f will be called a *conjugate* of f (relative to L/K).

Lemma 2.2. *Consider L/K an extension of number fields and $S \subseteq \operatorname{Hom}(L/K, \mathbb{C})$. Then the set $\{x \in L \mid \forall \sigma \in S, x^\sigma = x\}$ is a subextension of L/K.*

Notation. Given an extension of number fields L/K and $S \subseteq \operatorname{Hom}(L/K, \mathbb{C})$ we will denote by $\operatorname{Inv}(S)$ or L^S the number field fixed by S.

Let us consider a few examples of number fields.

1. First let us fix $P(X) = X^2 - 2$. Then $P(X)$ is irreducible over $\mathbb{Q}$ and $K = \mathbb{Q}[X]/(P(X))$ is a number field of degree 2. It can also be seen as $\mathbb{Q}(\sqrt{2})$, the smallest field in $\overline{Q}$ containing $\mathbb{Q}$ and $\sqrt{2}$.

2. One important type of number fields are the *cyclotomic fields.* They are widely studied and are the most used in cryptography. They are generated by a fundamental root of unity ζ_m, with m being called the *conductor* of the field. Such a field is then of the form $\mathbb{Q}(\zeta_m)$ and its degree is $\phi(m)$, where ϕ is Euler's totient function.

3. Consider $K = \mathbb{Q}(\zeta_m)$ a cyclotomic field with $m = 2^n$ for some $n \geqslant 1$, $P(X) = X^2 - \zeta_m \in K[X]$ and $L = K[X]/(P(X))$. Then a root of $P(X)$ in $\mathbb{C}$ is $\zeta_{2^{n+1}}$ so L is isomorphic to the cyclotomic field $\mathbb{Q}(\zeta_{2m})$, and $[L : K] = 2$.

Given several number fields, one can construct a number field containing all of them.

Definition 2.22. Consider K_1 and K_2 two number fields. The *compositum* of K_1 and K_2 is the smallest number field containing $K_1 \cup K_2$. It is denoted by K_1K_2.

Remark 12. In general $[K_1K_2 : \mathbb{Q}] \neq [K_1 : \mathbb{Q}][K_2 : \mathbb{Q}]$, one can only say $[K_1K_2 : \mathbb{Q}] \leqslant [K_1 : \mathbb{Q}][K_2 : \mathbb{Q}]$. For example if $K_1 = \mathbb{Q}(\sqrt{2}, \sqrt{3})$ and $K_2 = \mathbb{Q}(\sqrt{2}, \sqrt{5})$ then K_1K_2 is equal to $K_1 = \mathbb{Q}(\sqrt{2}, \sqrt{3}, \sqrt{5})$. Therefore $[K_1 : \mathbb{Q}][K_2 : \mathbb{Q}] = 16$ is different from $[K_1K_2 : \mathbb{Q}] = 8$.

2.3.2 Galois extensions

Definition 2.23 (Galois group)**.** Consider a field extension L/K. Then the *Galois group of* L/K, denoted by $\mathrm{Gal}(L/K)$, is the group of field automorphisms of L which are congruent to the identity when restricted to K, i.e.

$$\mathrm{Gal}(L/K) = \{\sigma \in \mathrm{Aut}(L) \mid \sigma \equiv \mathrm{Id}_K\}.$$

The Galois group $\mathrm{Gal}(L/K)$ of an extension can be seen as a subset of $\mathrm{Hom}(K, \mathbb{C})$. It has important properties, especially when the extension itself is Galois.

Definition 2.24 (Galois extension)**.** An extension of number fields L/K is called a *Galois extension* when $|\mathrm{Gal}(L/K)| = [L : K]$. If $K = \mathbb{Q}$ then we say that L is a *Galois field*, or more simply that L is Galois.

For example the cyclotomic fields are Galois number fields as well as the multiquadratic fields considered in [6]. However this property is not satisfied by a general number field K and we have to consider the Galois closure of K which is in fact the smallest extension containing all the roots of the irreducible polynomial $P(X)$.

Proposition 2.5 ([90]). *Consider a Galois extension L/K. Then there is a bijection between the set of subextensions of L/K and the subgroups of* $\mathrm{Gal}(L/K)$. *It is realised by the two following maps:*

$$\Phi : M \longmapsto \mathrm{Gal}(L/M),$$

and

$$\Psi : H < G \longmapsto L^H.$$

When an extension is not Galois, we might have to consider its Galois closure.

Definition 2.25 (Galois closure).

1. Consider L/K an extension of number fields. We call the *Galois closure of* L/K and denote by $\widetilde{L}$ the smallest number field M containing L such that M/K is Galois.
2. Given a number field L, *the Galois closure of* L is the Galois closure of the extension $L/\mathbb{Q}$.

2.3.3 Traces and norms

Definition 2.26. Let L/K be an extension of number fields, and $x \in L$. Then one defines the *trace* (resp. the *norm*) of x *relative to* L/K to be the trace (resp. the determinant) of the $K-$linear map $[x] : L \to L$. The trace (resp. norm) relative to $L/\mathbb{Q}$ will be called the absolute trace (resp. norm) of x.

Notation. The relative trace (resp. norm) of an extension L/K is denoted $\mathrm{Tr}_{L/K}$ (resp. $\mathrm{N}_{L/K}$). The absolute trace (resp. norm) of L is then written Tr (resp. N) when there is no ambiguity.

One can alternatively describe the trace and norm of an element in terms of its conjugates, i.e. the elements $\sigma(x)$ for $\sigma \in \mathrm{Hom}(L/K, \mathbb{C})$

Proposition 2.6 ([90]). *Consider a number field extension L/K of degree n, write* $\mathrm{Hom}(L/K, \mathbb{C}) = \{\sigma_1, \ldots, \sigma_n\}$ *and fix $x \in L/K$. Then the following are true,*

$$\mathrm{Tr}_{L/K}(x) = \sum_{i=1}^{n} \sigma_i(x), \qquad \mathrm{N}_{L/K}(x) = \prod_{i=1}^{n} \sigma_i(x).$$

Following their definition, one can easily deduce some properties of $\mathrm{Tr}_{L/K}$ and $\mathrm{N}_{L/K}$. The map $\mathrm{Tr}_{L/K}$ is $K-$linear and $\mathrm{N}_{L/K}$ is multiplicative. Moreover if $a \in K$ then $\mathrm{N}_{L/K} = a^{[L:K]}$.

An important quantity in number fields is the discriminant. First let us define the discriminant of a family. We will define the discriminant of an extension L/K later on.

Definition 2.27 (Discriminant of a family)**.** Consider an extension of number fields L/K of degree n, and $(x_1, \ldots, x_n) \in L^n$. Then the *discriminant of* $(x_1, \ldots, x_n)$ *(relative to L/K)*, denoted by $D_K(x_1, \ldots, x_n)$ is the element $\det \left[\mathrm{Tr}_{L/K}(x_i x_j)\right]_{\substack{i \in [\![1,n]\!] \\ j \in [\![1,n]\!]}}$.

Proposition 2.7 ([90])**.** *Consider an extension of number fields L/K of degree n, and write $\sigma_1, \ldots, \sigma_n$ the elements of* $\mathrm{Hom}(L/K, \mathbb{C})$*. For any $(x_1, \ldots, x_n) \in L^n$ one has*

$$D_K(x_1, \ldots, x_n) = \det \left[\sigma_i(x_j)\right]_{\substack{i \in [\![1,n]\!] \\ j \in [\![1,n]\!]}}.$$

Moreover if $(x_1, \ldots, x_n)$ is a basis of L over K then $D_K(x_1, \ldots, x_n) \neq 0$.

2.3.4 Orders

Now let us describe important ring structures attached to number fields.

Definition 2.28. An *order* of a number field K is a subring of K which is a finitely generated $\mathbb{Z}$-module and of maximal rank $[K : \mathbb{Q}]$.

We can typically consider orders generated by $\mathbb{Q}$-basis of K. In particular, if α is a root of an irreducible polynomial $P(X)$ defining a number field K, then $\mathbb{Z}[\alpha]$ is an order of K. It is generated by the successive powers of α and is isomorphic to the quotient ring

$$\frac{\mathbb{Z}[X]}{(P(X))}.$$

The most important order is the ring of integers of K, which generalises the notion of integers to algebraic numbers.

Definition 2.29 (Ring of integers)**.** Let K a number field. The *ring of integers* of K, denoted by $\mathcal{O}_K$ is the ring of *integral elements* of K, defined by

$$\mathcal{O}_K = \{x \in K \mid \exists P(X) \in \mathbb{Z}[X] \text{ monic}, P(x) = 0\}.$$

Proposition 2.8 ([31])**.** *The ring of integers of a number field K is a free $\mathbb{Z}$-module of rank $[K : \mathbb{Q}]$. Moreover any order of K is included in $\mathcal{O}_K$.*

Because of Proposition 2.8, the ring of integers $\mathcal{O}_K$ is also called *the maximal order* of K. For some number fields $K = \mathbb{Q}(\alpha)$, the maximal order $\mathcal{O}_K$ is isomorphic to $\mathbb{Z}[\alpha]$. It is the case for cyclotomic fields for example. However we stress that it is not true in general. For example, the maximal order of the quadratic field $\mathbb{Q}(\sqrt{5})$ is generated by $\{1, \frac{1+\sqrt{5}}{2}\}$.

Remark 13. Given an extension of number fields L/K, we know that $\mathcal{O}_L$ and $\mathcal{O}_K$ are both free over $\mathbb{Z}$, and that $L \cong K^{[L:K]}$. Then one could wonder whether $\mathcal{O}_L$ is free over $\mathcal{O}_K$, i.e. if $\mathcal{O}_L \cong \mathcal{O}_K^{[L:K]}$. While it is the case if $\mathcal{O}_K$ is a principal ring, it is not the case in general [90].

Orders can be useful to approximate the ring of integers when it is unknown, and too complicated to obtain.

Proposition 2.9 ([90]). *Consider an extension of number fields L/K of degree n. The following properties are true.*

1. *For any $x \in \mathcal{O}_L$, $\mathrm{Tr}_{L/K}(x)$ and $\mathrm{N}_{L/K}(x)$ are elements of $\mathcal{O}_K$.*
2. *For any family $(x_1, \ldots, x_n) \in \mathcal{O}_L^n$, the discriminant $D_K(x_1, \ldots, x_n)$ belongs to $\mathcal{O}_K$.*

Moreover the trace and norm maps are transitive, i.e. if $M/L/K$ is a tower of number fields then $\mathrm{Tr}_{M/K} = \mathrm{Tr}_{L/K}\mathrm{Tr}_{M/L}$ and $\mathrm{N}_{M/K} = \mathrm{N}_{L/K}\mathrm{N}_{M/L}$.

Definition 2.30 (Discriminants).

1. Let K be a number field. The *(absolute) discriminant* of an order $\mathcal{O}$ of K is the integer $D_K(b_1, \ldots, b_n)$, where $(b_1, \ldots, b_n)$ is any $\mathbb{Z}$-basis of $\mathcal{O}$. The *(absolute) discriminant* of K is the discriminant of its ring of integers $\mathcal{O}_K$.
2. Given an extension of number fields L/K, its *relative discriminant* is defined to be the (well-defined) ideal of $\mathcal{O}_K$ generated by the discriminants of all bases of L/K which are contained in $\mathcal{O}_L$. It is denoted by $\mathfrak{d}(L/K)$.

Notation. The absolute discriminant of an order will be denoted by $D_K(\mathcal{O})$. The discriminant of K is simply denoted by D_K or $D(K)$.

Remark 14. We need a different way of defining relative discriminants because the ring of integers $\mathcal{O}_L$ *is not* (in general) free over $\mathcal{O}_K$, i.e. there is no basis of $\mathcal{O}_L$ over $\mathcal{O}_K$ to consider.

The relation of inclusion for orders can be described by a simple property of their discriminants.

Proposition 2.10 ([31]). *Consider $\mathcal{O}_1$ and $\mathcal{O}_2$ two orders of a number field K. Then the following is true:*

$$\mathcal{O}_1 < \mathcal{O}_2 \iff \exists f \in \mathbb{Q}, D_K(O_1) = D_K(O_2)f^2.$$

2.3.5 Ideals

Now let us consider part of the arithmetic of number fields and number field extensions, by considering their ideals.

Definition 2.31. Let K be a number field. An *integral ideal* of $\mathcal{O}_K$ is simply an ideal of the ring $\mathcal{O}_K$. A *fractional ideal* of $\mathcal{O}_K$ is a $\mathbb{Z}$-submodule J of K such that there is $d \in \mathbb{Z}$ satisfying the fact that dJ is an ideal of the ring $\mathcal{O}_K$.

Remark 15. We will sometimes refer to *ideals of* K instead of ideals of $\mathcal{O}_K$.

Notation. The set of fractional ideals of a number field K is denoted by $\mathcal{I}(K)$.

The first important structural result on the set fractional ideals is that it has a group structure.

Theorem 2.13 ([90])**.** *Let K be a number field, and denote by $\mathcal{P}$ the set of prime integral ideals of K. Then the following propositions are true.*

1. *The set $\mathcal{I}(K)$ is an abelian group, where the law is the standard ideal product.*
2. *Every fractional ideal I can be uniquely expressed as a product of prime integral ideals*
$$I = \prod_{\mathfrak{p} \in \mathcal{P}} \mathfrak{p}^{v_{\mathfrak{p}}(I)}, \tag{2.8}$$
such that for all $\mathfrak{p} \in \mathcal{P}$, $v_{\mathfrak{p}}(I) \in \mathbb{Z}$, and for almost all $\mathfrak{p} \in \mathcal{P}$, $v_{\mathfrak{p}}(I) = 0$.

As it was the case for elements, one can define the trace and the norm of an ideal. First we can define the absolute norm.

Proposition 2.11 ([90])**.** *Let K be a number field and I an integral ideal of K. Then I is a submodule of $\mathcal{O}_K$ of maximal rank. Thus the quotient ring $\mathcal{O}_K/I$ is finite. Moreover the map $I \mapsto |\mathcal{O}_K/I|$ is multiplicative over the set of integral ideals.*

Definition 2.32 (Absolute norm of an ideal)**.** Let K be a number field. One defines the *(absolute) norm* of an ideal as follows:

1. The norm of an integral ideal I is the positive integer $|\mathcal{O}_K/I|$;
2. Given I and J two integral ideals, the norm of the fractional ideal I/J is the quotient of the norm of I by the norm of J.

The norm map is denoted by N_K, or N when there is no ambiguity.

An important object in number theory is the class group.

Definition 2.33 (Class group)**.** Consider K a number field. The *class group of* K, denoted by $\mathrm{Cl}(K)$ is the quotient group of fractional ideals by the subgroup of principal ideals (generated by a single element).

We can remark that since a prime ideal $\mathfrak{p}$ is in fact a maximal ideal of $\mathcal{O}_K$ then the quotient ring $\mathcal{O}_K/\mathfrak{p}$ is a finite field, so $\mathrm{N}_K(\mathfrak{p}) = p^f$ for some prime integer p and $f \in \mathbb{N}^*$. We will see below what is f, when looking into the splitting of ideals. There are several ways of defining the relative norm of an ideal. The most simple is certainly the following.

Definition 2.34 (Relative norm of ideals)**.** Let L/K be a number field extension. The *norm* of an ideal I of L *relative* to L/K, denoted by $\mathrm{N}_{L/K}(I)$, is the fractional ideal of K generated by the norms of elements of I relative to L/K. In mathematical terms, one has $\mathrm{N}_{L/K}(I) = \langle \mathrm{N}_{L/K}(x) \mid x \in I \rangle_{\mathcal{O}_K}$.

As it is the case for elements of L, the relative norm of an ideal I can be expressed using the action of $\mathrm{Hom}(L/K, \mathbb{C})$ onto I.

Proposition 2.12. *Consider L/K an extension of number fields, and I an ideal of L. Moreover denote by H the set of $K-$embeddings of L into $\mathbb{C}$. Then the norm of I relative to L/K satisfies the following equation, where the products are done over a suitable extension of L.*

$$\mathrm{N}_{L/K}(I) = \left(\prod_{\sigma \in H} \sigma(I) \right) \cap K. \tag{2.9}$$

As it was the case for elements the relative norm of ideals is transitive. The relative norm is also involved in a formula concerning relative discriminants.

Proposition 2.13. *Let $M/L/K$ be a tower of number fields, and I be an ideal of M. Then $\mathrm{N}_{M/K} = \mathrm{N}_{L/K}\mathrm{N}_{M/L}$, and $\mathfrak{d}(M/K) = \mathfrak{d}(L/K)^{[M:L]}\mathrm{N}_{L/K}(\mathfrak{d}(M/L))$.*

2.3.6 Representation of elements and structures

First let us describe the structure of orders and ideals.

Proposition 2.14 ([90])**.** *Given ideal I of a number field K, one can find a basis $(b_1, \ldots, b_n)$ of elements of $\mathcal{O}_K$ such that $K = \bigoplus_{i=1}^n \mathbb{Q}b_i$, $\mathcal{O}_K = \bigoplus_{i=1}^n \mathbb{Z}b_i$ and $I = \bigoplus_{i=1}^n \mathbb{Z}d_ib_i$ with $(d_1, \ldots, d_n) \in \mathbb{Z}^n$.*

Proposition 2.14 shows that the ring of integers as well as its integral ideals are full rank $\mathbb{Z}$-submodules of K. Therefore, images of $\mathcal{O}_K$ and of any ideal I of $\mathcal{O}_K$ under the action of any embedding of K into $\mathbb{R}^n$ are lattices. Then, in the representation given by said embedding, one can describe the volume of any ideal compared to the volume of the ring of integers.

Proposition 2.15. *Consider K a number field, I an ideal of K. Then* $\mathrm{vol}(I) = \mathrm{N}_K(I)\mathrm{vol}(\mathcal{O}_K)$.

Let us describe the two main ways of representing elements of a number field, leading to different geometrical situations.

The standard representation

The usual embedding corresponds to viewing a number field K as a quotient $\frac{\mathbb{Q}[X]}{(P(X))}$. Then every element $g(X) = g_0 + \cdots + g_{n-1}X^{n-1}$ of K can be seen as the vector with coordinates $(g_0, \ldots, g_{n-1})$ in $\mathbb{Q}^n$. This defines the coefficient embedding of K into $\mathbb{R}^n$.

Definition 2.35. Given a number field K defined by a degree n irreducible polynomial $P(X)$, the *coefficient embedding* or *polynomial embedding* is defined as

$$\begin{array}{rccc} \sigma_{\mathbf{coeff}}: & \dfrac{\mathbb{Q}[X]}{(P(X))} & \longrightarrow & \mathbb{Q}^n \hookrightarrow \mathbb{R}^n \\ & \displaystyle\sum_{i=0}^{[K:\mathbb{Q}]-1} c_i X^i & \longmapsto & (c_0, \ldots, c_{n-1}). \end{array} \tag{2.10}$$

In this standard representation, one can use the classical Euclidean norm l_2 of $\mathbb{R}^n$.

Remark 16. Since this embedding corresponds to the description of a number field as a quotient of a polynomial ring, we will forget about $\sigma_{\texttt{coeff}}$. Thus in general, the geometrical properties of elements or structures of a number field are considered under this morphism.

Remark 17. This embedding is not canonical. Indeed, it depends on the basis chosen for K. We described it with $(X^i \pmod{P(X)})_{i\in[\![0,n-1]\!]}$ but one could consider an integral basis of $\mathcal{O}_K$. Different bases give different geometries. For example, the embedding given by an integral basis sends $\mathcal{O}_K$ to $\mathbb{Z}^n$, thus the volume of I is $\mathrm{N}_K(I)$.

The canonical embedding

The other fundamental embedding is canonical, and uses the complex embeddings of K. First let us describe this set in more detail. As already mentioned, an element $\sigma_\alpha \in \mathrm{Hom}(K, \mathbb{C})$ is a field embedding of K into $\mathbb{C}$ corresponding a root α of $P(X)$ in $\mathbb{C}$. Let us denote by R the set of roots of $P(X)$. Then one can remark that R is globally invariant by the action of the complex conjugation.

Definition 2.36. The *signature* of a number field K defined by a polynomial $P(X)$ is the pair $(r_1, r_2) \in \mathbb{N}^2$ where:

- $r_1 = |\{\alpha \in R \mid \alpha \in \mathbb{R}\}|$;
- $2r_2 = |\{\alpha \in R \mid \alpha \in \mathbb{C} \setminus \mathbb{R}\}|$.

There are r_1 real embeddings and r_2 pairs of (strictly) complex embeddings. The two elements of a given pair are conjugates one from each other. It is usual to write $\sigma_1, \ldots, \sigma_{r_1}$ the real embeddings and to consider that $\sigma_{j+r_2} = \overline{\sigma_j}$ for all $j \in [\![r_1+1, r_1+r_2]\!]$.

Definition 2.37 (Minkowski's embedding)**.** Given a number field K defined by a degree n irreducible polynomial $P(X)$, the *canonical embedding* or *Minkowski's embedding* is defined as

$$\begin{array}{rccl} \sigma_{\mathbf{K}}: & K & \longrightarrow & \mathbb{R}^{r_1} \times \mathbb{C}^{r_2} \cong \mathbb{R}^n \\ & x & \longmapsto & (\sigma_i(x))_{i \in [\![1, r_1+r_2]\!]} \, . \end{array} \tag{2.11}$$

Then K can be seen as embedded in $\mathbb{R}^n$. More precisely it defines an isomorphism between $(K_{\mathbb{R}} = K \otimes_{\mathbb{Q}} \mathbb{R}, T_2)$ and $(\mathbb{R}^n, l_2)$, where the T_2 norm is defined as $T_2 : x \in K \mapsto \sum_{i=1}^{n} \sigma_i(x)\overline{\sigma_i(x)}$.

Proposition 2.16. *Let K be a number field defined by a degree n irreducible polynomial $P(X) \in \mathbb{Q}[X]$. Then under Minkowski's embedding,* $\mathrm{vol}(\mathcal{O}_K) = 2^{-r_2}\sqrt{|D_K|}$, *thus leading to* $\mathrm{vol}(I) = 2^{-r_2}\mathrm{N}_K(I)\sqrt{|D_K|}$.

Throughout this thesis, elements in number fields are essentially identified with their polynomial representation. Moreover, we will often consider σ_K to be defined as

$$\sigma_{\mathbf{K}} : x \longmapsto (\sigma_i(x))_{i \in [\![1,n]\!]},$$

where all complex embeddings are taken into account. Similarly for a field extension L/K we define $\sigma_{L/K} : x \longmapsto (\sigma(x))_{\sigma \in \mathrm{Hom}(K/L, \mathbb{C})}$. We call this map *Minkowski's embedding relative to L/K*. The maps σ_K and $\sigma_{L/K}$ will often be used for the fact that they establish linear bijections between the fields considered and a set of complex vectors.

2.3.7 Unit group and Log embedding

The group of units of $\mathcal{O}_K$ written $\mathcal{O}_K^\times$ is the set $\{u \in \mathcal{O}_K \mid u^{-1} \in \mathcal{O}_K\}$. It has a specific structure that we can take advantage of.

Proposition 2.17. *Given a number field K of degree n with $n = r_1 + 2r_2$ as before, we have*

$$O_K^\times \cong \frac{\mathbb{Z}}{m\mathbb{Z}} \times \mathbb{Z}^{r_1+r_2-1},$$

where m is the largest integer for which a primitive m-th root of unity belongs to K.

This isomorphism which allows seeing the units of $\mathcal{O}_K^\times$ modulo its torsion group as a lattice is realised by an important embedding which is the *Log-embedding* of K.

Definition 2.38. Let K be a number field of degree n, and (r_1, r_2) be its signature. Consider $(c_j)_{j\in[\![1,r1+r2]\!]}$ such that $c_j = 1$ if $j \leqslant r_1$ and $c_j = 2$ otherwise. Then the *Log-embedding* of K is defined as

$$\begin{array}{rccl} \mathrm{Log}_K : & K^* & \longrightarrow & \mathbb{R}^{r_1+r_2} \\ & x & \longmapsto & (c_i \ln|\sigma_i(x)|)_{i\in[\![1,r_1+r_2]\!]} \,. \end{array} \tag{2.12}$$

Theorem 2.14 ([72])**.** *Consider K a number field of degree n and signature (r_1, r_2). The set $\mathrm{Log}_K(\mathcal{O}_K^\times)$ is a lattice of the hyperplane orthogonal to the all ones vector. The volume of $\mathrm{Log}_K(\mathcal{O}_K^\times)$ is $\sqrt{r_1 + r_2}R_K$, where R_K is the* regulator *of K.*

Definition 2.39. Given a number field K, the lattice $\mathrm{Log}_K(\mathcal{O}_K^\times)$ is called the *Log-unit lattice* of K. We will denote by V_K its volume.

One can also define the Log-embedding by using all of the embeddings σ_i and forgetting the c_j. By doing so the Log-unit lattice is a lattice of rank $r_1 + r_2 - 1$ in $\mathbb{R}^n$ and its volume is $\sqrt{\frac{n}{2^{r_2/2}}}R_K$. In the rest of the thesis we will use this last form of the Log-embedding.

2.3.8 Splitting of an ideal in an extension

An important arithmetical phenomenon is the splitting of an ideal in an extension. More precisely, given I an ideal of K, then $J = I\mathcal{O}_L$ is an ideal of L. Following Theorem 2.13 J can be expressed as a product of prime ideals of L. We want to study this factorisation. Since an integral ideal can always be factored as a product of prime ideals, it is sufficient to consider prime ideals. First, given such an ideal $\mathfrak{p}$ of K, we can characterise the prime ideals of L being factors of $\mathfrak{p}$ in L.

Lemma 2.3 ([90])**.** *Consider L/K an extension of number fields, $\mathfrak{p}$ an ideal of K, and $\mathfrak{P}$ an ideal of L. Then $\mathfrak{P}$ divides $\mathfrak{p}$ in L if, and only if, $\mathfrak{P} \cap K = \mathfrak{p}$.*

Definition 2.40. Given an extension of number fields L/K, $\mathfrak{p}$ an ideal of K and $\mathfrak{P}$ an ideal of L, we say that $\mathfrak{P}$ is *above* $\mathfrak{p}$ if $\mathfrak{P} \mid \mathfrak{p}$.

Remark 18. Following Lemma 2.3 one can see that if $\mathfrak{P} \mid \mathfrak{p}$ then we get the field embedding $\frac{\mathcal{O}_K}{\mathfrak{p}} \hookrightarrow \frac{\mathcal{O}_L}{\mathfrak{P}}$. As they are both finite fields, one can remark it defines a finite field extension, and can consider the degree $[\frac{\mathcal{O}_L}{\mathfrak{P}} : \frac{\mathcal{O}_K}{\mathfrak{p}}]$.

Theorem 2.15 ([90])**.** *Consider L/K an extension of number fields, $\mathfrak{p}$ an ideal of K, and $\mathfrak{p} = \prod_{i=1}^{g} \mathfrak{P}_i^{v_{\mathfrak{P}_i}(\mathfrak{p})}$ its factorisation in L. Then the following propositions are true.*

1. $\sum_{i=1}^{g} v_{\mathfrak{P}_i}(\mathfrak{p})[\frac{\mathcal{O}_L}{\mathfrak{P}_i} : \frac{\mathcal{O}_K}{\mathfrak{p}}] = [L : K]$.

2. For all $i \in [\![1, g]\!]$, $\mathrm{N}_{L/K}(\mathfrak{P}_i) = \mathfrak{p}^{[\frac{\mathcal{O}_L}{\mathfrak{P}_i} : \frac{\mathcal{O}_K}{\mathfrak{p}}]}$.

Definition 2.41 (Residual degree and ramification index)**.** Consider L/K an extension of number fields, $\mathfrak{p}$ an ideal of K, and $\mathfrak{p} = \prod_{i=1}^{g} \mathfrak{P}_i^{v_{\mathfrak{P}_i}(\mathfrak{p})}$ its factorisation in L. Moreover fix $j \in [\![1, g]\!]$.

1. The *residual degree* or *inertial degree* of $\mathfrak{P}_j$ over $\mathfrak{p}$ is the index $[\frac{\mathcal{O}_L}{\mathfrak{P}_j} : \frac{\mathcal{O}_K}{\mathfrak{p}}]$. It is denoted by $f(\mathfrak{P}_j|\mathfrak{p})$.

2. The exponent $v_{\mathfrak{P}_j}(\mathfrak{p})$ is called the *ramification index* of $\mathfrak{P}_j$ over $\mathfrak{p}$, and is denoted by $e(\mathfrak{P}_j|\mathfrak{p})$.

One can rewrite the formulae in Theorem 2.15 as $[L : K] = \sum_{i=1}^{g} e(\mathfrak{P}_i|\mathfrak{p})f(\mathfrak{P}_i|\mathfrak{p})$ and $\mathrm{N}_{L/K}(\mathfrak{P}_i) = \mathfrak{p}^{f(\mathfrak{P}_i|\mathfrak{p})}$. Moreover the factorisation is even simpler if the extension is Galois, as shown by Proposition 2.18.

Proposition 2.18 ([90])**.** *Consider L/K an extension of number fields which is Galois, and $\mathfrak{p}$ a prime ideal of K. Then the maps $e(\cdot|\mathfrak{p})$ and $f(\cdot|\mathfrak{p})$ are constants over the primes of L dividing $\mathfrak{p}$. If e and f are the respective constant values and g the number of prime ideals of L above $\mathfrak{p}$, then $[L : K] = efg$.*

Definition 2.42 (Types of splitting)**.** Let L/K be an extension of number fields and $\mathfrak{p}$ be a prime ideal of $\mathcal{O}_K$. Let $\mathfrak{p} = \prod_{i=1}^{g} \mathfrak{P}_i^{e(\mathfrak{P}_i|\mathfrak{p})}$ be the factorisation of $\mathfrak{p}$ in L.

1. The ideal $\mathfrak{p}$ *ramifies* in L/K if there is $i \in [\![1, r]\!]$ with $e(\mathfrak{P}_i|\mathfrak{p}) > 1$.

2. If $g = 1$ and $f(\mathfrak{P}_1|\mathfrak{p}) = 1$, we say that $\mathfrak{p}$ *ramifies completely* in L.

3. We say that $\mathfrak{p}$ is *completely split* or *totally split* (or splits completely) in L if for all $i \in [\![1, g]\!]$, $e(\mathfrak{P}_i|\mathfrak{p}) = f(\mathfrak{P}_i|\mathfrak{p}) = 1$.

4. If $g = 1$ and $e(\mathfrak{P}_1|\mathfrak{p}) = 1$ then $\mathfrak{p}$ is said to be *inert* in L.

Now let us state how the discriminant ideal of an extension L/K is related to the splitting of prime ideals.

Theorem 2.16 ([30])**.** *Given L/K an extension of number fields, a prime ideal $\mathfrak{p}$ of K ramifies in L if, and only if, it divides the relative discriminant ideal $\mathfrak{d}(L/K)$.*

Another important object, related to the discriminant is the *different.*

Definition 2.43 (Different)**.** Consider an extension of number fields L/K. The *relative different* $\mathfrak{D}(L/K)$ is the ideal defined as follows,

$$\mathfrak{D}(L/K)^{-1} = \{x \in L \mid \mathrm{Tr}_{L/K}(x\mathcal{O}_L) \subset \mathcal{O}_K\}.$$

As it is the case for norm, the different is transitive.

Proposition 2.19. *Let $M/L/K$ be a tower of number field extensions. Then $\mathfrak{D}(M/K) = \mathfrak{D}(M/L)\mathfrak{D}(L/K)$.*

The different is useful because it can be used to compute the discriminant.

Proposition 2.20 ([30])**.** *Let L/K be an extension of number fields. Then the prime ideals of L dividing $\mathfrak{D}(L/K)$ are exactly the ones ramified in L/K. Moreover $\mathrm{N}_{L/K}(\mathfrak{D}(L/K)) = \mathfrak{d}(L/K)$.*

Proposition 2.21 ([87])**.** *Consider L/K an extension of number fields, $\mathfrak{p}$ an ideal of K, $\mathfrak{P}$ and ideal of L above $\mathfrak{p}$ and p the characteristic of $\mathcal{O}_K/(\mathfrak{p})$. Then if p and $e(\mathfrak{P} \mid \mathfrak{p})$ are coprime, one has $v_{\mathfrak{P}}(\mathfrak{D}(L/K)) = e(\mathfrak{P} \mid \mathfrak{p}) - 1$.*

2.4 Lattice based cryptography

In this Section we will describe the generic construction of encryption schemes using Euclidean lattices, then we will describe quickly some of the famous frameworks using (mainly structured) lattices, or which can been described with lattices. For a good survey on lattice based cryptography, we refer the reader to [77].

2.4.1 Generic construction of a scheme with lattices

General encryption scheme

The most basic encryption scheme, without added "features", is usually composed of three functions:

- **Setup()** outputs a pair of keys S_k, P_k. S_k is kept secretly and P_k is given publicly.
- **Encrypt(**m,P_k**)** outputs a ciphertext c given a public key P_k and a plaintext m.
- **Decrypt(**c,S_k**)** outputs a plaintext m given a ciphertext c.

Overall, the encryption scheme is deemed *correct* if the equality

$$\mathrm{Decrypt}(\mathrm{Encrypt}(m, P_k), S_k) = m \tag{2.13}$$

holds for any m from the message space, and any output (S_k,P_k) given by Setup(). Being correct however, does not mean that the scheme is secure. To ensure the security of the cryptographic schemes, we usually base them on computationally hard problems.

Generic scheme with lattices

We will now describe how a general encryption scheme can be designed with lattices.

- The public key is a "bad" basis H of a lattice; typically the HNF.
- The private key is a "good" basis, which is the trapdoor of the problem. In particular it needs to allowus to solve the problem the system is based on.
- A generic encryption can be done as follows. If m is the plaintext vector one can encrypt as follows:

  $$c = \text{Encrypt}(m, H) = mH + e$$

  where e is a short error vector, typically shorter than $\lambda_1(\mathcal{L}(H))/2$.
- In this configuration, recovering m can be done by solving the BDD. It is typically the case in the GGH encryption scheme [47]. This gives a decryption function:

  $$\text{Decrypt}(c, B) = \texttt{BDDsolver}(c, B)$$

Another option for encryption is

$$\text{Encrypt}(m, H) = sH + m = e + m$$

for some $s \in \mathbb{Z}^n$. The lattice vector is now e, and m is now short compared to e. Again, solving a BDD allows retrieving m.

$$\text{Decrypt}(c, B) = c - \texttt{BDDsolver}(c, B).$$

The BDD solver is typically `BabaiRounding` (Alg. 7). As we will see, for most systems which can be viewed as lattice constructions, the decryption phase correspond to solving a BDD.

Using a GDD solver

Now let us explain how the use of a BDD solver such as those given by Babaï's algorithms can be replaced by a GDD solver. First let us remind the reader of the following property.

Lemma 2.4 (Vector class unicity). *Let $\mathcal{L}$ be a lattice and $a, b \in \text{span}(\mathcal{L})$ such that $\|a\| + \|b\| < \lambda_1(\mathcal{L})$. Then*

$$a \equiv b \bmod \mathcal{L} \iff a = b.$$

Proof. The difference between a and b must be a vector of $\mathcal{L}$. The only vector of $\mathcal{L}$ that has a norm lower than $\lambda_1(\mathcal{L})$ is the null vector. □

Proposition 2.22. *Consider a lattice $\mathcal{L}$ and $(\gamma_m, \gamma_s) \in \mathbb{R}_+$ such that $\gamma_m + \gamma_s < 1$. Then a GDD$_{\gamma_s}$ solver is also a BDD$_{\gamma_m}$ solver.*

Proof. Assume that we have access to a GDD$_{\gamma_s}$ solver. This is equivalent to having a reduction algorithm `Reduce` which given v outputs w such that $w \equiv v \bmod \mathcal{L}$ and $\|w\| \leqslant \gamma_s \lambda_1(\mathcal{L})$. Now consider $v \in \operatorname{span}(\mathcal{L})$ such that $d(v, \mathcal{L}) < \gamma_m \lambda_1(\mathcal{L})$. Denote by w' the vector such that $v \equiv w' \bmod \mathcal{L}$ and $\|w'\| < \gamma_m \lambda_1(\mathcal{L})$. Then we get $w \equiv w' \bmod \mathcal{L}$ and

$$\|w\| + \|w'\| < (\gamma_s + \gamma_m)\lambda_1(\mathcal{L}) < \lambda_1(\mathcal{L}).$$

Thus, by Lemma 2.4 one has $w = w'$. □

Therefore one can deduce properties which ensure that a decryption function based on a GDD solver leads to a valid encryption. Indeed, assume that $c = sH + m$ with m in $B(0, M)$. Then suppose that one has access to `Reduce` which reduces modulo $\mathcal{L}$ in $B(0, R)$. Following Proposition 2.22 if $R + M < \lambda_1(\mathcal{L})$ then $\texttt{Reduce}(c, B) = m$.

2.4.2 NTRU construction

One of the first cryptosystems linked to structured lattices is NTRU [54]. Its first description is as a ring-based system. Let us describe a simple version of the NTRU framework.

NTRU framework Consider a polynomial ring $R = \frac{\mathbb{Z}[X]}{(P(X))}$ where $P(X) \in \mathbb{Z}[X]$ is not necessarily irreducible. The original construction suggests $P(X) = X^n - 1$. The advantage of this polynomial is that operations are efficient, especially the multiplication by the basis elements X^i, which correspond to cyclic shifts of the coefficients. Moreover choose an integer q and denote by R/q the ring $\frac{\mathbb{Z}/q\mathbb{Z}[X]}{(P(X))}$. Then the system is broadly as follows:

1. the *secret key* is a pair (f, g) of short polynomials – their coefficients are small compared to q – such that f is invertible in R/q;
2. the *public key* is $h \in R/q$ such that $h \equiv gf^{-1}$ in R.

Then the NTRU problem is the following: *«Given h, retrieve f and g »*. It can be restated as a lattice problem. If one considers Id to be the identity matrix and H

to be the matrix of the multiplication in R/q by h, one can define the *NTRU lattice* to be generated by the matrix B defined as follows.

$$B = \begin{bmatrix} \mathrm{Id} & H \\ 0 & q\mathrm{Id} \end{bmatrix}.$$

Moreover $h = gf^{-1}$ in R/q is equivalent to the existence of $k \in A$ such that $fh+kq = g$. Therefore the vector of coefficients of $[f, g]$ is equal to $[f, k]B$, so belongs to the NTRU lattice. Then, f and g having small coefficients compared to q, the vector $[f, g]$ is short in the lattice $\mathcal{L}(B)$. For example, if their coefficients are in $\{-1, 0, 1\}$ then $\|[f, g]\|_2 \leqslant \sqrt{2n}$, and the Gaussian heuristic applied on the NTRU lattice gives $\lambda_1 \sim \sqrt{\frac{2nq}{2\pi e}}$. Thus, $[f, g]$ is expected to be the shortest vector of $\mathcal{L}(B)$, and retrieving the secret key amounts to solving an SVP instance.

Security and modifications Since the original paper, several improvements or modification have been suggested, in particular to cope with progress made in lattice reduction [53].

As mentioned the security can be linked to the SVP. It is however over a special category of structured lattices, which are the NTRU lattices. It is unknown if these lattices are weaker than general lattices. Moreover it is possible that the ring structure of A can be used to retrieve the secret key, or speed-up computations. It has been done in [3, 29], where the authors use the relative trace and norm to map the problems to subfields. However these do not introduce a security breach in the NTRU problem with the parameters used in cryptography.

Finally NTRU is a long studied problem, and is believed to be secure despite the lack of strong security proofs. Its framework has been suggested in several candidates for the NIST standardisation process [43, 11].

2.4.3 Learning with errors and variants

We will present the cryptosystems based on another problem called *Learning With Errors* (LWE), and its structured variants.

Learning With Errors

The Learning With Errors problem was defined by O. Regev in 2005 [86], and several cryptosystems suggested over the years rely on its hardness [78].

LWE framework For the LWE problem, one fixes integers n, m and q, as well as two distributions $\mathcal{D}_s$ and $\mathcal{D}_e$ over $\mathbb{Z}^n$ and $\mathbb{Z}^m$ respectively. Then the system is broadly as follows:

1. the *secret key* is a pair $(s, e) \in \mathbb{Z}^n \times \mathbb{Z}^m$ of vectors, drawn following $\mathcal{D}_s \times \mathcal{D}_e$.
2. the *public key* is a pair (A, b), where A is uniformly drawn in $\mathrm{M}_{n,m}(\mathbb{Z})$ and $b = sA + e \bmod q$

Then the LWE problem (search variant) consistsof: *«Given (A, b), retrieve s. »*.
As for NTRU, it can be linked to lattices. Indeed, if one considers the lattice $\mathcal{L}_q(A) = \{x \in \mathbb{Z}^m \mid x \equiv As \bmod q\}$, then for typical LWE parameters e is short compared to the determinant of $\mathcal{L}_q(A)$. Thus retrieving (s, e) amounts to solving a BDD with respect to $\mathcal{L}_q(A)$.

Security We saw that the LWE problem can be rephrased as a lattice problem. Even if it is over a special kind of lattices called the *q-ary lattices* – which contain $q\mathbb{Z}^n$ – the LWE problem enjoys worst-case to average-case reductions [78, 86] and is as hard to solve as problems on lattices. These strong hardness reductions led to several cryptographic constructions. For example, a candidate to the NIST standardisation process built following the LWE framework is Frodo [21].

Ring Learning With Errors

In order to improve efficiency, it has been suggested to modify the LWE setup and place the operations in a polynomial ring [66, 98]. Thus it is usually called *Ring Learning With Errors* (RLWE).

RLWE framework In the RLWE setting, one fixes as parameters a polynomial $P(X)$ and an integer q. Then, as for NTRU, denote by R the polynomial ring $\frac{\mathbb{Z}[X]}{(P(X))}$ and R/q the ring $\frac{\mathbb{Z}/q\mathbb{Z}[X]}{(P(X))}$. Then the system is as follows:

1. the *secret key* is an small element $s \in R/q$ drawn from a distribution $\mathcal{D}_s$, usually uniform;
2. the *public key* is a pair (a, b), where a is uniformly drawn in R and $b = sa + e \bmod q$, with e drawn in R following a distribution $\mathcal{D}_e$.

Clearly, since the elements handled are in polynomial rings, the storage needed is smaller and operations (such as multiplications) are way faster than matrix-vector computations. Again the RLWE problem can be seen as a problem involving lattices, and more precisely ideal lattices. Indeed, in place of A in LWE, one considers here the ideal generated by a in R, which can be viewed as a lattice.

Security It is unclear whether the additional algebraic structure allows faster attacks. It has to be noticed however that as it is the case for the LWE problem, the RLWE problem enjoys worst-case to average-case reductions, and is as hard to solve as problems on *ideal lattices* [66, 98]. An example of cryptosystem based on the RLWE framework is NewHope [4], a candidate to the NIST standardisation process.

Module Learning With Errors

It is possible to obtain a better trade-off between security and efficiency than with the RLWE setting. The operations will again take place in a polynomial ring, but one adds a block structure. More precisely the underlying structure can be seen as a module over the chosen polynomial ring. This leads to the *Module Learning With Errors* (MLWE) setting [23, 61]. Let us describe a simple version of it.

MLWE framework In the MLWE setting, we have a polynomial $P(X)$, an integer q and an integer d. Then denote by R the polynomial ring $\frac{\mathbb{Z}[X]}{(P(X))}$ and R/q the ring $\frac{\mathbb{Z}/q\mathbb{Z}[X]}{(P(X))}$. Then the system is as follows:

1. the *secret key* is an short element $s \in R^d/q$, drawn from a distribution $\mathcal{D}_s$;
2. the *public key* is a pair (a, b), where a is uniformly drawn in R^d and $b = (s \mid a)_{R^d} + e \bmod q$, with e drawn in R following a distribution $\mathcal{D}_e$.

As for NTRU and RLWE, the extra algebraic structure allows better storage and faster computations. The MLWE setting can also be seen as a lattice, defined by a block matrix where each block corresponds to an ideal lattice. Indeed $a \in R^d$ so each of its coordinates $a_i \in R$ defines an ideal lattice.

Security Again the MLWE problem enjoys worst-case to average-case reductions, and is as hard to solve as problems on *module lattices* [23, 61]. Several systems suggested as candidates to the NIST standardisation process are based on the MLWE framework, such as Kyber [20] or Saber [35].

2.4.4 Principal ideal lattices

For a general introduction to ideal lattices and their use in cryptography, one could refer to the survey of Ducas [37].

The Principal Ideal Problem

First let us describe the problems related to ideals that are used in some cryptosystems using ideal lattices.

Definition 2.44 (Principal Ideal Problem (PIP))**.** Given a basis of a *principal ideal* I in a number field K, retrieve a generator of I.

The PIP is referred to as one of the main tasks of Computational Number Theory by H. Cohen in [31]. Generic algorithms solving this problem essentially require the computation of the ideal class group $\mathrm{Cl}(K)$ of the number field K [31]. The best algorithms run in subexponential time. It was first described over imaginary quadratic fields [48], then generalised to arbitrary number fields with fixed dimension [24]. Then several works provided subexponential algorithms to solve the PIP over arbitrary classes of number fields [15, 12] and improvement over cyclotomic fields [14, 18]. Even if the best classical algorithms are subexponential, quantum computing can be used to solve the PIP in polynomial time [13].

Definition 2.45 (Short Principal Ideal Problem (SPIP))**.** Given a basis of a *principal ideal* I in a number field K, generated by a *short element* g, retrieve g or another short generator.

Because of the classical hardness of solving the PIP and the hardness of finding short elements in lattices, several cryptographic constructions were built around the SPIP.

Cryptosystems based on the SPIP

The simplest cryptosystems using ideal lattices such as in [45, 46, 97] are thus based on the problem of finding a short generator of a principal ideal. They can be broadly described as follows.

Consider a number field K and $I = g\mathcal{O}_K$ a principal ideal with a short g when I is considered as a lattice, i.e. the Euclidean norm of g is small compared to the determinant of I. Then the setting is:

1. the *secret key* is g;
2. the *public key* is I, given by a "bad" representation such as its HNF.

The *private key security* relies on the hardness of finding g or another short generator. We will come back to potential attacks below.

2.4.5 Diagonally Dominant Matrices

It is still unknown up to what extent lattices with additional algebraic structure are safe to be used in cryptography. There is no guarantee to that algebraic attacks known to date cannot be extended to cryptographic constructions such as NTRU, RLWE or MLWE. However, using random lattices (which would be safer) is difficult for efficiency reasons. These considerations led some researchers to build cryptosystems linked to less structured lattices. One can mention *Middle-Product Learning With Errors* (MPLWE) [89, 52], which is an adaptation of RLWE trying to remove the structure of quotient ring. Moreover one can consider constructions without any special arithmetical structure behind it. This is the case of *Diagonally Dominant Matrices.*

Definition 2.46 (Diagonally Dominant Matrices)**.** Consider a matrix $M = [m_{i,j}] \in \mathrm{M}_n(\mathbb{R})$. Then M is said to be *diagonally dominant on the rows* or *row-diagonally dominant* if the following holds,

$$\forall i \in [\![1, n]\!], m_{i,i} \geqslant \sum_{\substack{j=1 \\ i \neq j}}^{n} |m_{i,j}|.$$

It is said to be *diagonally dominant on the columns* or *column-diagonally dominant* matrix if the following holds,

$$\forall j \in [\![1, n]\!], m_{j,j} \geqslant \sum_{\substack{i=1 \\ i \neq j}}^{n} |m_{i,j}|.$$

Definition 2.47. A lattice $\mathcal{L}$ is a *diagonally dominant type lattice* (of dimension n) if there is a diagonally dominant matrix B such that $\mathcal{L} = \mathcal{L}(B)$.

Notation. We will write *c.d.d.* for column-diagonally dominant and *r.d.d.* for row-diagonally dominant.

This structure has been used in several cryptographic constructions. One can cite [83], which was a candidate of the round 1 and has known some attacks and variants [101, 96]. The work of [101, 83] relied on the fact that the matrices used as lattice bases are diagonally dominant (or almost), which allows the GDD to be solved with an algorithm adapted from [81].

2.4.6 Analysis of structured lattices

We already mentioned that it is unknown up to what point extra algebraic structures weaken lattice based constructions. Since the NIST call for a post-quantum standardisation, a lot of research is dedicated to study structured lattices.

Reduction algorithms

Several works have studied how to reduce ideal or module lattices, and speed-up computations using said structure. First one can mention an older work [82], which describes an algorithm using ideal structure to speed-up LLL. However the speed-up is only linear. Then, as mentioned, it has known a massive interest throughout the past few years. First let us mention the works of Lee et al. [62] and Mukherjee and Stephens-Davidowitz [71]. They describe an extension of the notion of basis reduction to $\mathcal{O}$-modules, where $\mathcal{O} < \mathcal{O}_K$ is an order of a number field K. These two theoretical works can be completed by the work of Kirchner et al. [59], which did the same with a focus on cyclotomic fields. Moreover, this last work contains extensive practical considerations, which allow the computations of LLL-reduced bases on structured lattices – including ideal lattices – to run considerably faster.

Solving the SVP

In addition to reducing a basis, several works were done to study the possibility of recovering a short vector of ideal or module lattices using their algebraic structure. The work of Cramer et al [33] is the first work. Then another article which extends the former with the use of preprocessing is [62]. This approach has been modified slightly in [9]. Finally, a recent work [75] showed that the problem can be solved in polynomial time for prime ideals in Galois extensions, under specific conditions. It describes a family of ideals over cyclotomic fields for which the ISVP can be solved in polynomial time.

Solving the SPIP

Consider again a principal ideal $I = g\mathcal{O}_K$ of a number field K, such that g is short. A generic way of recovering g is done in two steps:

1. recover a generator h of I, i.e. solve the PIP;

2. find a short generator given h.

As mentioned earlier, the first step is considered a hard problem in classical computational number theory and the best generic algorithm runs in subexponential time. However it can be efficiently done by using quantum computing. Thus in a post-quantum perspective, the security relies on the hardness of the second step, i.e. of retrieving a short generator g from another generator h. This computation is a reduction phase, which is the kind of task that seems difficult even for quantum computers. It is an argument which leaves open the possibility of a post-quantum cryptosystem based on the SPIP. However, as always with structured lattices, one may wonder if the structure can be used to solve the problem.

Log-unit lattice and SPIP In order to solve the SPIP, one may use the structure of the set of generators of I and the Log-unit lattice. First let us describe the overall strategy.

The set of generators of I is $\{gu \mid u \in \mathcal{O}_K^\times\}$. Therefore solving the PIP yields $h = gu$ with $u \in \mathcal{O}_K^\times$. It is then possible to retrieve g from h by finding u. This is where the Log-unit lattice can be used. If we transpose the situation with the Log-embedding, for every generator h we have $\mathrm{Log}_K(h) = \mathrm{Log}_K(g) + \mathrm{Log}_K(u)$. Using that remark and finding the element of the Log-unit lattice closest to h it is possible to retrieve g. This corresponds to solve the CVP with respect to the target h and the lattice $\mathrm{Log}_K(\mathcal{O}_K^\times)$, and even the BDD because we know the generator g is short. The success of this method is therefore dependent on the length of $\mathrm{Log}_K(g)$ and the particular geometry of the Log-unit lattice meaning that we want to have access to a somehow good basis, i.e. orthogonal enough. This approach requires:

1. solving the PIP : this is considered hard classically and can be done in quantum polynomial time;
2. computing $\mathcal{O}_K^\times$: as the PIP this is considered hard classically and can be done in quantum polynomial time;
3. shortening a generator h by solving the BDD with respect to $\mathrm{Log}_K(\mathcal{O}_K^\times)$: this will depend on the basis obtained.

One can remark that since the Log-unit lattice lies in H, the hyperplane orthogonal to $\mathbf{1} = (1, \dots, 1)$, the last step is to be carried out over H. Thus the attack will require the retrieval of $p_H(\mathrm{Log}_K(g))$ from $p_H(\mathrm{Log}_K(h))$, where p_H is the projection operator on H. We will then call $p_H(\mathrm{Log}_K(g))$ the *target (vector)* of the problem. As a matter of fact, step 3 will correspond to a BDD depending on the norm of the target.

Existing results This strategy was mentioned in [26] where it was claimed that in the case of cyclotomic fields the group of cyclotomic units has a good enough geometry in the Log-unit lattice to help recovering a short generator. A proper analysis over cyclotomic fields has been done by Cramer et al. in [34] where the authors gave a bound for the norm of the vectors of the dual basis. In [6] Bauch et al. studied another family of fields, namely the multiquadratic fields, and were able to recover a short generator of an ideal in classical polynomial time for a wide range of fields.

Motivations to study the SPIP Even though the actual propositions of lattice based cryptosystems essentially rely on other problems such as the ISVP it is important to study the SPIP. Indeed such work can help determining which fields or structures are weak. Then one could build upon such analysis a successful strategy for harder problems, or even draw a definitive line between these problems. Moreover one has to remark that one step of the strategy to solve the ISVP [9, 33, 79] is precisely solving an instance of the SPIP.

Finally from a post-quantum perspective the PIP can be solved in polynomial time. Indeed all the number theoretical objects needed can be computed efficiently following [13, 38]. Over general number fields the last unknown is therefore the possibility of retrieving a short generator using the Log-unit lattice. In order to study these problems without a quantum computer, it is important to obtain more efficient algorithms to be able to operate over number fields.

Chapter 3

An encryption using diagonally dominant matrices

3.1 Motivation

The recent call of the NIST for a post-quantum standardisation aims at selecting the best protocols resilient to the quantum computer for encryption, key exchange and digital signature. The third round of this process has been recently completed, and only a few candidates remain of the 69 initially proposed. The algorithms chosen at the end of this process are supposed to become cryptographic standards for the next decade(s).

Meanwhile, research on other cryptographic primitives external to the NIST call still continue. [83] was a candidate of the round 1 and has known some attacks and variants [101, 96]. The work of [101, 83] relied on the fact that the matrices used as secret lattice basis are diagonal-dominant (or almost), as it was often the case for a lot of lattice-based cryptosystems such as the GGH cryptosystem [47] before the apparition of the popular cryptosystems based on NTRU [54] or LWE [86].

We propose in this chapter another encryption primitive based on the basis structure and related reduction algorithms of [81, 83]. While the signature scheme has been shown to have some minor leak [101], an encryption scheme would not have to deal with such issues.

The construction is purely theoretical, while it could be easily implemented with a reasonable efficiency, the goal here is to present a sensible mathematical construction that could be improved if further research is conducted.

3.2 Background and notations

In this chapter we only consider full-rank integral lattices, i.e. such that their bases can be represented by a $n \times n$ non-singular integral matrix.

3.2.1 Framework

Let us now describe the encryption scheme framework we are considering. It is based on l_∞. We fix as parameters $(D, n, M) \in \mathbb{N}^2$.

- Setup(): the secret key $S_K = B \in \mathrm{M}_n(\mathbb{Z})$ is a c.d.d. or r.d.d. matrix with diagonal coefficient D, and the public key P_K is $H = \mathrm{HNF}(B)$.
- The message space is $\mathcal{F}(M) = [\![-M, M]\!]^n$.
- The encryption function will be $\texttt{Encrypt}(m, P_K) = sH + m$, for some $s \in \mathbb{Z}^n$.
- The decryption function will be $\texttt{Decrypt}(c, S_K) = \texttt{Reduce}(c, B)$. The convergence radius of `Reduce` will be denoted by R.

Remark that here, `Reduce` is a GDD solver, not a reduction algorithm like LLL.

In order to obtain a *correct* scheme we need to determine parameters ensuring the correctness of the decryption. As mentioned before, they need to satisfy

$$R + M \leqslant \lambda_1^{(\infty)}(\mathcal{L}). \tag{3.1}$$

We will therefore study $\lambda_1^{(\infty)}$ and the possibility of reducing vectors within a certain radius over a diagonally dominant matrix. Moreover, remark that the existence of such reduction algorithm directly gives an upper bound on the covering radius $\mu^{(\infty)}$ of the corresponding lattice.

3.2.2 Specific notations

Let us consider the matrix $B = (D \times Id_n) + N$. We will use the following objects and notations.

- $CN(B, j) = \sum\limits_{\substack{i=1 \\ i \neq j}}^{n} |b_{i,j}|$ i.e $CN(B, j)$ is the sum of the non-diagonal absolute values of the column j of B.
- $CN(B) = \max\limits_{j \in [\![1,n]\!]} CN(B, j)$.
- $RN(B, i) = \sum\limits_{\substack{j=1 \\ i \neq j}}^{n} |b_{i,j}|$ i.e $RN(B, i)$ is the sum of the non-diagonal absolute values of the row i of B.

- $RN(B) = \max_{i \in [\![1,n]\!]} RN(B,i)$.
- $D \in \mathbb{N}^*$ is called the *diagonal coefficient* of the basis B.
- N is called the *noise matrix* of B and its elements *noise values*.
- For $I \subset [\![1,n]\!]$, we note $B_I \in \mathrm{M}_{|I|,|I|}(\mathbb{Z})$ the submatrix of B composed of the rows and columns of indexes in I. Naturally, if B is a r.d.d/c.d.d matrix, so is B_I.
- $S_\infty(l)$ is the set of positions i given $l \in \mathbb{Z}^n$ such that $|l_i| = \|l\|_\infty$
- $\mathcal{B}(I,B) = \min\left\{\max_{j \in I}\{|(lB)_j| \mid \|l\|_\infty = 1, S_\infty(l) = I\}\right\}$ given any set of indexes I. It is simply $\min\{\|lB_I\|_\infty \mid l \in \{-1,1\}^{|I|}\}$. We denote $\mathcal{B}(I,B)$ by $\mathcal{B}_I$ when B is implied, and stress that $\mathcal{B}_I \neq \lambda_1(B')$.

3.3 Shortest vector and reduction algorithms

In this section we provide generic results on the shortest vector and reduction algorithms regarding diagonal dominant lattices, relative to the infinity norm l_∞.

3.3.1 Short vectors and reduction algorithms for c.d.d. matrices

First let us consider c.d.d. matrices. The results proven in this subsection will prove the following theorem.

Theorem 3.1. *Consider $B \in \mathbb{Z}^n$ a c.d.d. matrix and $\mathcal{L} = \mathcal{L}(B)$. Then $\lambda_1(\mathcal{L}) \geqslant D - CN(B)$ and there is an algorithm* `RSR` *(Alg. 11) running in polynomial time such that*

$$\forall v \in \mathrm{span}(\mathcal{L}), \mathtt{RSR}(v) \equiv v \bmod \mathcal{L}, \|\mathtt{RSR}(v)\|_\infty \leqslant \frac{D + CN(B)}{2}.$$

Consequently one has $\mu^{(\infty)}(\mathcal{L}) \leqslant \frac{D+CN(B)}{2}$.

Short vectors

First let us study the norm of a shortest vector.

Lemma 3.1 (Minimal largest value of non-zero combinations). *Consider $k \in \mathbb{Z}^n \setminus \{0\}$, $j \in [\![1,n]\!]$ such that $|k_j| = \|k\|_\infty$, B be a c.d.d matrix, and $v = kB$. Then one has $|v_j| \geqslant \|k\|_\infty \times (D - CN(B,j))$.*

Proof. Without any loss of generality we can assume $v_i \geq 0$ and $k_j > 0$. Then

$$|v_i| = \left|\sum_{i=1}^{n} k_i b_{i,j}\right| \geqslant k_j D - \sum_{\substack{i=1 \\ i\neq j}}^{n} |k_i b_{i,j}| \geqslant k_j(D - \sum_{\substack{i=1 \\ i\neq j}}^{n} |b_{i,j}|) = k_j(D - CN(B,j)).$$

□

This directly implies that $\lambda_1^{(\infty)}(\mathcal{L}(B)) \geqslant D - CN(B)$. Let us show some additional results on c.d.d. matrices.

Lemma 3.2 (Submatrix bound on non-zero combinations)**.** *Consider B a c.d.d. matrix, $k \in \mathbb{Z}^n$, $I = S_\infty(k)$ and $v = kB$. Then there is $j \in I$ such that $|v_j| \geqslant \mathcal{B}(I, B)$.*

Proof. Clearly if $k \in \{-\|k\|_\infty, 0, \|k\|_\infty\}^n$ then there is $j \in S_\infty(k)$ such that $|v_j| \geqslant \|k\|_\infty \times \mathcal{B}(S_\infty(l), B)$. Now suppose that there is $j_1 \notin S_\infty(k)$ with $k_{j_1} \neq 0$. One can assume $|k_{j_1}| \geqslant |k_j|$ for all $j \notin S_\infty(k)$. Consider the vectors k' and k'' such that $k = k' + k''$ and

$$k'_j = \begin{cases} \text{sign}(k_j)(|k|_\infty - |k_{j_1}|), & \text{if } j \in I \\ 0, & \text{otherwise.} \end{cases}$$

Therefore we also have

$$k''_j = \begin{cases} \text{sign}(k_j)(|k_j|), & \text{if } j \in I \\ k_j, & \text{otherwise.} \end{cases}$$

Remark that for all $j \in S_\infty(k)$ we have $\text{sign}(k''_j) = \text{sign}(k'_j) = \text{sign}(k_i)$ and $|k''_j| = |k''|_\infty$. From what precedes we know that there is $j \in S_\infty(k)$ such that $|(k'B)_j| \geqslant \mathcal{B}(S_\infty(k), B)$. Moreover $S_\infty(k) \subset S_\infty(k'')$ and the signs are the same, so we have $\text{sign}((k''B)_j) = \text{sign}((k'B)_j)$. Thus we obtain $|(kM)_j| \geqslant \mathcal{B}(S_\infty(k), B)$.

□

This gives us the following theorem.

Theorem 3.2 (Bound by the minimal submatrix)**.** *Let B be a c.d.d. matrix. Then $\lambda_1^{(\infty)}(\mathcal{L}(B)) \geqslant \min_{I \subseteq [\![1,n]\!]} \mathcal{B}_I$.*

Reduction algorithms for c.d.d. matrices

Popular lattice reduction algorithms such as LLL or Babaï's algorithms are general purpose algorithms that could prove relatively expensive for large dimensions. Moreover they are targeted on the Euclidean norm l_2. A cheaper (in practice) alternative for reducing vectors modulo a lattice generated by a diagonally dominant matrix was given by Plantard et al. in [81] and successfully applied to their signature scheme. We will propose here a different algorithm relying on the c.d.d structure.

Before we present the full algorithm, we first introduce the core part that we denote by `SingleReduce`. It is described in Algorithm 10, and exhibits nice properties.

Algorithm 10 `SingleReduce`

```
Require: $v \in \mathbb{Z}^n$, $B$ a c.d.d matrix, $R_i \geqslant \frac{D+CN(B,i)}{2}$.
Ensure: $w \equiv v \bmod \mathcal{L}(B)$ and $\|w\|_\infty \leqslant \max(R_i, \|v\|_\infty - (D - CN(B)))$.
 1: $w \leftarrow v$
 2: $s \leftarrow [0, ...., 0] \in \{0,1\}^n$        ▷ Initialise reduction status in all indexes
 3: $i \leftarrow 1$        ▷ initial index
 4: while $\bigvee_{j=1}^{n}((|w_j| > R_j) \wedge (s_j = 0))$ do
 5:     if $|w_i| > R_i$ and $s_i = 0$ then
 6:         $w \leftarrow w - \frac{w_i}{|w_i|}B_i$        ▷ Reduce $|w_i|$
 7:         $s_i \leftarrow 1$        ▷ "Update" the reduction status of index $i$
 8:     end if
 9:     $i \leftarrow (i \bmod n) + 1$
10: end while
11: return $w$
```

Lemma 3.3. *Consider a vector $v \in \mathbb{Z}^n$ and a c.d.d. matrix B with diagonal coefficient D. Moreover let $R \in \mathbb{Z}^n$ be such that $R_i \geqslant \frac{D+CN(B,i)}{2}$. Then* `SingleReduce` *(Alg. 10) transforms v into $w \in \mathbb{Z}^n$ satisfying the following properties.*

1. *$v \equiv w \bmod \mathcal{L}(B)$.*
2. *$\forall i \in [\![1,n]\!], |v_i| > R_i \implies |v_i| > |w_i|$.*
3. *$\forall i \in [\![1,n]\!], |v_i| \leqslant R_i \implies |w_i| \leqslant R_i$.*

Moreover the algorithm performs at most n additions on vectors.

Proof. First remark that we add or remove at most one time each row vector to the variable w during the execution of the algorithm. This is ensured by the flag vector s. Therefore we add at most n vectors to w. Write $v = w^{(0)}, w^{(1)}, \ldots, w^{(r)} = w$ the two by two distinct values of the variable w with $r \leqslant n$. Similarly write $s^{(0)}, \ldots, s^{(r)}$ the different values taken by s. Fix some index $i \in [\![1,n]\!]$. First assume $s_i^{(r)} = 0$. Then we know that $|w_i^{(r)}| \leqslant R_i$ and w_i satisfies the claimed properties. Now assume $s_i^{(r)} = 1$. Let us denote by k_0 the integer such that $w_i^{(k_0)} = w_i^{(k_0-1)} \pm D$. Without loss of generality we can assume $v_i \geqslant 0$. First we consider the case where $w_i^{(0)} > R_i$. Then for some $J \subset [\![1,n]\!] \setminus \{i\}$ we have

$$w_i^{(k_0-1)} = w_i^{(0)} + \sum_{j \in J} \pm b_{j,i} \geqslant w_i^{(0)} - CN(B,i) > R_i - CN(B,i) \geqslant \frac{D - CN(B,i)}{2} > 0$$

therefore $w_i^{k_0} = w_i^{(k_0-1)} - D$. We can write

$$w_i^{(0)} > w_i^{(n)} = w_i^{(0)} - D + \sum_{\substack{j \in [\![1,n]\!] \\ j \neq i}} \pm b_{j,i} > R_i - D - CN(B,i) \geqslant -\frac{D + CN(B,i)}{2}$$

which ensures $|w_i^{(n)}| < |w_i^{(0)}|$. Now consider the case where $w_i^{(0)} \leqslant R_i$. From $\frac{D+CN(B,i)}{2} > CN(B,i)$ we deduce that $w_i^{(k_0-1)} > 0$ and $w_i^{(k_0)} = w_i^{(k_0-1)} - D$. With the same reasoning as before we can conclude $w_i^n < w_i^0$ and $w_i^{(n)} > w_i^{(k_0)} - D - CN(B,i) > -\frac{D+CN(B,i)}{2}$ which ensures $|w_i^{(n)}| \leqslant R_i$. Finally we remark that the results obtained are independent of the choice of i. □

This building block naturally gives us the RSR reduction algorithm, which is guaranteed to finish given a c.d.d. lattice basis.

Algorithm 11 RSR

```
Require: $v \in \mathbb{Z}^n$, $B$ a c.d.d matrix, $R_i \geqslant \frac{D+CN(B,i)}{2}$.
Ensure: $w \equiv v \bmod \mathcal{L}(B)$ and $|w_i| \leq R_i$.
1: $w \leftarrow v$
2: while $\bigvee_{j=1}^{n}(|w_j| > R_j)$ do
3:     $w \leftarrow$SingleReduce($w$,$B$,$R$).
4: end while
5: return $w$
```

Theoretically, there is no general case algorithm that can provide strictly better bounds on l_∞: the covering radius cannot be lower than half the size of the shortest vector, and for $CN(B) = 0$ we do reach this extremity.

Proposition 3.1. *Given a vector $v \in \mathbb{Z}^n$, $R \in \mathbb{Z}^n$ such that $R_i \geqslant \frac{D+CN(B,i)}{2}$ where $D, CN(B,i)$ are associated to a c.d.d. matrix B,* RSR *(Alg. 11) transforms v into $w \in \mathbb{Z}^n$ satisfying the following properties.*

1. *$v \equiv w \bmod \mathcal{L}(B)$.*
2. *$w \in \mathcal{F}(R)$.*

Moreover the algorithm performs at most $n\,\|v\|_\infty$ additions on vectors.

We want to stress this does not show the algorithm is practically efficient: SingleReduce might run a *quadratic* amount of absolute value comparisons on scalars in a single call.

Memory-wise, the algorithm only requires an amount of scalars that is linear in the dimension: this is a significant advantage compared to alternatives that could require at least quadratic amount of elements whose size could be larger than the scalar entries themselves – for example, LLL which requires the computation of the GSO.

3.3.2 Short vector and reduction algorithm on r.d.d

Now let us consider c.d.d. matrices. Again, the results proven in this subsection can be grouped in the following theorem.

Theorem 3.3. *Consider $B \in \mathrm{M}_n(\mathbb{Z})$ a r.d.d. matrix and $\mathcal{L} = \mathcal{L}(B)$. Then $\lambda_1(\mathcal{L}) \geqslant D - RN(B)$ and there is an algorithm* `PSW` *(Alg. 12) running in polynomial time such that*

$$\forall v \in \operatorname{span}(\mathcal{L}), \mathtt{PSW}(v) \equiv v \bmod \mathcal{L}, \|\mathtt{PSW}(v)\|_\infty \leqslant \frac{D + RN(B)}{2}.$$

Consequently one has $\mu^{(\infty)}(\mathcal{L}) \leqslant \frac{D+RN(B)}{2}$.

Short vectors

Exposing a simple relationship between $RN(B)$ and λ_1 does not seem simple, and does not seem to have been studied in detail. We proved that for c.d.d. matrices, a small value of $CN(B)$ enforces the shortest vector to be large. We will show the same property for r.d.d. matrices.

Lemma 3.4. *Let $B \in \mathrm{M}_n(\mathbb{Z})$ be a r.d.d. matrix. Then $\lambda_1^{(\infty)}(\mathcal{L}(B)) \geqslant D - RN(B)$.*

Proof. Consider $l \in \mathbb{Z}^n$, and write $v = lB$. Then write $l' = (|l_i|)_{i \in [\![1,n]\!]}$. Clearly there is $B' \in \mathrm{M}_n(\mathbb{Z})$ a matrix such that $|B'_{i,j}| = |B_{i,j}|$ for any pair $(i,j) \in [\![1,n]\!]^2$, and for all $i \in [\![1,n]\!], B'_{i,i} = D$ and $v_i = \pm(l'B')_i$. Thus B' is a r.d.d. matrix such that $RN(B',i) = RN(B,i)$ for all $i \in [\![1,n]\!]$. Now let us show that $\|v\|_\infty \geqslant D - RN(B)$. We will first bound the taxicab norm, and then use

$$\|v\|_\infty \leqslant \|v\|_1 \leqslant n \|v\|_\infty . \tag{3.2}$$

First remark that we have the following:

$$\|v\|_1 = \sum_{j=1}^{n} |(l'B')_j| \geqslant \left| \sum_{j=1}^{n} \sum_{i=1}^{n} l_i B'_{i,j} \right| .$$

Moreover for any $i \in [\![1,n]\!]$, $l'_i \geqslant 0$ and $D > RN(B,i)$, so we have

$$\left| \sum_{j=1}^{n} \sum_{i=1}^{n} l_i B'_{i,j} \right| = \sum_{j=1}^{n} \sum_{i=1}^{n} l_i B'_{i,j} \geqslant \sum_{i=1}^{n} l'_i (D - RN(B,i)).$$

Therefore, if $k = |\{i \in [\![1, n]\!] \mid l_i \neq 0\}|$ we obtain

$$\|v\|_1 \geqslant k(D - RN(B)).$$

If $k = n$ then Equation (3.2) gives

$$\|v\|_\infty \geqslant D - RN(B).$$

Now consider the case with $k < n$. Without any loss of generality, assume $\forall i \in [\![1, k]\!], l_i \neq 0$. Denote by l'' the tuple $(l'_1, \ldots, l'_k)$ and B'' the top left $k \times k$ submatrix of B'. Then B'' is r.d.d. and $\forall i \in [\![1, k]\!], RN(B'', i) \leqslant RN(B', i) = RN(B, i)$. We have

$$\forall \in [\![1, k]\!], (lB)_i = (l'B')_i = (l''B'')_i.$$

Then, since $|\{i \in [\![1, k]\!] \mid l''_i \neq 0\}| = k$, we can apply the previous result to l'' and B'', therefore $\|l''B''\|_\infty \geqslant D - RN(B'')$ and $\exists i_0 \in [\![1, k]\!], |(l''B'')_{i_0}| = \|l''B''\|_\infty$. Finally we get

$$|(lB)_{i_0}| = |(l'B')_{i_0}| = |(l''B'')_{i_0}| \geqslant D - RN(B'') \geqslant D - RN(B') = D - RN(B).$$

□

r.d.d-specific reduction algorithm

The PSW reduction algorithm we will describe is not new: it was first introduced in [81], and is a known approximation of Babaï's Round-off algorithm [5] in the case of matrices of the form $D - M$ where MD^{-1} has a spectral radius lower than 1. It was then used a second time in cryptography [83] in the case of r.d.d. matrices. The algorithm was proven to finish for $\delta = D$ in [83], but did not take account of the gap between $RN(B)$ and D. A slight modification of the reduction proof given in [96] gives us a tighter bound by changing the loop condition in line 3 of the algorithm to a comparison with a value $R_i = \frac{D+RN(B,i)}{2}$ for every index i. This gives us the modified version, described in Algorithm 12.

Algorithm 12 PSW reduction

```
Require: $v \in Z^n$, $B$ a r.d.d matrix, a vector $R \in \mathbb{N}^n$
Ensure: $w \equiv v \bmod \mathcal{L}(B)$ and $\|w\|_\infty < D$.
1: $w \leftarrow v$
2: while $\bigvee_{j=1}^n (|w_j| > R_j)$ do
3:     $i \leftarrow$ any index such that $|w_i| = \|w\|_\infty$
4:     $w \leftarrow w - \lfloor \frac{w_i}{D} \rceil B_i$                ▷ Reduce $|w_i|$
5: end while
6: return $w$
```

Lemma 3.5 (Tighter bound in PSW-reduction algorithm)**.** *For any $v \in \mathbb{Z}^n$ and a r.d.d. matrix B, the PSW reduction algorithm 12 can output $w \equiv v \bmod \mathcal{L}(B)$ where $\forall i,\ |w_i| \leq \frac{D+RN(B,i)}{2}$.*

Proof. Let f be the function defined on $\mathbb{Z}^n \times [\![1,n]\!]$ by $f : (w,i) \mapsto w - \lfloor \frac{w_i}{D} \rceil B_i$. In order to show that Algorithm 12 ends and outputs a correct vector, we will prove the following:

$$\bigvee_{j=1}^{n} (|w_j| > R_j) \implies \forall i \in S(w,R), \|f(w,i)\|_1 < \|w\|_1 . \tag{3.3}$$

First let us show if the left side of (3.3) is satisfied, then f modifies w. Remark that for all $i \in [\![1,n]\!]$, $f(w,i) = w$ if, and only if, $\lfloor \frac{w_i}{D} \rceil = 0$, which is clearly equivalent to $|w_i| \in [\![-\frac{D}{2}, \frac{D}{2}]\!]$. This condition is clearly satisfied for any $i \in [\![1,n]\!]$ such that $|w_i| > R_i$. Now let us show that (3.3) is true. First assume that there is $i \in S(w,R)$ such that $|w_i| > D$. Then $f(w,i)_i$ has the same sign as w_i, therefore $|f(w,i)| = |w_i| - \lfloor \frac{w_i}{D} \rceil D$. Moreover we have

$$\forall j \in [\![1,n]\!] \setminus \{i\}, |w_j| \leqslant |w_j| + \left\lfloor \frac{w_i}{D} \right\rceil |B_{i,j}|,$$

which gives

$$\|f(w,i)\|_1 \leqslant |f(w,i)_i| + \sum_{\substack{j=1 \\ j \neq j}}^{n} |f(w,i)_j| \leqslant |w_i| - \left\lfloor \frac{w_i}{D} \right\rceil D + \sum_{\substack{j=1 \\ j \neq i}}^{n} |w_j| + \left\lfloor \frac{w_i}{D} \right\rceil |B_{i,j}|.$$

This leads to

$$\|f(w,i)\|_1 \leqslant \|w\|_1 + \left\lfloor \frac{w_i}{D} \right\rceil (RN(B,i) - D) \leqslant \|w\|_1 - \left\lfloor \frac{w_i}{D} \right\rceil < \|w\|_1 .$$

Now consider $i \in S(w,R)$ such that $|w_i| < D$. Then $\lfloor \frac{w_i}{D} \rceil = 1$, and the signs of w_i and $f(w,i)_i$ are different. Moreover if we write $|w_i| = R_i + t$ with $t \in [\![1, \frac{D-RN(B)}{2}]\!]$, we obtain $|f(w,i)_i| = |R_i - D + t| = \frac{D-RN(B,i)}{2} - t$. Therefore we have

$$|f(w,i)_i| = \frac{D + RN(B,i)}{2} - t - RN(B,i) = |w_i| - RN(B,i) - 2t.$$

Following the same reasoning as before to bound $\|f(w,i)\|_1$ we obtain

$$\|f(w,i)\|_1 \leqslant \|w\|_1 - RN(B,i) - 2t + RN(B,i) < \|w\|_1 .$$

□

Note that again, there is no general polynomial-time algorithm that will give strictly better bounds on l_∞ in every case: by setting $RN(B) = 0$ we do obtain a covering radius that is half the size of the shortest vector.

This algorithm like RSR uses a linear memory. The average-case time-complexity of the algorithm was briefly experimentally hinted in [81], however a proper worst-case analysis is not as simple here as in RSR and does not seem to have been done in the literature.

Proposition 3.2. *Let B be a r.d.d. matrix and $v \in \mathbb{Z}^n$, and denote by b the value $\frac{nD}{nD-(D-RN(B))}$. An upper bound on the worst-case vector operations complexity of* `PSW` *is*

$$O\left(\log_b\left(\frac{\|v\|_1}{D}\right) + \frac{nD}{2}.\right)$$

Proof. Let us consider the reduction of $\|w\|_1$ to count the number of reduction steps. Using the reasoning of the above proof, we will consider two cases: $\|w\|_\infty > D$ and $\|w\|_\infty \leqslant D$. Assume first that $\|w\|_\infty > D$, and denote by w' the value of the vector after the update in step 4 of Algorithm 12. Then $\|w\|_1$ is updated as

$$\|w'\|_1 = \|w\|_1 - qD + qRN(B) = \|w\|_1 - q(D - RN(B))$$

with $q = \left\lceil \frac{\|w\|_\infty}{D} \right\rfloor > 1$. From $\|w\|_\infty \leqslant \|w\|_1 \leqslant n\,\|w\|_\infty$ we obtain $q \geqslant \frac{\|v\|_1}{nD}$. Thus we get

$$\|w'\|_1 \leq \|w\|_1 - \frac{\|w\|_1}{nD}(D - RN(B)) = \|w\|_1 \left(\frac{nD - (D - RN(B))}{nD}\right)$$

If we use this inequality and we write k for the number of steps necessary to reach the condition $\|w\|_\infty \leqslant D$, we obtain in the worst case

$$\|w\|_1 = \left(\frac{nD - (D - RN(B))}{nD}\right)^k \|v\|_1 \leqslant D.$$

This gives $O\left(\log_{\frac{nD}{nD-(D-RN(B))}}\left(\frac{\|v\|_1}{D}\right)\right)$ number of vector operations to reach $\|w\|_\infty \leqslant D$. When this condition is true, each step reduces $\|w\|_1$ by at least 2, and the worst-case scenario would be to reduce until $\|w\|_1 = 0$. Therefore, it would require $\frac{\|w\|_1}{2} \leqslant \frac{nD}{2}$ iterations. Thus, the final worst-case complexity analysis in terms of *vector operations* is

$$O\left(\log_{\frac{nD}{nD-(D-RN(B))}}\left(\frac{\|v\|_1}{D}\right) + \frac{nD}{2}\right)$$

□

This overestimated complexity does not reflect at all the experimental results reported in [81, 96, 83], which is understandable: the probability to trigger a *single* worst-case iteration is 2^{n-1}, i.e as probable as solving a $\{0,1\}$-knapsack problem randomly. However, our result still proves polynomial operation complexity and constant memory as far as vector operations (i.e fixed dimension) are concerned.

3.4 DRE: Diagonal Dominant Encryption scheme

We provide in this section an encryption scheme based on our previous results, following the framework described. It is a simple application to demonstrate there could be some practical use to our earlier study: as diagonal dominant lattices were successfully used to create the DRS signature scheme [83, 96], we here "create" DRE.

The construction is purely theoretical, while it could be easily implemented with a reasonable efficiency, the goal here is to present a sensible mathematical construction that could be improved if further research is conducted.

3.4.1 Correctness of the scheme

Using the previous properties and as little structure as possible, we can deduce a sufficient (but not necessary) condition for a cryptosystem to be correct. We showed that a c.d.d. (resp. r.d.d.) matrix B satisfies $\lambda_1^{(\infty)}(\mathcal{L}(B)) \geqslant D - CN(B)$ (resp. $D - RN(B)$). Moreover we have access to Algorithm 11 (resp. 12) which reduces any vector v to $w \equiv v \bmod \mathcal{L}(B)$ such that $\|w\|_\infty \leqslant R$ with $R = \frac{D+CN(B)}{2}$ (resp. $R = \frac{D+RN(B)}{2}$). Therefore, if $\mathcal{F}(M)$ is the message space, following Eq. (3.1) the different parameters have to be such that

$$M + \frac{D + CN(B)}{2} < D - CN(B) \qquad \text{(resp. } M + \frac{D + RN(B)}{2} < D - RN(B))$$

This leads to

$$CN(B) < \frac{D - 2M}{3} \qquad \text{(resp. } RN(B) < \frac{D - 2M}{3}) \tag{3.4}$$

which is very easy to construct. It is important to note that a larger shorter vector or a smaller convergence radius R immediately leads to a weaker condition for $CN(B)$ (resp. $RN(B)$). Remark that the choice of M also influences this condition.

3.4.2 Instantiation of the encryption scheme

To instantiate our encryption scheme, we first need to fix some public parameters as the diagonal coefficient D, and the dimension n. We assume the message space is composed of vectors over $\{-1, 0, 1\}^n$, but we showed earlier that could also be subject to change. We also have to choose between a r.d.d ($RN(B)$ to be fixed) and a c.d.d ($CN(B)$ to be fixed).

From an external point of view, our scheme is actually a knapsack problem, such as the first proposition of Merkle-Hellman [67]. The major difference is within the setup and the decryption, which are details that are hidden from message senders.

We will describe a possible instantiation of DRE using c.d.d. matrices. Again, since the bounds proved for c.d.d. and r.d.d. matrices are identical, all of what follows can be done for r.d.d. matrices. One only has to replace $CN(B)$ by $RN(B)$ and use the corresponding reduction algorithm.

Setup

The setup is composed of two steps. For the secret key, we generate a diagonal dominant matrix with our chosen parameters (D, n). Since the message space is $P(1) = [\![-1, 1]\!]^n$, following Equation (3.4), we will fix $CN(B) = \frac{D-2}{3}$.

For the public key, we compute the HNF of the secret key, assuming it has perfect form. If the HNF does not hold a perfect form, we can choose to discard the key or use a permutation to attempt obtaining a perfect HNF as reported in [96].

The public key is then the resulting HNF, with a small twist: we choose to remove the determinant of the lattice, to effectively transform our modular knapsack instance into a knapsack problem. This also removes information about the lattice, which decreases the success rate of lattice reduction attacks for key recovery, and leave P_K as a set of $n-1$ large integers.

Encryption

Since our public key is a knapsack problem, we just sum or subtract the corresponding values of the public key P_K according to our message m. The resulting integer is our ciphertext c.

Because the keys (B, H) are chosen such that $H = \text{HNF}(B)$ is perfect and h is the last column of H minus the last coefficient, the output of `Encrypt` as described in Algorithm 14 is the last coefficient of a vector of the form $[0, \ldots, 0, c] = m + v$

Algorithm 13 `Setup`

```
Require: $(D, n) \in \mathbb{N}^2$.
Ensure: $(P_K, S_K)$ the public and secret keys
1: $CN(B) \leftarrow \frac{D-2}{3}$
2: $B \leftarrow$ CDDgen$(D, n, CN(B))$
3: $H \leftarrow \mathrm{HNF}(B)$
4: while IsPerfect$(H) =$ false do
5:     $B \leftarrow$ CDDgen$(D, n, CN(B))$
6:     $H \leftarrow$ HNF(B)
7: end while
8: $h \leftarrow H[1..n-1, n]$
9: return $(B, h)$
```

Algorithm 14 `Encrypt`

```
Require: A plaintext $m \in [\![-1, 1]\!]^n$ and the public key $P_K = h \in \mathbb{Z}^{n-1}$.
Ensure: A ciphertext $c$
1: $c \leftarrow 0$
2: for $i = 1$ to $n-1$ do
3:     $c \leftarrow c - m_i h_i$
4: end for
5: $c \leftarrow c + m_n$
6: return $c$
```

with $v \in \mathcal{L}(B)$. Indeed, if one reduces the vector m with the HNF H, as follows

$$\left[\begin{array}{ccccc} m_1 & \dots & \dots & m_{n-1} & m_n \\ \hline 1 & 0 & \dots & 0 & h_1 \\ 0 & 1 & \ddots & \vdots & \vdots \\ \vdots & \ddots & \ddots & 0 & \vdots \\ 0 & \dots & 0 & 1 & h_{n-1} \\ 0 & \dots & \dots & 0 & \det(B) \end{array}\right],$$

then using the first $n-1$ rows of H one can remark that the first vector will be transformed into

$$[0, \dots, 0, m_n - \sum_{i=1}^{n-1} m_i h_i] = m - mH + m_n[0, \dots, 0, \det(B)].$$

Decryption

We can use the reduction algorithms studied earlier to recover m from c. From our study, Algorithm 15 will output the correct plaintext m.

Algorithm 15 `Decrypt`

```
Require: A ciphertext $c = \texttt{Encrypt}(m, h) \in \mathbb{Z}^n$ and the secret key $S_K = B$.
Ensure: The plaintext $m$
1: $CN(B) \leftarrow \frac{D-2}{3}$
2: $R \leftarrow [CN(B), \ldots, CN(B)]$
3: $m \leftarrow c \pmod{\det(B)}$                ▷ Reduction modulo the determinant
4: $m \leftarrow [0, \ldots, 0, m]$
5: $m \leftarrow \texttt{RSR}(m, B, R)$
6: return $m$
```

Further work

The encryption scheme we described previously is essentially a toy example. Before considering it as usable scheme, one would need to assess the following points.

(*i*) The first thing to do would be to estimate the security provided by such a system.

(*ii*) The key generation is relatively slow as it requires the computation of HNF of large matrices. One could look into using the structure of diagonally dominant matrices in order to accelerate HNF computations.

(*iii*) Then the reduction algorithms might be improved. Experimentally, it is especially the case of `RSR` for c.d.d. matrices.

Chapter 4

Practical computations in number fields

The main goal of this thesis is to study ideal lattices, with a special concern for high degree number fields. Indeed, cryptographic sizes are large (at least larger than 256). In order to obtain data for such dimensions, a significant part of our work has been to implement and improve in practice some computational tasks over number fields. In particular, in our study of real Kummer extensions of the form $\mathbb{Q}(\sqrt[p]{m_1}, \ldots, \sqrt[p]{m_r})$ or $K(\sqrt[p]{m_1}, \ldots, \sqrt[p]{m_r})$ with $L = \mathbb{Q}(\sqrt[p]{n_1}, \ldots, \sqrt[p]{n_s})$, we developed recursive algorithms (which are generalisations of the work done over multiquadratic fields by Bauch et al. [6]) to compute the unit group and solve the PIP. These two main algorithms require two tasks which can be costly over large degree number fields, even if they run in polynomial time. These are the computation of relative norms of ideals and the extraction of p-th roots of elements. In this chapter we will present the practical improvements that we made regarding these two tasks. More precisely, the chapter is as follows.

- We study in Section 4.1 two methods of computing norms of ideals relative to extensions L/K. The first one is certified and runs in polynomial time over extensions such that the Galois closure of L/K satisfies $[\widetilde{L} : L] = \text{Poly}([L : \mathbb{Q}])$. This is the case of the Kummer fields considered in this thesis. The second is heuristic and probabilistic. We were not able to prove an acceptable bound of its running time, but its practical efficiency is very good compared to our first method and the implementation of MAGMA [22].

- In Section 4.2 we develop a method to retrieve the roots of a polynomial $f(X) \in L[X]$ where L is a number field. It runs in polynomial time and uses complex embeddings. Moreover, we show how it can be adapted to take advantage of an extension structure L/K, in order to decode approximations relative to K instead of L. This comes at the cost of searching in a large set which makes this process impractical when $[L : K]$ increases. We also present several heuristic observations which allow both our methods to compete with the classical algorithm implemented in PARI/GP [76]. It is particularly the case over Kummer extensions and small degree polynomial equations.

4.1 Relative norms of ideals

The computation of relative norms of ideals can be computed following several methods, depending on which characterisation of the norm one considers. In [30], H. Cohen shows how one can compute efficiently the relative norm from a pseudo-Hermite Normal Form (pseudo-HNF) of an ideal. Then if one considers that $\mathrm{N}_{L/K}(I)$ can be expressed through the product of ideals in the Galois extension following Equation (2.9), it is possible to compute this product. Finally it is possible to use a compact representation of ideals called the *two-elements representation* in order to compute the relative norms more efficiently.

First let us define the two-elements representation of an ideal.

Definition 4.1 (Two-elements representation)**.** Let K be a number field, and I be an ideal of K. Then a pair $(\alpha, \beta) \in I^2$ is called a *two-elements* representation of I if $I = \alpha\mathcal{O}_K + \beta\mathcal{O}_K$.

Proposition 4.1. 1. *Consider a number field K, an ideal I of K and $\alpha \in I$. Then there is $\beta \in I$ such that (α, β) is a two-elements representation of I.*

2. *Consider an extension of number fields L/K, an ideal I of L. Then there is $(\alpha, \beta) \in I^2$ such that $(\mathrm{N}_{L/K}(\alpha), \mathrm{N}_{L/K}(\beta))$ is a two-elements representation of $\mathrm{N}_{L/K}(I)$.*

One can find in [30] the following probabilistic algorithm which computes a two-elements representation of an ideal, from an integral basis.

Notation. We will denote by `RandomElement` the procedure which given a family $\mathcal{B} = (b_1, \ldots, b_n)$ and a range $R \in \mathbb{N}$, outputs an element in the $\mathbb{Z}$-module generated by $\mathcal{B}$ which coefficients are drawn uniformly at random in $[\![-N, N]\!]$.

We do not specify the range for the procedure `RandomElement` used in Algorithm 16, nor the shape of the basis B. In [30], Cohen picks a LLL-reduced basis and a range equal to 3. This procedure can be costly when the dimension of K is large, because of the use of LLL. When the determinant (i.e. the absolute norm of the ideal) is known or easily computable – which is typically the case when the ideal is given by its HNF – one can replace the reduction by LLL by a reduction modulo the determinant.

Algorithm 16 `TwoElements`

```
Require: An ideal $I$ of a number field $K$ given by an integral basis matrix $B$
Ensure: A pair $(\alpha, \beta)$ being a two-elements representation of $I$
 1: $\alpha \leftarrow$ RandomElement($B$)
 2: $H_\alpha \leftarrow \mathrm{HNF}((\alpha))$
 3: $d \leftarrow \det H_\alpha$
 4: while $d \neq \det B$ do
 5:     $\beta \leftarrow$ RandomElement($B$)
 6:     $H_\beta \leftarrow \mathrm{HNF}((\beta))$
 7:     $H \leftarrow \mathrm{HNF}\left([H_\alpha \mid H_\beta]^\intercal\right)$
 8:     $d \leftarrow \det H$
 9: end while
10: return $(\alpha, \beta)$
```

Product of two ideals There are several ways of computing the product of two ideals, depending on the choice of representation. Consider two ideals I and J, each given by an integral basis. Let us denote by $(e_i)_i$ and $(f_i)_i$ these bases. Then IJ is generated over $\mathbb{Z}$ by the products $(e_i f_j)_{i,j}$. Thus, if I and J are given by their HNF in a fixed basis, one could recover the HNF of IJ by computing the HNF of all the products. It amounts to computing the HNF of a $n^2 \times n$ matrix, where n is the dimension of the field. This method is clearly polynomial in the dimension, but can still be long especially if n is large. In order to speed-up this naive process, one can use the more compact two-elements representation. If I is given by (α, β) and J by its HNF H_J then IJ is generated over $\mathbb{Z}$ by

$$\begin{bmatrix} \alpha H_J \\ \beta H_J \end{bmatrix}.$$

Then one needs to compute a HNF of a $2n \times n$ matrix with this method.

For the following, let us fix L/K an extension of number fields and write $H = \mathrm{Hom}(L/K, \mathbb{C}) = \{\sigma_1, \ldots, \sigma_n\}$. Let N be the absolute degree of L.

4.1.1 Product of the conjugates

We will first present the algorithm computing the product of the $\sigma(I)$ where σ runs through H. The global procedure is described in Algorithm 17.

Quick analysis One can see that $[L : K] - 1$ products are computed in $\tilde{L}$ in Algorithm 17. However one needs to be careful with the dimension. Indeed the products can be made in the smallest extension containing J and $\sigma_i(I)$. Each step can multiply the dimension by at most $[L : K] = n$. Thus in the worst-case, one

Algorithm 17 `NaiveRelativeNorm`

```
Require: An ideal $I$ of a number field extension $L/K$
Ensure: The relative norm $\mathrm{N}_{L/K}(I)$
1: $J \leftarrow I$
2: for $i = 2$ to $n$ do                          ▷ Assuming that $\sigma_1 = \mathrm{Id}$
3:     $J \leftarrow J\sigma_i(I)$
4: end for
5: return $J \cap K$
```

needs to compute at step i a HNF of a $N^2 n^{i-2} \times Nn^{i-1}$ matrix. This leads to a matrix of maximum size $N^2 n^{n-2} \times Nn^{n-1}$, in the worst case. In this situation and $n = O(N)$ the final matrix size is exponential in N. However if $n = O(\ln N)$ then it is polynomial in N. Finally, if the degree of $\tilde{L}/L$ is polynomial in N then the size of the matrices stays polynomial in N and the final complexity is also polynomial. This leads to the following result.

Proposition 4.2. *Consider $\mathcal{F}$ a family of extensions of number fields L/K such that over $\mathcal{F}$, $[\tilde{L} : L] = \mathrm{Poly}([L : \mathbb{Q}])$. Then Algorithm 17 runs in polynomial time over $\mathcal{F}$.*

Remark 19. Given a Kummer extension L/K of exponent n, its Galois closure is in $L(\zeta_n)$. Thus one has $[\tilde{L} : L] \leqslant [L : K]$, and following Proposition 4.2 Algorithm 17 runs in $\mathrm{Poly}([L : \mathbb{Q}])$.

Speeding-up the computation We will now present heuristic strategies we implemented to speed up the computations. The main idea is to check from time to time during the computation if we reached our goal. This can be done with the determinant. Indeed for any ideal I, $\det \mathrm{HNF}(I) = \mathrm{N}_{L/\mathbb{Q}}(I) = \mathrm{N}_{K/\mathbb{Q}}(\mathrm{N}_{L/K}(I)) = \det \mathrm{HNF}(\mathrm{N}_{L/K}(I))$. Thus at chosen points during the computation, one can check whether the current ideal – or equivalently current sublattice – has the targeted determinant. Moreover one can remark that the intersection $\cap_i \sigma_i(I)$ is an ideal containing $\prod_i \sigma_i(L)$, and faster to compute given access only to the HNF of the ideals. It is therefore possible to compute the intersection at the beginning of the computation and then intersect it with the ideal currently computed in hope to find the desired ideal. These ideas lead to Algorithm 18.

We will report in Subsection 4.1.3 on the practical speed-up these ideas offer.

Notation. We will denote the methods of Algorithm 17 (resp. Algorithm 18) by `NaiveRelativeNorm_hnf` (resp. `RelativeNorm_hnf`) and `NaiveRelativeNorm_2el` (resp. `RelativeNorm_2el`) when only HNF are used for the products, or when two-elements representations are also used.

Algorithm 18 `RelativeNorm`

```
Require: An ideal $I$ of a number field extension $L/K$
Ensure: The relative norm $\mathrm{N}_{L/K}(I)$
1: $H \leftarrow \cap_{i=1}^{n} \sigma_i(I)$
2: $J \leftarrow I$
3: for $i = 2$ to $n$ do                                  ▷ Assuming that $\sigma_1 = \mathrm{Id}$
4:     $J \leftarrow J\sigma_i(I)$
5:     if $\mathrm{N}_{K/\mathbb{Q}}((J \cap H) \cap K) = \mathrm{N}_{L/\mathbb{Q}}(I)$ then
6:         return $J \cap K$
7:     end if
8: end for
9: return $J \cap K$
```

4.1.2 Algorithm 2: probabilistic algorithms

Let us now present two probabilistic algorithms to compute $\mathrm{N}_{L/K}(I)$. They follow two different strategies, but are both inspired by the use of `RandomElement` in [30].

- The first is using Definition 2.34 of the relative norm.
- The second is inspired by the two-elements representation and Proposition 4.1.

The disadvantage of these two methods is that they are probabilistic. However they behave well in practice and do not require computations in the Galois closure of L/K.

A first method Following Definition 2.34, $\mathrm{N}_{L/K}(I)$ is the ideal of $\mathcal{O}_K$ generated by the elements of the form $\mathrm{N}_{L/K}(x)$ where $x \in I$. Thus, provided that one is able to compute random elements of I, a simple strategy is to compute such elements until the ideal they generate is the target $\mathrm{N}_{L/K}(I)$. This equality can again be tested by the absolute norm. If the elements $x \in I$ are sufficiently random in I, one might expect their relative norms $\mathrm{N}_{L/K}(x)$ to be also random in $\mathrm{N}_{L/K}(I)$, thus quickly generating the relative norm of I. This method is summed up in Algorithm 19.

Computing the two-elements representation The method explained previously compute an integral basis of $\mathrm{N}_{L/K}(I)$. In order to obtain a two-elements representation, then one needs to use Algorithm 16 after Algorithm 19. One can however design a probabilistic and heuristic algorithm to compute it at once. Indeed following Proposition 4.1, there is $(\alpha, \beta) \in I^2$ such that $\mathrm{N}_{L/K}(I) = \langle \mathrm{N}_{L/K}(\alpha), \mathrm{N}_{L/K}(\beta)\rangle_{\mathcal{O}_K}$. Thus following the same rationale as for Algorithms 16 and 19, one can expect to find such a pair doing the following. First fix an element α, then compute random elements β until $(\mathrm{N}_{L/K}(\alpha), \mathrm{N}_{L/K}(\beta))$ generates the ideal $\mathrm{N}_{L/K}(I)$. This can be found in Algorithm 20.

Algorithm 19 `RelativeNormProb`

```
Require: An ideal $I$ of a number field extension $L/K$ given by an integral basis matrix $B$
Ensure: An integral basis $H$ of $\mathrm{N}_{L/K}(I)$
1: $x \leftarrow$ RandomElement($B$)
2: $a \leftarrow \mathrm{N}_{L/K}(x)$
3: $H \leftarrow \mathrm{HNF}\,((a))$
4: $d \leftarrow \det H$
5: while $d \neq \det B$ do
6:     $x \leftarrow$ RandomElement($B$)
7:     $a \leftarrow \mathrm{N}_{L/K}(x)$
8:     $H \leftarrow \mathrm{HNF}\left([H \mid \mathrm{HNF}((a))]^{\mathsf{T}}\right)$
9:     $d \leftarrow \det H$
10: end while
11: return $H$
```

Algorithm 20 `RelativeNormTwoEl`

```
Require: An ideal $I$ of a number field extension $L/K$ given by an integral basis matrix $B$
Ensure: A two-elements representation $(\mathrm{N}_{L/K}(\alpha), \mathrm{N}_{L/K}(\beta))$ of $\mathrm{N}_{L/K}(I)$
1: $\alpha \leftarrow$ RandomElement($B$)
2: $H_\alpha \leftarrow \mathrm{HNF}\left((\mathrm{N}_{L/K}(\alpha))\right)$
3: $d \leftarrow \det H_\alpha$
4: while $d \neq \det B$ do
5:     $\beta \leftarrow$ RandomElement($B$)
6:     $H_\beta \leftarrow \mathrm{HNF}\left((\mathrm{N}_{L/K}(\beta))\right)$
7:     $H \leftarrow \mathrm{HNF}\left([H_\alpha \mid H_\beta]^{\mathsf{T}}\right)$
8:     $d \leftarrow \det H$
9: end while
10: return $\left(\mathrm{N}_{L/K}(\alpha), \mathrm{N}_{L/K}(\beta)\right)$
```

Analysis As mentioned, Algorithms 19 and 20 are probabilistic (and heuristic) algorithms, contrary to Algorithm 17. They should however run in polynomial time, without conditions on the extension L/K. In particular, the degree of the Galois closure should not impact the running time, since one can compute the relative norm of an element $x \in L/K$ in polynomial time, without needing to compute the product $\prod_{i=1}^{n} \sigma_i(x)$ in the Galois closure.

4.1.3 Experimental results

Computation through product of ideals

First let us explore the practical performances of Algorithms 17 and 18. We will first study the impact of using a two-elements representation to compute the product of ideals, then compare the naive method of Algorithm 17 with the improved one of Algorithm 18. We implemented both algorithms over real Kummer extensions studied in Section 5.2. Such number field extensions L/K satisfy $L = K(\sqrt[p]{m_r})$ and $K = \mathbb{Q}(\sqrt[p]{m_1}, \ldots, \sqrt[p]{m_{r-1}})$, where p is a prime integer and $(m_i)_{i \in [\![1,r]\!]} \in \mathbb{Z}^r$. We did experiments for several p. For each p we fixed (m_i) to be the first r primes, with increasing r.

HNF vs two-elements representation As previously mentioned, using a two-elements representation of an ideal I together with the HNF matrix of a second ideal J allowsfor the computation of a basis of the product IJ. It reduces a $2n \times n$ matrix instead of a $n^2 \times n$ matrix. However in Algorithm 17 the products are done in the Galois closure of the extension L/K. Thus the rank of the lattice at hand (generated by the rows of the matrix obtained) is susceptible to increase after each product of the form $J\sigma_i(J)$ during the computation. If one only uses HNF representation as in `NaiveRelativeNorm_hnf`, then the maximal possible rank after a product will be $\min\{n^2, [\tilde{L} : \mathbb{Q}]\}$. If one uses a two-elements representation as in `NaiveRelativeNorm_2el` then it is $\min\{2n, [\tilde{L} : \mathbb{Q}]\}$. This means that the lattices computed with the two different representations can be different at given steps during the computation.

One can find in Table 4.1 the timings obtained for both representations and real Kummer fields of exponents $3, 5$.

Table 4.1: Average timings for `NaiveRelativeNorm_hnf` and `NaiveRelativeNorm_2el` in real Kummer fields $\mathbb{Q}(\sqrt[p]{2}, \sqrt[p]{3}, \ldots, \sqrt[p]{p_r})$

(a) $p = 3$

Sequence length r	$[L : \mathbb{Q}]$	$[\tilde{L} : \mathbb{Q}]$	Time for HNF	Time with two-elements
2	9	18	0.2522	0.07860
3	27	54	8.182	1.026

(b) $p = 5$

Sequence length r	$[L : \mathbb{Q}]$	$[\tilde{L} : \mathbb{Q}]$	Time for HNF	Time with two-elements
2	25	100	47.59	5.453

From the data gathered, one can see the influence of the Galois closure, and of the representation used. When p increases, the Galois closure is larger comparatively to the field L. Thus one obtains longer computations for fields with similar degrees. Compare for example $L = \mathbb{Q}(\sqrt[3]{2}, \sqrt[3]{3}, \sqrt[3]{5})$ and $L = \mathbb{Q}(\sqrt[5]{2}, \sqrt[5]{3})$. Moreover, one can clearly remark that using the two-elements representation for the products is faster than using the HNF for both ideals. From our computations, the rank of J is maximal in the Galois closure after one product when using only HNF, whereas when using a two-elements representation the rank of J is multiplied by two after each product.

Improved versions Let us now compare the performances of Algorithms 17 and 18, again with both representations for ideals. We will also distinguish two variants of Algorithm 18: if the intersection $\cap_{i=1}^{n}\sigma_i(I)$ is computed or not. Since one checks after each product $J\sigma_i(I)$ if $\mathrm{N}_{L/K}(I)$ has been computed, we also give details step by step. More precisely, in what follows, "product i" will designate the state of computation after the i-th product $J\sigma_i(I)$. By extension, "product 0" will be the state after the first intersection $\cap_i\sigma_i(I)$ has been computed in Algorithm 18. For each field with exponent p and any integer $i \in [\![1, p-1]\!]$ we provide the percentage of ideals I whose norm has been successfully computed after the i-th product, and the corresponding average computation times. Moreover we provide the average computation time for the complete set of ideals considered.

Without the intersection: First we consider the version of Algorithm 18 where the intersection is not computed. Recall that the matrices representing the ideal is in HNF so computing the determinant of its intersection with K is fast. The data gathered can be found in Table 4.2.

Table 4.2: Experimental results for `RelativeNorm` in real Kummer fields $\mathbb{Q}(\sqrt[p]{2}, \sqrt[p]{3}, \dots, \sqrt[p]{p_r})$, without computing $\cap_i\sigma_i(I)$

(a) $p = 3$

		HNF			Two-elements		
	Product #	1	2	Total	1	2	Total
$r = 2$	Percentage	75.8	24.2	100	0	100	100
	Average Time	0.036	0.111	0.054	–	0.081	0.081
$r = 3$	Percentage	84.2	15.8	100	0	100	100
	Average Time	0.568	3.88	1.09	–	1.18	1.18

(b) $p = 5$

		HNF					Two-elements				
	Product #	1	2	3	4	Total	1	2	3	4	Total
$r = 2$	Percentage	86.8	0.4	4	8.8	100	0	0	37.2	62.8	100
	Average Time	1.97	8.43	22.31	35.45	5.74	–	–	3.78	5.17	4.65

One can remark that the performances of the naive method and Algorithm 18 are similar when using a two-elements representation for the products. However Algorithm 18 performs way better when using HNF only. It even outperforms the method where one uses two-elements representations in some cases. This is due to the fact that the lattices computed in the intermediate steps are not the same depending on the choice of ideal representation. Then the targeted ideal is not found at the same step. With `RelativeNorm_hnf`, one can see in Table 4.2 that one finds $\mathrm{N}_{L/K}(I)$ in the first steps with overwhelming probability. However with `RelativeNorm_2el` it is generally computed only in the last steps.

With the intersection: Now consider the case where the intersection ideal $\cap_i \sigma_i(I)$ is computed at the beginning of Algorithm 18 and used each time we check if we have already obtained $\mathrm{N}_{L/K}(I)$. One can find the results of our computations in Table 4.3.

Table 4.3: Experimental results for `RelativeNorm` in real Kummer fields $\mathbb{Q}(\sqrt[p]{2}, \sqrt[p]{3}, \ldots, \sqrt[p]{p_r})$, with the computation of $\cap_i \sigma_i(I)$

(a) $p = 3$

		HNF				Two-elements			
	Product #	0	1	2	Total	0	1	2	Total
$r = 2$	Percentage	59.6	14	26.4	100	59.6	0	40.4	100
	Average Time	0.003	0.035	0.112	0.037	0.002	–	0.082	0.034
$r = 3$	Percentage	54.6	26.2	19.2	100	54.6	0	45.4	100
	Average Time	0.028	0.892	3.231	0.869	0.028	–	0.973	0.457

(b) $p = 5$

		HNF						Two-elements					
	Product #	0	1	2	3	4	Total	0	1	2	3	4	Total
$r = 2$	Percentage	79	9.8	0.4	2.8	8	100	79	0	0	8.8	12.2	100
	Average Time	0.04	2.02	8.80	23.62	35.44	3.76	0.037	–	–	3.81	5.14	0.99

It improves the global performance for one main reason. The intersection $\cap_i \sigma_i(I)$ is equal to the norm $\mathrm{N}_{L/K}(I)$ with high probability, and one can see in Table 4.3 that computing this intersection is way faster than computing the subsequent products. Otherwise, the number of products necessary before finding the norm seems to be more or less the same.
These observations are made over a specific family of number fields. One may wonder if the same phenomena would stay true over general extensions.

Global comparisons

Now let us compare the different general methods to compute relative norms: product of ideals as in Algorithm 18, by random elements as in Algorithm 19, and by the implementation in MAGMA.

Remark 20 (MAGMA and PARI/GP)**.** In MAGMA and PARI/GP, the representations of objects linked to a relative extension L/K correspond to the ones described by Cohen in [30]. Moreover the algorithms implemented to compute the relative norm of an ideal are also found [30]. In particular it requires computing the

pseudo-HNF of I relative to L/K, which can be time and space consuming, even if its complexity is polynomial [16]. Moreover in MAGMA, defining an ideal in a relative order $\mathcal{O}$ can be done only if $\mathcal{O}$ is a relative order over a maximal order. Therefore, in order to define I as an ideal in L/K, the process requires the computation of the maximal order of K. This can be time consuming (subexponential), thus prohibiting us from using MAGMA method over large degree number fields such as the ones studied in Chapter 5. However we will see that it behaves nicely over the small degree number fields considered above.
Thus for the method of MAGMA, we provide two timings. The first is the time taken by the function `Norm`, and the second is the time taken to compute an absolute basis of $\mathrm{N}_{L/K}(I)$ from an absolute basis of I. The last allows us to take into account the creation of the ideal structure with the function `ideal<O | · >` of MAGMA before using `Norm`, and the computation of the absolute basis after.

Real Kummer extensions First let us look at the performances of the function `Norm` of MAGMA and Algorithm 19 over the real Kummer extensions considered above. One can find results of our computations in Table 4.4.

Table 4.4: Average timings for `Norm` (MAGMA) and `RelativeNormProb` in real Kummer fields $\mathbb{Q}(\sqrt[p]{2}, \sqrt[p]{3}, \ldots, \sqrt[p]{p_r})$

(a) $p = 3$

Sequence length r	$[L:\mathbb{Q}]$	$[\tilde{L}:\mathbb{Q}]$	`Norm` (net)	`Norm` (full)	`RelativeNormProb`
2	9	18	0.011	0.065	0.005
3	27	54	7.39	8.53	0.020

(b) $p = 5$

Sequence length r	$[L:\mathbb{Q}]$	$[\tilde{L}:\mathbb{Q}]$	`Norm` (net)	`Norm` (full)	`RelativeNormProb`
2	25	100	0.089	0.220	0.031

(c) $p = 7$

Sequence length r	$[L:\mathbb{Q}]$	$[\tilde{L}:\mathbb{Q}]$	`Norm` (net)	`Norm` (full)	`RelativeNormProb`
2	49	294	1.086	1.632	0.2416

From the data gathered, the function `Norm` of MAGMA seems to be more efficient than the product method over the majority of the fields considered. However one can remark that Algorithm 18 is more efficient when using both the intersection and two-elements representation. Moreover, computing the intersection at the beginning is also faster than computing `Norm`. Thus for a non negligible proportion of ideals, it might be faster to start by this intersection to check if it is $\mathrm{N}_{L/K}(I)$. Finally `RelativeNormProb` is the most efficient procedure, independently of the field chosen.

Other extensions We now look at "random" extensions L/K with degrees in the same range than the extensions already considered. This way, it is still possible to compute $\mathcal{O}_K$ in a reasonable amount of time, and we are able to compare `Norm` and `RelativeNormProb`. We computed the time taken to compute $\mathrm{N}_{L/K}(I)$ for extension degrees $[L : K]$ in $\{3, 5, 7\}$ and random dimensions $[K : \mathbb{Q}]$ so that $[L : \mathbb{Q}] \leqslant 50$, as well as for extension degrees $[L : K]$ in $\{11, 13\}$ and random dimensions $[K : \mathbb{Q}]$ so that $[L : \mathbb{Q}] \leqslant 91$. One can find the data gathered in Tables 4.5 and 4.6.

Table 4.5: Average timings (in s) for `Norm` (MAGMA) and `RelativeNormProb` in random extensions L/K of degree less than 50

(a) $[L : K] = 3$

$[K : \mathbb{Q}]$	6	8	11	12	13	13	13	14	15	16
`Norm` (net)	0.017	0.144	0.785	0.638	1.995	0.620	0.811	1.410	2.998	5.046
`Norm` (full)	0.026	0.196	1.077	0.947	2.663	1.011	1.299	2.604	4.169	7.215
`RelativeNormProb`	0.002	0.004	0.015	0.013	0.029	0.018	0.019	0.027	0.035	0.073

(b) $[L : K] = 5$

$[K : \mathbb{Q}]$	6	7	7	8	8	9	9	9	10	10
`Norm` (net)	0.245	0.551	0.389	3.154	1.849	3.962	1.871	1.892	3.101	5.337
`Norm` (full)	0.413	0.898	0.652	7.887	2.986	6.557	3.393	3.644	6.775	8.278
`RelativeNormProb`	0.008	0.015	0.015	0.039	0.040	0.062	0.047	0.042	0.068	0.097

(c) $[L : K] = 7$

$[K : \mathbb{Q}]$	5	5	5	6	6	6	6	7	7	7
`Norm` (net)	0.612	1.167	1.113	2.428	1.239	1.223	2.622	2.704	3.266	5.594
`Norm` (full)	1.020	1.601	1.557	4.421	1.968	1.894	5.447	5.315	6.054	9.622
`RelativeNormProb`	0.014	0.025	0.025	0.040	0.034	0.033	0.044	0.062	0.060	0.080

Table 4.6: Average timings (in s) for `Norm` (MAGMA) and `RelativeNormProb` in random extensions L/K of degree less than 81

(a) $[L : K] = 11$

$[K : \mathbb{Q}]$	4	4	4	5	6	6	7	7	7	7
`Norm` (net)	2.052	2.834	4.299	11.926	18.066	18.315	52.715	105.66	33.527	69.160
`Norm` (full)	2.714	3.883	6.401	29.872	42.616	36.462	81.419	162.90	48.040	103.83
`RelativeNormProb`	0.034	0.041	0.053	0.129	0.189	0.191	0.632	0.754	0.352	0.627

(b) $[L : K] = 13$

$[K : \mathbb{Q}]$	4	4	5	5	5	5	6	6	6	6
`Norm` (net)	5.787	6.321	44.520	37.906	31.769	30.503	46.335	50.036	96.253	45.768
`Norm` (full)	8.223	9.184	74.145	61.903	43.387	42.274	64.560	71.803	140.93	92.144
`RelativeNormProb`	0.067	0.074	0.347	0.313	0.294	0.292	0.449	0.433	0.797	0.488

One can remark that the function `Norm` of MAGMA is more influenced by the dimension. The computation of $\mathrm{N}_{L/K}(I)$ increases a lot with $[L : \mathbb{Q}]$. Moreover its performance can vary from one extension to another with the same parameters. Finally, even if the time taken by `RelativeNormProb` to compute relative norms increases also when the dimension of the extension increases, it is still very efficient compared to `Norm` with average timings smaller than 1 second.

4.1.4 Conclusion

In this section we studied two methods to compute relative norms of ideals in number field extensions L/K. The first, corresponding to Algorithm 18, computes the norm as the product $\prod_\sigma \sigma(I)$ with σ running through $\mathrm{Hom}(L/K, \mathbb{C})$. Over Kummer extensions this method is proved to be polynomial. Moreover we showed that using the determinant, one can check during the computation if the norm has already been reached. Experimental data show that this offers great speed-ups. Despite these facts, this method can still be quite heavy because it requires computations in the Galois closure of L/K, which leads to large matrices to handle. The second method that we explored is described by Algorithm 19. This method is only heuristic and probabilistic. However it behaves very well in practice, and outperforms greatly both our first method and the implementation of MAGMA [22].

In future work, one could prove the probabilistic complexity of Algorithm 19.

4.2 Roots of a polynomial

We are interested in the following problem. Given K a number field and $f(X) \in K[X]$, find the roots of $f(X)$ in K i.e. find $Z_K(f) = \{x \in K \mid f(x) = 0\}$. In fact we will only consider the cases such that all the roots can be expressed as integral combinations of a known basis of K i.e.

$$\exists (b_i)_i \in K^n, \forall x \in Z_K(f), \exists (x_i)_i \in \mathbb{Z}^n \mid x = x_1 b_1 + \cdots + x_n b_n.$$

First let us describe quickly how the standard computation is done. We will call it the *algebraic method* or *standard method*. It follows the same ideas presented in [40, 7]. The procedure is as follows.

1. Pick a prime ideal $\mathfrak{p}$ which defines an isomorphism between $\mathcal{O}_K/\mathfrak{p}$ and $\mathbb{F}_{p^s}$ for some prime p.

2. Determine the roots modulo $\mathfrak{p}$ by factorising $f(X)$ in $\mathbb{F}_p[X]$.

3. Lift these to the solutions modulo a power $\mathfrak{p}^k$.

4. If $\mathfrak{p}^k$ is large enough compare to the size of the solutions, and if $\mathfrak{p}^k$ is given by a LLL-reduced basis then we can recover a solution x given $y \equiv x \bmod \mathfrak{p}^k$.

The last step is usually done by using Babaï's rounding technique [7, 40]. Finally one can see that this technique can be described by three main steps.

1. Compute first "approximations" of the solutions.
2. Compute a reduced basis of a lattice.
3. Retrieve the solutions from the approximations using the lattice.

Our method follows these three steps, but uses complex embeddings instead of prime ideals.

4.2.1 A simple algorithm

Our method is pretty simple. First let us describe the part related to steps 2 and 3.

Decoding through LLL

First let us fix the decoding problem we will be interested in. Let K be a number field, and $\mathcal{B} = (b_1, \dots, b_n)$ be a $\mathbb{Q}$-basis of K. Now consider $x \in K$ such that $x \in \mathbb{Z}[\mathcal{B}]$. Our problem is the following: *«Given an approximation of $\sigma_i(x)$ for some $i \in [\![1, n]\!]$, retrieve x ».*

Decoding with approximations Our decoding method can be directly linked to the original paper describing the LLL algorithm [63]. A. Lenstra, H. Lenstra and Lovasz mention several applications of their lattice reduction algorithm, such as finding short integral relations between algebraic numbers or finding the irreducible polynomial of an element in a number field. As an example, assume one knows approximations of real numbers $\alpha_1, \dots, \alpha_n$. Now define the embedding

$$\begin{array}{ccc} \mathbb{Z}^n & \longrightarrow & \mathbb{R}^{n+1} \\ (\lambda_i)_{i \in [\![1,n]\!]} & \longmapsto & (C \sum_{i=1}^{n} \lambda_i \alpha_i, \lambda_1, \dots, \lambda_n) \end{array}$$

which gives a lattice of $\mathbb{R}^{n+1}$ represented by the matrix

$$\begin{bmatrix} C\alpha_1 & 1 & 0 & \dots & 0 \\ C\alpha_2 & 0 & 1 & \ddots & \vdots \\ \vdots & \vdots & \ddots & \ddots & 0 \\ C\alpha_n & 0 & \dots & 0 & 1 \end{bmatrix}.$$

Here C is a large coefficient used to ensure the shortness of the solution. Reducing this basis would allow us to retrieve a short vector $(\lambda_i)_{i \in [\![1,n]\!]}$ such that $C \sum_{i=1}^{n} \lambda_i \alpha_i$

is small. Similarly it is common to use LLL algorithm to solve knapsack problems. Assume we are given $(\alpha_1, \ldots, \alpha_n) \in \mathbb{Z}^n$ and $S \in \mathbb{Z}$, and that we want to find a short vector $(\lambda_1, \ldots, \lambda_n)$ such that $\sum_{i=1}^{n} \lambda_i \alpha_i = S$. Then one can consider the similar matrix

$$\begin{bmatrix} -C\alpha_1 & 1 & 0 & \ldots & 0 \\ -C\alpha_2 & 0 & 1 & \ddots & \vdots \\ \vdots & \vdots & \ddots & \ddots & 0 \\ -C\alpha_n & 0 & \ldots & 0 & 1 \\ CS & 0 & \ldots & \ldots & 0 \end{bmatrix}.$$

Again a LLL reduction would yield a small solution of the linear equation. One can combine both approaches to be able to retrieve the coefficient of $x \in \mathbb{Z}[\mathcal{B}]$ from an embedding $\sigma_i(x)$. For all $j \in [\![1, n]\!]$, denote by β_j an approximation of $\sigma_i(b_j)$ and S an approximation if $\sigma_i(x)$. Now consider $(x_i)_{i \in [\![1,n]\!]} \in \mathbb{Z}^n$ the coefficients of x in the basis $\mathcal{B}$. Then one can expect $S - \sum_{i=1}^{n} x_i \beta_i$ to be close to 0 (or at least small). Thus reducing the matrix

$$\begin{bmatrix} -C\beta_1 & 1 & 0 & \ldots & 0 \\ -C\beta_2 & 0 & 1 & \ddots & \vdots \\ \vdots & \vdots & \ddots & \ddots & 0 \\ -C\beta_n & 0 & \ldots & 0 & 1 \\ CS & 0 & \ldots & \ldots & 0 \end{bmatrix}$$

is expected to yield the solution for well-chosen parameters, i.e. high enough precision for the approximations and large enough C. However this does not ensure finding the searched for vector, and if one needs to retrieve several elements, it might be time consuming to reduce such a matrix for each element. Thus we chose to adapt this approach as follows.

Our technique Before we explain how to use the lattices mentioned before to retrieve coefficients from approximations or a complex embedding we need to fix some objects and notations.

Definition 4.2. If x is in $\mathbb{R}$, the *approximation of x up to l-bits* for $l \in \mathbb{N}$ is the integer $\lfloor 2^l x \rceil$. If $x \in \mathbb{C}$ then its *approximation up to l-bits* will be $\lfloor 2^l \Re(x) \rceil + i \lfloor 2^l \Im(z) \rceil$. In both cases it will be denoted by $[x]_l$.

Remark 21. We commonly identify a complex number x with the pair of real numbers $(\Re(x), \Im(x))$. We extend this identification to approximations. In particular, this is true when such elements are presented in integral matrices.

Definition 4.3. Consider a number field K, $\mathcal{B}$ a $\mathbb{Q}$-basis of K, $\sigma \in \mathrm{Hom}(K, \mathbb{C})$, and $l \in \mathbb{N}$. We will call a *basis lattice of K up to precision l relative to $\mathcal{B}$ and σ* and denote by $\mathcal{L}(\mathcal{B}, \sigma, l)$ the lattice generated by the matrix $B(\mathcal{B}, \sigma, l)$ defined as follows:

$$B(\mathcal{B}, \sigma, l) = \begin{bmatrix} -[\sigma(b_1)]_l & C & 0 & \dots & 0 \\ -[\sigma(b_2)]_l & 0 & C & \ddots & \vdots \\ \vdots & \vdots & \ddots & \ddots & 0 \\ -[\sigma(b_n)]_l & 0 & \dots & 0 & C \end{bmatrix} \tag{4.1}$$

The matrix $B(\mathcal{B}, \sigma, l)$ will be called the *basis matrix* of $\mathcal{L}(\mathcal{B}, \sigma, l)$.

Remark 22.
- When there is no ambiguity regarding K, $\mathcal{B}$ or σ, we will simply denote $\mathcal{L}(\mathcal{B}, \sigma, l)$ by $\mathcal{L}_l$ and call it the *lattice basis of K up to precision l*. Similarly the matrix $B(\mathcal{B}, \sigma, l)$ will be written B_l.
- The constant C is used to ensure the validity of the decoding, and accelerate the reduction algorithm on B_l.
- If the embedding σ is not a real embedding, then the first column of the matrix in Equation 4.1 is in fact two columns, containing respectively the real and imaginary parts of the corresponding complex numbers.

Notation. If $\sigma(K) \subset \mathbb{R}$, we will denote by $\sigma(\mathcal{B})_l$ the first column vector of a basis matrix $B(\mathcal{B}, \sigma, l)$. If $\sigma(K) \not\subset \mathbb{R}$, we will write $\sigma(\mathcal{B})_l = [\Re(\sigma(\mathcal{B})_l), \Im(\sigma(\mathcal{B})_l)]$ for the matrix composed of the two first column vectors of $B(\mathcal{B}, \sigma, l)$.

As before, consider K a number field, $\mathcal{B} = (b_1, \dots, b_n)$ a $\mathbb{Q}$-basis of K, and $x \in K$ such that there is $(x_i)_{i \in [\![1,n]\!]} \in \mathbb{Z}^n$ with $x = x_1 b_1 + \cdots + x_n b_n$. Now assume that $[\sigma(x)]_l$ is known for some $\sigma \in \operatorname{Hom}(K, \mathbb{C})$ and $l \in \mathbb{N}$. Then one can use Kannan's embedding technique using $\mathcal{L}(\mathcal{B}, i, l)$ and $t = ([\sigma(x)]_l, 0, \dots, 0)$ to obtain the same result as Babaï's nearest plane algorithm. It is also better to reduce $B(\mathcal{B}, \sigma, l)$ with LLL first, and use Algorithm 9 with input this reduced basis and t. This leads to Algorithm 21.

Notation. Given K a number field, $\mathcal{B}$ a $\mathbb{Q}$-basis of K, $\sigma \in \operatorname{Hom}(K, \mathbb{C})$ and a precision l, we will denote by $L(\mathcal{B}, \sigma, l)$ the LLL-reduced basis of $B(\mathcal{B}, \sigma, l)$. Similarly, we will write L_l when there is no ambiguity.

The output of Algorithm 21 is a vector of coefficients which are expected to be the coefficients of x in the basis $\mathcal{B}$. The correctness of the outcome will depend on the parameters chosen, i.e. the precision l and the constant C.

Algorithm 21 `TestDecode`

```
Require: An integer $l$, the matrix $L_l$ of a reduced basis of $\mathcal{L}(\mathcal{B}, \sigma, l)$, an integer $[\sigma(x)]_l$ for some $x \in K$ and a coefficient and a coefficient $M$.
Ensure: A candidate $y = (y_1, \dots, y_n)$ for the vector of coefficients of $x$ expressed in $\mathcal{B}$.
1: $t \leftarrow ([\sigma_i(x)]_l, 0, \dots, 0) \in \mathbb{Z}^{n+1}$
2: return Kannan($L_l, t, M$)$/C$                                  ▷ Alg. 9
```

Choice of parameters and correctness of the method Let us now determine which parameters to choose. First one can determine the determinant of the lattice basis of K used to decode.

Let us state a classical result that we will use.

Lemma 4.1 (Matrix determinant lemma [44])**.** *Let A be a ring, $M \in \mathrm{M}_n(A)$ be an invertible matrix and $U, V \in \mathrm{M}_{n,m}(A)$. Then the following is true:*

$$\det(M + UV^{\mathsf{T}}) = \left(\mathrm{Id}_m + V^{\mathsf{T}} M^{-1} U\right) \det(M). \tag{4.2}$$

Lemma 4.2. *Let K be a number field and $\mathcal{L}_l = \mathcal{L}(\mathcal{B}, \sigma, l)$ be a basis lattice of K. Also let m be an integer equal to 1 if σ is real, and 2 otherwise. Then*

$$\mathrm{vol}(\mathcal{L}_l)^2 = C^{2n} \det\left(\mathrm{Id}_m + \frac{1}{C^2} \sigma(\mathcal{B})_l^{\mathsf{T}} \sigma(\mathcal{B})_l\right) \tag{4.3}$$

Proof. By definition $\mathrm{vol}(\mathcal{L}_l)^2$ is the determinant of the matrix $B(\mathcal{B}, \sigma, l) B(\mathcal{B}, \sigma, l)^{\mathsf{T}} = C^2 \mathrm{Id}_n + \sigma(\mathcal{B})_l \times \sigma(\mathcal{B})_l^{\mathsf{T}}$. Then by Lemma 4.1 one has

$$\mathrm{vol}(\mathcal{L}_l)^2 = \det\left(\mathrm{Id}_m + \sigma(\mathcal{B})_l^{\mathsf{T}} \times \frac{1}{C^2} \mathrm{Id}_n \times \sigma(\mathcal{B})_l\right) \det(C^2 \mathrm{Id}_n)$$

which gives the claimed identity. □

Then one can deduce from Lemma 4.2 a lower bound on the volume of $\mathcal{L}(\mathcal{B}, \sigma, l)$ which depends on the precision l and the size of $\sigma(\mathcal{B})$.

Notation. Consider a number field K, $\mathcal{B}$ a $\mathbb{Q}$-basis of K, $\sigma \in \mathrm{Hom}(K, \mathbb{C})$ and $l \in \mathbb{N}$. Let us define the value $\Delta(\mathcal{B}, \sigma, l)$. If $\sigma(K) \subset \mathbb{R}$ we set

$$\Delta(\mathcal{B}, \sigma, l) = \|\sigma(\mathcal{B})\|_2^2 - \frac{\|\sigma(\mathcal{B})\|_1}{2^l},$$

and if $\sigma(K) \not\subset \mathbb{R}$ we set

$$\Delta(\mathcal{B}, \sigma, l) = \|\Re(\sigma(\mathcal{B}))\|_2^2 + \|\Im(\sigma(\mathcal{B}))\|_2^2 - \frac{\|\Re(\sigma(\mathcal{B}))\|_1}{2^l} - \frac{\|\Im(\sigma(\mathcal{B}))\|_1}{2^l}.$$

Proposition 4.3. *Let K be a number field and $\mathcal{L}_l = \mathcal{L}(\mathcal{B}, \sigma, l)$ be a basis lattice of K. Also let m be an integer equal to 1 if σ is real, and 2 otherwise. Then the following is true,*

$$\mathrm{vol}(\mathcal{L}_l)^2 \geqslant C^{2n}\left(1 + \frac{2^{2l}}{C^2}\Delta(\mathcal{B}, \sigma, l)\right). \tag{4.4}$$

Proof. From Equation 4.3 we can write for $m = 1$

$$\mathrm{vol}(\mathcal{L}_l)^2 = C^{2n}\left(1 + \frac{1}{C^2}\left\|\sigma(\mathcal{B})_l\right\|_2^2\right) = C^{2n}\left(1 + \frac{1}{C^2}\sum_{i=1}^{n}[\sigma(b_i)]_l^2\right).$$

Then for each $i \in [\![1, n]\!]$, there is $\epsilon_i \in [-\frac{1}{2}, \frac{1}{2}]$ such that $[\sigma(b_i)]_l = 2^l\sigma(b_i) + \epsilon_i$. Thus we obtain

$$\begin{aligned}\mathrm{vol}(\mathcal{L}_l)^2 &= C^{2n}\left(1 + \frac{1}{C^2}\sum_{i=1}^{n}(2^l\sigma(b_i) + \epsilon_i)^2\right)\\ &= C^{2n}\left(1 + \frac{2^{2l}}{C^2}\sum_{i=1}^{n}(\sigma(b_i)^2 + 2\sigma(b_i)\frac{\epsilon_i}{2^2} + \frac{\epsilon_i^2}{2^{2l}})\right).\end{aligned}$$

Since for all $i \in [\![1, n]\!]$, one has $|\epsilon_i| \leqslant \frac{1}{2}$, we obtain the inequality

$$\mathrm{vol}(\mathcal{L}_l)^2 \geqslant C^{2n}\left(1 + \frac{2^{2l}}{C^2}\sum_{i=1}^{n}(\sigma(b_i)^2 - 2 \times \frac{1}{2} \times \frac{|\sigma(b_i)|}{2^l})\right).$$

If $\sigma(K) \not\subset \mathbb{R}$ then from Equation 4.3 we have

$$\mathrm{vol}(\mathcal{L}_l)^2 = C^{2n}\det\left(\mathrm{Id}_2 + \frac{1}{C^2}\sigma(\mathcal{B})_l^{\mathsf{T}}\sigma(\mathcal{B})_l\right).$$

If we write $M = \mathrm{Id}_2 + \frac{1}{C^2}\sigma(\mathcal{B})_l^{\mathsf{T}}\sigma(\mathcal{B})_l$, then

$$M = \begin{bmatrix} 1 + \dfrac{\|\Re(\sigma(\mathcal{B})_l)\|_2^2}{C^2} & \dfrac{(\Re(\sigma(\mathcal{B})_l) \mid \Im(\sigma(\mathcal{B})_l))}{C^2} \\ \dfrac{(\Re(\sigma(\mathcal{B})_l) \mid \Im(\sigma(\mathcal{B})_l))}{C^2} & 1 + \dfrac{\|\Im(\sigma(\mathcal{B})_l)\|_2^2}{C^2} \end{bmatrix}$$

so we get

$$\begin{aligned}\det(M) =& 1 + \frac{\|\Re(\sigma(\mathcal{B})_l)\|_2^2}{C^2} + \frac{\|\Im(\sigma(\mathcal{B})_l)\|_2^2}{C^2} + \frac{\|\Re(\sigma(\mathcal{B})_l)\|_2^2\,\|\Im(\sigma(\mathcal{B})_l)\|_2^2}{C^4}\\ &- \frac{(\Re(\sigma(\mathcal{B})_l) \mid \Im(\sigma(\mathcal{B})_l))^2}{C^4}.\end{aligned}$$

By Cauchy-Schwarz inequality we have

$$\frac{\|\Re(\sigma(\mathcal{B})_l)\|_2^2\,\|\Im(\sigma(\mathcal{B})_l)\|_2^2}{C^4} - \frac{(\Re(\sigma(\mathcal{B})_l) \mid \Im(\sigma(\mathcal{B})_l))^2}{C^4} \geqslant 0,$$

therefore

$$\mathrm{vol}(\mathcal{L}_l)^2 \geqslant C^{2n}\left(1 + \frac{\|\Re(\sigma(\mathcal{B})_l)\|_2^2}{C^2} + \frac{\|\Im(\sigma(\mathcal{B})_l)\|_2^2}{C^2}\right).$$

Following the same reasoning that we did for the real case, we can conclude that

$$\frac{\|\Re(\sigma(\mathcal{B})_l)\|_2^2}{C^2} \geqslant \frac{2^{2l}}{C^2}\left(\|\Re(\sigma(\mathcal{B}))\|_2^2 - \frac{\|\Re(\sigma(\mathcal{B}))\|_1}{2^l}\right)$$

and

$$\frac{\|\Im(\sigma(\mathcal{B})_l)\|_2^2}{C^2} \geqslant \frac{2^{2l}}{C^2}\left(\|\Im(\sigma(\mathcal{B}))\|_2^2 - \frac{\|\Im(\sigma(\mathcal{B}))\|_1}{2^l}\right),$$

which gives the desired result. □

It is now possible to certify the correctness of the decoding. As Proposition 2.3 states, the output is known to be correct if the distance between the target and the lattice is smaller than $\frac{1}{2}\min\{\left\|\tilde{b}_i\right\| \mid i \in [\![1, r]\!]\}$.

Proposition 4.4. *Consider K a number field, $\mathcal{L}_l = \mathcal{L}(\mathcal{B}, \sigma, l)$ a basis matrix of K, and $x \in \mathbb{Z}[\mathcal{B}]$. Let m be an integer such that $m = 1$ if $\sigma(K) \subset \mathbb{R}$ and $m = 2$ otherwise. Then Algorithm 21 outputs the correct vector of coefficients $(x_1, \dots, x_n)$ of x in $\mathcal{B}$ if the following holds:*

$$\frac{2^{m-1}}{4}\left(1 + 2\|x\|_1 + n\|x\|_2^2\right) + C^2\|x\|_2^2 \leqslant \frac{\lambda_1(\mathcal{L}_l)^2}{2^{2n}}. \tag{4.5}$$

Proof. In `TestDecode` we wish to solve the BDD with input $\mathcal{L}(\mathcal{B}, \sigma, l)$ and target vector $t = ([\sigma(x)]_l, 0, \dots, 0)$. The vector v in $\mathcal{L}_l$ which is assumed to be the closest to t is $v = (\sum_{i=1}^n x_i[\sigma(b_i)], -Cx_1, \dots, -Cx_n)$. The error vector is then

$$e = \left([\sigma(x)]_l - \sum_{i=1}^n x_i[\sigma(b_i)], Cx_1, \dots, Cx_n\right).$$

Now let us consider the case where $m = 1$ and look at the first coordinate of e. First write $(\eta, \epsilon_1, \dots, \epsilon_n) \in [-\frac{1}{2}, \frac{1}{2}]^{n+1}$ the vector of errors due to the approximations, i.e. $[\sigma(x)]_l = 2^l\sigma(x) + \eta$ and for all $i \in [\![1, n]\!]$, $[\sigma(b_i)]_l = 2^l\sigma(b_i) + \epsilon_i$. Then we have

$$e_1 = 2^l\sigma(x) + \eta - \sum_{i=1}^n 2^l x_i\sigma(b_i) + \epsilon_i = \eta - \sum_{i=1}^n x_i\epsilon_i$$

which gives

$$e_1^2 \leqslant \frac{1}{4}\left(1 + \sum_{i=1}^n |x_i|\right)^2 = \frac{1}{4}\left(1 + 2\|x\|_1 + n\|x\|_2^2\right).$$

If $m = 2$, or equivalently $\sigma(K) \not\subset \mathbb{R}$, one needs to consider the real and imaginary parts. Thus we get $e \in \mathbb{R}^{n+2}$ and $(\eta, \epsilon_1, \dots, \epsilon_n) \in [-\frac{1}{2}, \frac{1}{2}]^{n+2}$ with $\eta = (\eta_1, \eta_2)$.

Following the previous analysis we obtain

$$e_1^2 \leqslant \frac{2}{4}\left(1+\sum_{i=1}^{n}|x_i|\right)^2 = \frac{1}{4}\left(1+2\left\|x\right\|_1+n\left\|x\right\|_2^2\right).$$

Thus we obtain the following upper bound for $\|e\|_2^2$:

$$\|e\|_2^2 \leqslant \frac{2^{m-1}}{4}\left(1+2\left\|x\right\|_1+n\left\|x\right\|_2^2\right)+C^2\left\|x\right\|_2^2.$$

Finally one has to remark that a LLL-reduced basis $(b'_1,\ldots,b'_n)$ of a lattice $\mathcal{L}$ (with $\delta=\frac{3}{4}$) satisfies

$$\left\|\tilde{b}_i\right\|_2^2 \geqslant \frac{\left\|\tilde{b}_1\right\|_2^2}{2^{n-1}} \geqslant \frac{\lambda_1(\mathcal{L})}{2^{n-1}}.$$

□

Remark 23.

- One can also use an expression involving only $\|x\|_2$ with

$$\frac{1}{4}\left(1+2\left\|x\right\|_2^2+n\left\|x\right\|_2^2\right)+C^2\left\|x\right\|_2^2 \leqslant \frac{\lambda_1(\mathcal{L}_l)^2}{2^{2n}}.$$

- If we know a priori that $\|x\|_\infty < M$ for some M, then the inequality can be written

$$\frac{1}{4}\left(1+2nM+n^2M\right)+C^2n^2M^2 \leqslant \frac{\lambda_1(\mathcal{L}_l)^2}{2^{2n}}.$$

Now if one has a lower bound for $\lambda_1(\mathcal{L}_l)$ depending on the parameters, one can deduce from it a condition for the correctness of the output of `TestDecode`. In particular it is possible to obtain a lower bound of the precision l for which the computation is correct. The Gaussian heuristic provides an *estimation* of $\lambda_1(\mathcal{L}_l)$, which can be used to obtain a *heuristic* condition for the correctness of the algorithm, as in Theorem 4.1. However it could be that the considered lattices $\mathcal{L}_l$ are special lattices such that their shortest vectors are way shorter than predicted by the Gaussian heuristic.

Theorem 4.1 (Correctness of decoding)**.** *Consider K a number field, $\mathcal{B}$ a $\mathbb{Q}$-basis of K and $x \in \mathbb{Z}[\mathcal{B}]$. Additionally fix $\sigma \in \mathrm{Hom}(K,\mathbb{C})$, $C \in \mathbb{N}$, and $m \in \mathbb{N}$ such that $m=1$ if $\sigma(K)\subset\mathbb{R}$ and $m=2$ otherwise. If $l\in\mathbb{N}$ with $l \geqslant 1$ satisfies*

$$\begin{aligned} 2l \geqslant\ & n\frac{(2(n-1)+(m-1))\ln(2)+\ln(\|x\|_2^2\left(2+n+4C^2\right)+1)}{\ln(2)} \\ & -\frac{n\ln(\frac{n}{2\pi e})+2(n-1)\ln C+\ln\left(\Delta(\mathcal{B},\sigma,1)\right)}{\ln(2)} \end{aligned} \tag{4.6}$$

then Algorithm 21 outputs the vector of coefficients of x in the basis $\mathcal{B}$.

Proof. Let us fix $l \in \mathbb{N}$. The Gaussian heuristic states $\lambda_1(\mathcal{L}) \sim \sqrt{\frac{n}{2\pi e}} \det \mathcal{L}^{1/n}$ Then one can combine Equation (4.4) to obtain a conditional inequality. In order to simplify the expression, we will express the above inequality using only $\|x\|_2$ as mentioned above. This leads to the following inequality:

$$\frac{2^{m-1}}{4}(\|x\|_2^2(2+n+4C^2)+1) \leqslant \frac{1}{2^{2n}}\left(\frac{n}{2\pi e}C^2\sqrt[n]{\left(1+\frac{2^{2l}}{C^2}(\Delta(\mathcal{B},\sigma,l))\right)}\right).$$

Remark that $l \geqslant 1$ so we have $\Delta(\mathcal{B},\sigma,l) \geqslant \Delta(\mathcal{B},\sigma,1)$. With a bit more work we can obtain the condition

$$2^{2(n-1)+(m-1)}(\|x\|_2^2(2+n+4C^2)+1) \leqslant \frac{n}{2\pi e}C^2\sqrt[n]{\left(\frac{2^{2l}}{C^2}(\Delta(\mathcal{B},\sigma,1))\right)}. \tag{4.7}$$

In order to obtain an inequality involving l directly (instead of 2^l) we will apply the logarithm map. The left side of Equation (4.7) gives $(2(n-1)+(m-1))\ln(2) + \ln(\|x\|_2^2(2+n+4C^2)+1)$ while the right side gives

$$\frac{1}{n}\times\left(n\ln(\frac{n}{2\pi e}) + 2(n-1)\ln C + 2l\ln 2 + \ln\left(\Delta(\mathcal{B},\sigma,1)\right)\right).$$

Putting everything together gives the claimed condition. □

One can obviously simplify the expression of the condition given in Equation (4.6) by removing terms which are not asymptotically relevant. Moreover it can also be simplified by specifying C. Indeed we can find C such that the right-hand side of Equation (4.6) is minimal.

Proposition 4.5. *The right-hand side of Equation (4.6) is minimal when*

$$C = \frac{1}{4}\left((n-1)(2+n)+\frac{1}{\|x\|_2^2}\right).$$

Remark 24. Remark that the precision l given by Theorem 4.1 is polynomial in $\ln\|x\|_2^2$ and the dimension n. Therefore the complexity of computing a LLL-reduced basis of $\mathcal{L}(\mathcal{B},\sigma,l)$ is also polynomial in $\ln\|x\|_2^2$ and n, as well as `Kannan`. Therefore, provided one is able to obtain an approximation $[\sigma(x)]_l$ for l high enough, it is possible to recover the coefficients of x in polynomial time using Algorithm 21.

In order to estimate the correctness of the value given by Theorem 4.1, we verified if the Gaussian heuristic holds for lattices $\mathcal{L}_l$. We computed the average value of the quotient $\lambda_1(\mathcal{L}_l)/\lambda_1(\mathcal{L}_l)_{\text{gauss}}$ for increasing values of l, over random number fields K with fixed degrees. The results can be found in Figure 4.1. One can see that $\lambda_1(\mathcal{L}_l)$ is very close to the value predicted by the Gaussian heuristic. Moreover the two values are getting closer when the precision l or the degree n are increasing. This shows that one can safely consider that Theorem 4.1 provides a good value certifying the correctness of the output of `TestDecode`.

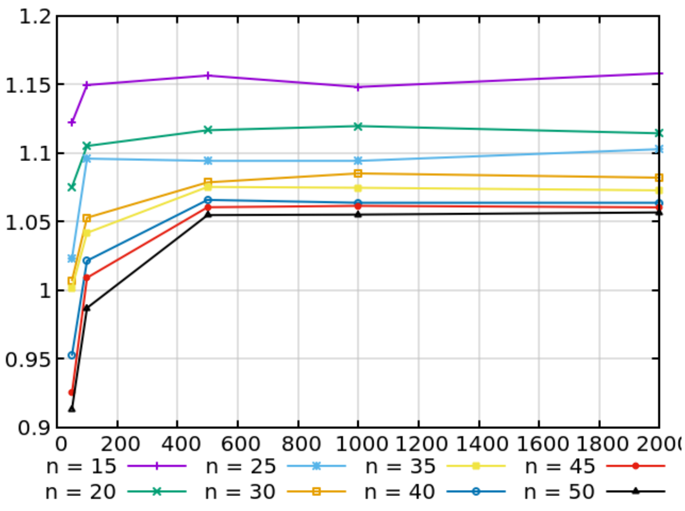


Figure 4.1: Average value of $\lambda_1(\mathcal{L}_l)/\lambda_1(\mathcal{L}_l)_{\text{gauss}}$ plotted against the precision l, for several $n = [K : \mathbb{Q}]$.

Heuristic precision While the precision given by Equation (4.6) allowsus to certify the correctness and the polynomial complexity of the method we described, it is possible to find a better one by experiments. It can be useful to know better experimental bounds to hasten the computation.

First, in some cases, one is happy with obtaining the result with some probability high enough. It is typically the case in cryptanalysis. Therefore if the probability that an element can be decoded with a smaller precision is high enough, then it is completely acceptable to use such a value which reduce the computing time. Secondly, one can have access to a way of verifying that the output of the decoding is correct. It is typically the case when one is looking for the roots of a polynomial $f(X)$. Once we get a candidate solution x by `TestDecode`, we can check if $f(x) = 0$. If not we can increase the precision until a solution is found. Such a strategy can be globally more efficient, especially if one has to decode several elements and that the probability that a smaller precision is sufficient to decode elements is high enough.

In order to obtain an experimental sufficient precision to decode, we did as follows. Fix K a number field given by an irreducible polynomial $P(X) \in \mathbb{Z}[X]$. Given $\mathcal{B} = (b_1, \ldots, b_n)$ a $\mathbb{Q}$-basis of K, we generated random elements $x \in \mathbb{Z}[\mathcal{B}]$ such that for all $i \in [\![1, n]\!], x_i \in [\![-2^s, 2^s]\!]$ for some s. Then increasing precision l (following an arithmetic progression) we computed $[\sigma(x)]_l$ and used Algorithm 21 until x was retrieved. Then we computed the quotient of the final precision by $l_t = [K : \mathbb{Q}]\frac{\ln\|x\|_2^2}{2\ln(2)}$.

We did such tests making several parameters varying: s, $[K : \mathbb{Q}]$ and the bit-size of the coefficients of $P(X)$.

We chose to test the experimental precision against l_t because when K and $\mathcal{B}$ are fixed, it is the term of Equation 4.6 which is asymptotically relevant. Moreover the algebraic method of Belabas [7] requires the norm $\mathrm{N}_{K/\mathbb{Q}}(\mathfrak{p}^k)$ to be greater than a value which is essentially l_t, so that the decoding is certified. Finally, this value was also suggested by experiments we did when we first used our technique in [64] to compute cube roots in multicubic fields. When comparing this value to (4.6), we can remark that we gain essentially $n(n - \frac{\ln(n)}{\ln(2)})$.

Increasing the precision As mentioned, one might want or need to increase the precision to which computations are performed. Recall that the heavier task consistsof finding a reduced basis of $\mathcal{L}(\mathcal{B}, \sigma, l)$. Therefore when increasing the precision, say from l to l', one has to reduce $\mathcal{L}(\mathcal{B}, \sigma, l')$. One can use the fact that $\mathcal{L}(\mathcal{B}, \sigma, l)$ has already been computed to speed-up the computation of $L_{l'}$ in the following way. When computing L_l, one can also compute $U_l \in \mathrm{GL}_n(\mathbb{Z})$ such that $U_l B_l = L_l$. Then, instead of applying directly LLL to $B_{l'}$, one can first multiply $B_{l'}$ by U_l then use LLL. This is expected to reduce the most significant bits of $B_{l'}$, thus accelerating the final LLL. With a generic matrix, the downside would be that computing the transformation matrix U_l together with the LLL-reduced matrix is slower than computing the reduced matrix alone. However the shape of B allows us to retrieve U_l directly from L_l. Indeed, one has $L_l = U_l B_l = U_l\,[\sigma(\mathcal{B})_l \mid C\mathrm{Id}_n] = [U_l \sigma(\mathcal{B})_l \mid CU_l]$.

Algorithm 22 `LatticeBasisUpdate`

```
Require: A number field $K$, $\mathcal{B}$ a $\mathbb{Q}$-basis of $K$, $L_l$ and $U_l$ such that $L_l = U_l B_l = \mathrm{LLL}(B_l)$, $l' > l$.
Ensure: $L_{l'}, U_{l'}$
1: $B \leftarrow B_{l'}$
2: $B \leftarrow U_l B$
3: $L, U \leftarrow \mathrm{LLL}(B)$
4: return $L, U$
```

Computing roots

We can easily imagine now how to compute roots of a polynomial $f(X) \in K[X]$. We described which lattice we use, and the decoding method. Obviously the reduced matrix $L(\mathcal{B}, \sigma, l)$ is computed once and used to retrieve all roots. Therefore we have the following main steps:

1. fix an embedding $\sigma : K \hookrightarrow \mathbb{C}$;

2. compute a precision $l \in \mathbb{N}$ allowing the decoding to be correct;
3. compute a LLL-reduced basis L_l of the lattice generated by the basis of K;
4. using σ, compute approximations of the roots of $f(X)$ up to a precision l;
5. use L_l to retrieve the roots of $f(X)$ using `TestDecode`.

We will call `PrecisionEvaluation` the function returning the needed precision for input $f(X) \in K[X]$. Following Theorem 4.1, it depends on the Euclidean norms of the roots of $f(X)$. In order to evaluate an upper bound of these norms, we can follow [7]. We will denote by `FloatPolynomialRoots` the procedure computing the real (resp. complex roots) of a real (resp. complex) polynomial. We obtain Algorithm 23 describing the method we implemented to compute $Z_K(f)$.

Algorithm 23 `PolynomialRoots`

```
Require: A number field $K$, $f(X) \in K[X]$, $\mathcal{B}$ a $\mathbb{Q}$-basis of $K$ such that $Z_K(f) \in \mathbb{Z}[\mathcal{B}]$.
Ensure: The set $Z_K(f)$
 1: $\sigma \leftarrow$ ChooseEmbedding($K$)
 2: $l \leftarrow$ PrecisionEvaluation($f(X)$)
 3: $L_l \leftarrow \mathrm{LLL}(B(\mathcal{B}, \sigma, l))$
 4: $Z \leftarrow$ FloatPolynomialRoots($f^\sigma, l$)
 5: $S \leftarrow \emptyset$
 6: for $z \in Z$ do
 7:     $y \leftarrow$ TestDecode($L_l, z$)
 8:     if $f(y) = 0$ then
 9:         $S \leftarrow S \cup \{y\}$
10:     end if
11: end for
12: return $S$
```

Following the results from the previous section, one can state the following theorem.

Theorem 4.2. *Consider a number field K, $\mathcal{B}$ a $\mathbb{Q}$-basis of K and $f(X) \in K[X]$ such that $Z_K(f) \subset \mathbb{Z}[\mathcal{B}]$. Then for input $(K, \mathcal{B}, f(X))$, Algorithm 23 outputs $Z_K(f)$ in polynomial time, under the heuristic that states that basis lattices satisfy the Gaussian heuristic.*

Norm of the roots Fix the factorisation of $f(X)$ over K as $f(X) = g(X)h(X)$ such $Z_K(g) = Z_K(f)$, $Z_K(h) = \emptyset$.
As mentioned we can follow [7] to bound $\|x\|_2^2$ for $x \in Z_K(f)$ given $f(X)$. One does essentially as follows.

1. Find B_{f,T_2} such $\left\|\sigma_{K/\mathbb{Q}}(x)\right\|_2^2 \leqslant B_f$ for all $x \in Z_K(f)$.

2. Compute $B_{\sigma_K^{-1}}$ the matrix norm of σ_K^{-1}, expressed relatively to the canonical basis of $\sigma_{K/\mathbb{Q}}(K)$ and $\mathcal{B}$.

3. The value $B_f = B_{f,T_2} B^2_{\sigma_K^{-1}}$ satisfies the desired property.

This bound can be quite large compared to $\sup\{\|x\|_2^2 \mid x \in Z_K(f)\}$. It is typically the case when $\|g\| \ll \|h\|$. Indeed, if this the case, B_f will essentially bound the complex roots of $h(X)$. Without extra-information this is the best one can do. If one does not need to find all the roots, a general strategy can be to compute several bounds corresponding to increasing norms. This way if $f(X)$ is of the form mentioned, it is possible to retrieve some of the roots faster. Moreover one can use heuristic evaluations for $\|x\|_2^2$, giving smaller results than B_f. Let us explain how we do it. Consider $\sigma \in \mathrm{Hom}(K, \mathbb{C})$ and $x \in K$. Then one has

$$|\sigma(x)|^2 = \left|\sum_{i=1}^{n} x_i \sigma(b_i)\right|^2 \leqslant \left(\sum_{i=1}^{n} |x_i \sigma(b_i)|\right)^2 \leqslant \|x\|_2^2 \, \|\sigma(\mathcal{B})\|_2^2 .$$

Therefore for any $x \in K$ and any $\sigma \in \mathrm{Hom}(K, \mathbb{C})$, $\|x\|_2^2 \geqslant \frac{|\sigma(x)|^2}{\|\sigma(\mathcal{B})\|_2^2}$. We chose to define the function `NormEvaluation_heur` as the max of such quotients, as follows. For any $\sigma \in \mathrm{Hom}(K, \mathbb{C})$, let $B_{f,\sigma}$ be such that

$$\forall x \in Z_K(f), |\sigma(x)|^2 \leqslant B_\sigma. \tag{4.8}$$

Then we fix

$$\texttt{NormEvaluation\_heur}(x) = \max \left\{ \frac{|\sigma(x)|^2}{\|\sigma(\mathcal{B})\|_2^2} \mid \sigma \in \mathrm{Hom}(K, \mathbb{C}) \right\}. \tag{4.9}$$

Even if `NormEvaluation_heur` gives a value which is a lower bound on $\|x\|_2^2$ instead of an upper bound, it allows a first precision to be obtained. This evaluation is usually smaller than B_f and gives a good starting point, i.e. we do not need to increase much the precision to retrieve at least *some roots*.

Comparison with `nfroots` If we obtain a good evaluation for the needed precision, we can pinpoint it with our method. Additionally we will see it can be adapted to take advantage of the structure of extension L/K. However, the basis that we reduce is close to a knapsack matrix. These lattices are more difficult to reduce than most basis matrices of ideals reduced in `nfroots`. Moreover, Belabas mentions that over these lattices, one can use a pre-reduction which decreases considerably the running time of the subsequent LLL algorithm [7].

4.2.2 Relative method

Let us now describe a method to recover the roots of a polynomial in an extension of number fields L/K.

Decoding in subfield

First let us describe how one can reduce knowledge of embeddings related to the extension L/K to decodings in K. We will use the relative Minkowski embedding $\sigma_{L/K}$. It defines a K-linear embedding of $L \cong K^n$ into Ω^n where n is the degree of L/K. More precisely, let us fix $\mathcal{E} = (e_1, \dots, e_n)$ a K-basis of L and let $x = x_1 e_1 + \cdots + x_r e_n \in L$. We assume that the action of each $\sigma \in \mathrm{Hom}(L/K, \Omega)$ on $\mathcal{E}$ is known. Then $\sigma_{L/K}$ sends K^n into the cartesian product of the conjugates of L/K,

$$\begin{array}{rccl} \sigma_{L/K}: & K^n & \longrightarrow & \widetilde{L}^n \subset \Omega^n \\ & (x_i)_{i \in [\![1,n]\!]} & \longmapsto & \left(\sum_{i=1}^n x_i \sigma(e_i)\right)_{\sigma \in \mathrm{Hom}(L/K,\Omega)} \end{array}$$

However we will need to consider this embedding as an embedding into $\mathbb{C}^n$. In order to do so, one needs to specify an embedding of K into $\mathbb{C}$. Let us fix $\tau \in \mathrm{Hom}(K, \mathbb{C})$. Then for any $\sigma \in \mathrm{Hom}(L/K, \mathbb{C})$, its action on $x \in L$ can be seen as $\sum_{i=1}^n \tau(x_i)\sigma(e_i)$. This way, K is identified with a subfield of $\mathbb{C}$ and L is identified with $\tau(K)^n$. We can then express the action of $\sigma_{L/K}$ from $\tau(K)^n$ into $\mathbb{C}^n$. It is expressed as a matrix in $\mathrm{M}_n(\mathbb{C})$. Let us write $\Sigma_{L/K}$ this matrix. We have $(\Sigma_{L/K})_i = \sigma_{L/K}(e_i)$.

Thus we are able to do the following. Given knowledge of $\sigma_{L/K}(x)$ one can apply $\Sigma_{L/K}^{-1}$ to it and obtain $(\tau(x_i))_i$. Obviously, computations are done up to a given precision l. Thus knowing $\sigma_{L/K}(x)$ up to l, one can find the the approximations of $(\tau(x_i))_i$. Then one can retrieve each x_i by the decoding method explained previously, with `TestDecode`.

Thus we obtain Algorithm 24 which retrieves coefficients of x knowing its Minkowski embeddings.

An algorithm for polynomial roots

Now we can apply the previous strategy to compute polynomial roots by decoding in the subfield K. Again, we fix some objects. The extension of number fields L/K is given by a $\mathbb{Q}$-basis $\mathcal{B}$ and and K-basis of L. Then consider a polynomial $f(X) \in L[X]$ such that $Z_L(f) \subset \mathbb{Z}[\mathcal{E} \otimes \mathcal{B}]$. From what we described previously, in order to retrieve the coefficients of $x \in Z_L(f)$, one can compute $\sigma_{L/K}$ and use

Algorithm 24 `Mink2coeff`

```
Require: An extension $L/K$, given by $\mathcal{B}$ a basis of $K$ and $\mathcal{E}$ a $K$-basis of $L$, an
    integer $l$, the matrix $L_l$ from a reduced basis of $\mathcal{L}(\mathcal{B}, \tau, l)$, $M = \Sigma_{L/K}^{-1}$ up to a
    precision $l + s$, and $X = [\sigma_{L/K}(x)]_{l+s}$ for some $x \in L/K$
Ensure: A candidate $y = (y_1, \dots, y_N)$ for the vector of coefficients of $x$ expressed
    in $\mathcal{B} \otimes \mathcal{E}$
1: $Y \leftarrow XM$
2: $y = ()$
3: for $i = 1$ to $n$ do
4:     $y \leftarrow [y \mid \texttt{TestDecode}(L_l, Y_i)]$        ▷ Concatenation of row vectors
5: end for
6: return $y$
```

`Mink2coeff`. One can imagine the following main steps:

1. compute a precision l certifying the correctness of the computation;

2. compute L_l a LLL reduced basis of $\mathcal{L}(\mathcal{B}, \tau, l)$;

3. compute $M = \Sigma_{L/K}^{-1}$ up to precision $l + s$ for some s;

4. compute $Z = \prod_{\sigma \in \mathrm{Hom}(L/K,\mathbb{C})} Z_{\mathbb{C}}(f^\sigma)$ up to precision $l + s$;

5. For each $x \in Z$, use `Mink2coeff` to obtain a root candidate.

This leads to Algorithm 25 below.

Remark 25. The set Z is the cartesian product of the sets $Z(f^\sigma)$ with σ in $\mathrm{Hom}(L/K, \mathbb{C})$. Each set $Z(f^\sigma)$ has at most $d = \deg f$ elements. Therefore, Z is a set of at most d^n complex numbers. Moreover one cannot tell a priori if an element $z \in Z(f^\sigma)$ is of the form $[\sigma(x)]_{l+s}$ for some $x \in Z_L(f)$, or even if a vector $(z_1, \dots, z_n) \in Z$ corresponds to a root x of $f(X)$. Thus one has to call `Mink2coeff` d^n times in the worst case. Even if $f(X)$ splits in L, this leads to a search of d vectors in a set of size d^n.

A large enumeration cost The cost of Algorithm 25 compared with Algorithm 23 is as follows. The absolute method `PolynomialRoots` requires at most $d = \deg f(x)$ decodings, while `RelativePolynomialRoots_naive` requires enumerating through d^n vectors, corresponding to n decodings each. The mere enumeration of d^n elements shows that `RelativePolynomialRoots_naive` has an exponential cost when n increases. In addition, several operations on vectors and matrices are done for each of the d^n possibilities. Thus it is quickly impractical.

Algorithm 25 `RelativePolynomialRoots_naive`

```
Require: An extension $L/K$, given by $\mathcal{B}$ a basis of $K$ and $\mathcal{E}$ a $K$-basis of $L$, and
    $f(X) \in L[X]$ such that $Z_L(f) \in \mathbb{Z}[\mathcal{E} \otimes \mathcal{B}]$
Ensure: $Z_L(f)$
 1: $l \leftarrow$ PrecisionEvaluation$(f(X))$
 2: $L_l \leftarrow \mathrm{LLL}(B(\mathcal{B}, \tau, l))$
 3: $M \leftarrow \Sigma_{L/K}^{-1}$                              ▷ Up to precision $l + s$
 4: $Z \leftarrow \prod_{\sigma \in \mathrm{Hom}(L/K,\mathbb{C})}$ FloatPolynomialRoots$(f^\sigma, l + s)$
 5: $S \leftarrow \emptyset$
 6: for $z \in Z$ do
 7:     $y \leftarrow$ Mink2coeff$(z, M, L_l)$                  ▷ Compute the inverse and decode
 8:     if $f(y) = 0$ then
 9:         $S \leftarrow S \cup \{y\}$
10:     end if
11: end for
12: return $y$
```

Speed-up for small relative degrees For fixed small n, Algorithm 25 can considerably speed-up the computation of $Z_K(f)$. Indeed one has to remember that an important part of the computation time is dedicated to the reduction of the lattice used to decode, as is also the case for the algebraic method. Algorithm 23 requires the reduction of a lattice of rank $[L : \mathbb{Q}]$, while the rank of the lattice reduced in Algorithm 25 is $\frac{[L:\mathbb{Q}]}{n}$. In addition, the precision needed to certify the computation shown in Theorem 4.1 involves the dimension. Therefore, dividing the dimension allows us to do computations at a smaller precision, which also leads to smaller coefficients in the matrix B_l which is reduced.

Improving the naive algorithm

In order to improve the relative method described above, one has two main directions to follow. The first is to reduce the number of possibilities, i.e. eliminate as quickly as possible the branches in the enumeration tree. The second is to speed-up intermediate computations, in particular the ones related to testing whether a vector of possible conjugates is a solution or not.

A better search A simple observation can be made. Let us write $Z = \prod_\sigma Z_{\mathbb{C}}(f^\sigma)$ where σ ranges over $\mathrm{Hom}(L/K, \mathbb{C})$. Then one has

$$\forall x \in Z_K(f), \forall \sigma \in \mathrm{Hom}(L/K, \mathbb{C}), \exists! z \in Z_{\mathbb{C}}(f^\sigma) \mid z = [\sigma(x)]_l, \tag{4.10}$$

which implies that once we found a correct vector in Z we can remove from the search tree all the nodes where any of its coordinates appears.

Notation. We will denote by `UpdateTree` the procedure updating the search space as described.

Algorithm 26 `TestAndUpdate`

```
Require: An extension $L/K$, given by $\mathcal{B}$ a basis of $K$ and $\mathcal{E}$ a $K$-basis of $L$, and
    $f(X) \in L[X]$ such that $Z_L(f) \in \mathbb{Z}[\mathcal{E} \otimes \mathcal{B}]$, $x \in L$, $S$ a set, $Z$
Ensure: Tests if $x \in Z_K(f)$, updates $S$ and $Z$
1: $b \leftarrow f(x) = 0$
2: if $b$ then
3:     $S \leftarrow S \cup \{x\}$
4:     $Z \leftarrow$ UpdateTree$(Z, x)$                ▷ Update the search space
5: end if
6: return $b$, $S$, $Z$
```

Identify vectors more quickly The other way to accelerate the computation is to decide more quickly if a vector $z \in Z$ is equal to $[\sigma_{L/K}(x)]_l$ for some $x \in Z_K(f)$. The naive method requires applying `Mink2coeff` to z and check if it is a root of $f(X)$. As already mentioned, this requires multiplying z by the complex matrix $\Sigma_{L/K}^{-1}$, then applying `TestDecode` to n vectors. Even if $\Sigma_{L/K}^{-1}$ is precomputed and `TestDecode` is generally fast (as it solves BDD), improving these procedures or removing some intermediate parts to reach a conclusion can really improve the overall performance, as one needs to repeat these operations a large amount of times. Let us explain how it can be done.

If l is large enough, then a decoding of a true vector is a BDD. This means that the target vector is very close to a vector of $\mathcal{L}(\mathcal{B}, \sigma, l)$. However if z is not a solution vector, then it does not correspond to $\sigma_{L/K}(y)$ for any $y \in L$. Thus the vector $z\Sigma_{L/K}^{-1}$ should be far from the lattice, or at least far enough so its distance can be distinguished from a BDD situation. If so, the decoding should yield larger coefficients. The Euclidean norm of the candidate vector should then be larger than the one of solutions. Then recall that the decoding is done with respect to the basis of K. Therefore, one can decode only the first coordinate of $z\Sigma_{L/K}^{-1}$ for each z and compare the norms of the vectors obtained. The ones with smaller norms should be the first part of the solutions, while the others can be discarded.

First let us define one intermediate procedure, `Check1Coord`, which compares the norm of a coefficient `Mink2coeff`(z, M, L_l) previously computed with the new vector. It outputs a boolean indicating if the new vector of Z is a potential root candidate. It is described in Algorithm 27.

Remark 26. The constant C found in Algorithm 27 is the value used as the comparison quotient. Consider two norms n_1 and n_2. If n_2 is known to be the partial norm

Algorithm 27 `Check1Coord`

```
Require: The matrix $L_l$ from a reduced basis of $\mathcal{L}(\mathcal{B}, \tau, l)$, $M = \Sigma_{L/K}^{-1}$ up to a
    precision $l + s$, $b_t$ the boolean used to test, $n_t$ the norm used to test, $I$ the index
    used to test, and $z \in Z$ the vector which is checked
Ensure: A boolean indicating if $z$ should be fully decoded
 1: $y \leftarrow (zM)_I$                          ▷ Computes coordinate $I$ of $z\Sigma_{L/K}^{-1}$
 2: $x \leftarrow$ TestDecode$(L_l, x)$
 3: $n \leftarrow \|x\|_2^2$
 4: $q \leftarrow n/n_t$
 5: $b \leftarrow$ false
 6: if $b_t$ then                  ▷ Check if the vector used for comparison is a solution
 7:     if $q < C$ then $b \leftarrow$ true
 8:     end if
 9: else
10:     if $q < 1/C$ then $b \leftarrow$ true
11:     end if
12: end if
13: return $b$
```

of a solution, we will consider that n_1 is the potential norm of a root if $n_1 < Cn_2$. However if n_2 is not the partial norm of a solution, then we will consider that the vector linked to n_1 is worth considering if $Cn_1 < n_2$. We chose not to put C as an input for clarity of the exposé. In our implementation we chose $C = 10^r$ for $r \in \{2, 3\}$.

Another auxiliary function needed consists of initialising the variables used for `Check1Coord` (index, norm, and boolean) with the first vector computed. One can find it described in Algorithm 28. It also verifies if the first element decoded is a solution, updates the search space Z and the set of solutions found S.

Algorithm 28 `InitTest`

```
Require: An extension $L/K$, given by $\mathcal{B}$ a basis of $K$ and $\mathcal{E}$ a $K$-basis of $L$,
    $f(X) \in L[X]$ such that $Z_L(f) \subset \mathbb{Z}[\mathcal{E} \otimes \mathcal{B}]$, and $y \in L$
Ensure: $b$ a booelan indicating if $f(y) = 0$, $n$ the norm of one the coefficient in $K$
    of $y\Sigma_{L/K}^{-1}$, $I$ the index of said coefficient, $S$ and $Z$ the updated sets of solutions
    found and search space respectively
1: $b, S, Z \leftarrow$ TestAndUpdate$(y, \emptyset, Z)$
2: $n \leftarrow 0$, $I \leftarrow 0$
3: while $n = 0$ do
4:     $I \leftarrow I+1$
5:     $n \leftarrow \|y_I\|_2^2$
6: end while
7: return $b$, $n$, $I$, $S$, $Z$
```

Now we are all set to write Algorithm 29, which computes the roots of a polynomial using the heuristic method we described. We already mentioned that the set Z is ordered, which allows it to be searched through and updated more efficiently.

Algorithm 29 `RelativePolynomialRoots`

Require: An extension L/K, given by $\mathcal{B}$ a basis of K and $\mathcal{E}$ a K-basis of L, and $f(X) \in L[X]$ such that $Z_L(f) \in \mathbb{Z}[\mathcal{E} \otimes \mathcal{B}]$
Ensure: $Z_L(f)$

```
1: $l \leftarrow$ PrecisionEvaluation$(f(X))$
2: $L_l \leftarrow \mathrm{LLL}(B(\mathcal{B}, \tau, l))$
3: $M \leftarrow \Sigma_{L/K}^{-1}$                                  ▷ Up to precision $l + s$
4: $Z \leftarrow \prod_{\sigma \in \mathrm{Hom}(L/K,\mathbb{C})}$ FloatPolynomialRoots$(f^\sigma, l + s)$
5: $S \leftarrow \emptyset$
6: $y \leftarrow$ Mink2coeff$(Z_1, M, L_l)$
7: $b_t, n_t, I, S, Z \leftarrow$ InitTest$(y, f, Z)$
8: for $z \in Z$ do
9:     $b \leftarrow$ Check1Coord$(z, b_t, n_t, I, L_l, M)$
10:    if $b$ then
11:        $x \leftarrow$ Mink2coeff$(x, M, L_l)$
12:        $b, S, Z \leftarrow$ TestAndUpdate$(x, S, Z)$
13:    end if
14: end for
15: return $S$
```

Average cost of the search

Let us study the average cost of the search phase of Algorithm 25 and Algorithm 29. In particular we will determine the average number of decodings that will occur before finding all roots. If one denotes by $\sigma_1, \ldots, \sigma_n$ the elements of $\mathrm{Hom}(L/K, \mathbb{C})$, then the two algorithms can be described as a search without replacement in the set $Z = Z_1 \times \cdots \times Z_n$ where $Z_i = Z_{\sigma_i}$.

Remark 27. A cartesian product $S = S_1 \times \cdots \times S_n$ will be ordered using the lexicographic order. This means that for all $(x, y) \in S^2$, if $i(x, y) = \min\{i \in [\![1, n]\!] \mid x_i \neq y_i\}$, then we have $x < y \iff x_{i(x,y)} < y_{i(x,y)}$.

From now on, let us consider the sets Z_i as ordered sets of elements $z_{i,j}$ with $z_{i,j} < z_{i,j'} \iff j < j'$. Moreover we consider that we run through Z following the lexicographic order. Assume that the state of the computation is at the state with index $j = (j_1, \ldots, j_n) \in [\![1, \deg f(X)]\!]^n$, and that we found a root x. We can write $x = (z_{1,j_1}, \ldots, z_{n,j_n})$. Then the action of `UpdateTree` on Z is as follows. We mentioned that it removes z_{i,j_i} from Z for all $i \in [\![1, n]\!]$. This amounts to removing z_{i,j_i} from Z for all $i \in [\![2, n]\!]$ and updating the index by doing $j_1 \leftarrow j_1 + 1$ and for all $i > 1, j_i \leftarrow 1$.

Definition 4.4. Let N, M, m be integers satisfying $m \leqslant M \leqslant N$. A random variable X taking non-negative values follows a *negative hypergeometric distribution with parameters (N, M, m)* if it satisfies the following formula:

$$\mathbb{P}(X = k) = \frac{\binom{k+m-1}{k}\binom{N-m-k}{M-m}}{\binom{N}{m}}.$$

We will write $X \sim NHG(N, M, m)$.

The negative hypergeometric distribution describes exactly what we want to study. Indeed, it arises as follows. Consider a set of N elements, containing M "success elements" and $N - m$ "fail elements". Then if one draws uniformly in the set *without replacement* until m successes are found, then the number of failures drawn follows a negative hypergeometric situation.

Proposition 4.6 ([55]). *Let N, M, m be integers, and X be a random variable such that $X \sim NHG(N, M, m)$. Then one has:*

$$\mathbb{E}[X] = m\frac{N-M}{M+1} \quad \text{and} \quad \mathrm{Var}[X] = m\frac{(N-M)(N+1)}{(M+1)(M+2)}\left(1 - \frac{m}{M+1}\right).$$

We are essentially interested in the average cost, so we focused on the expectation.

Proposition 4.7. *Let L/K be an extension of number fields such that $[L : K] = n$. Consider $f(X) \in L[X]$ such that $|Z_K(f)| = s$ and write $d = \deg f(X)$. Then the average number of "failed" decodings done in Algorithm 25 before finding one root is $\frac{d^n+1}{s+1}$. The average number of "failed" decodings before finding all the roots is $s\frac{d^n+1}{s+1}$.*

Proof. Let X_1 be the random variable representing the number of failures before finding the first root, and X_s be the random variable of the number of failures before finding all the roots. Clearly one has $X_1 \sim NGH(d^n, s, 1)$ and $X_s \sim NGH(d^n, s, 1)$, and can apply directly the formula of the expectation from Proposition 4.6 to find $\mathbb{E}[X_1]$ and $E[X_s]$. □

The study is slightly more complex for the search done in Algorithm 29. One can remark that the number of elements removed from the search space Z by `UpdateTree` depends on the index of state. We will therefore consider the random variables corresponding to the number of failures between two found solutions.

Notation. Let L/K be an extension of number fields such that $[L : K] = n$. Consider $f(X) \in L[X]$ such that $|Z_K(f)| = s$. We will denote by $x^{(1)} < \cdots < x^{(s)}$ the elements of $Z_K(f)$ ordered in Z. For each $k \in [\![1, s]\!]$ we will consider several random variables.

1. $X^{(k)}$ is the random variable corresponding to the number of failures between the $(k-1)$-th solution and the k-th solution.

2. $X_1^{(k)}$ is the random variable corresponding to the number of failures in the first coordinate between the $(k-1)$-th solution and the k-th solution.

3. $X_2^{(k)}$ is the random variable corresponding to the number of failures occurred between the $(k-1)$-th solution and the k-th solution such that $j_1 = x_1^{(k)}$.

4. $Y^{(k)}$ is the random variable corresponding to the number of failures which occured before the k-th solution is found.

Lemma 4.3. *Let L/K be an extension of number fields such that $[L : K] = n$. Consider $f(X) \in L[X]$ such that $|Z_K(f)| = s$. Then one has the following:*

$$\forall k \in [\![1, s]\!], Y^{(k)} = \sum_{j=1}^{k} \left(X_1^{(j)}(d-j+1)^{n-1} + X_2^{(j)} \right).$$

Proof. Let us fix $k \in [\![1, s]\!]$. Clearly one has $Y^{(k)} = \sum_{j=1}^{(k)} X^{(j)}$. Now let us denote by C_k the integer $|Z_2 \times \cdots \times Z_n|$ after the $(k-1)$-th root and before the k-th root are found. Recall that after each new solution is found, `UpdateTree` removes one element of each $Z_i, i > 1$. Therefore, one obtains $C_k = (d-k+1)^{n-1}$. Because the search is done following the lexicographic order, it is easy to see that for each fixed $j_1 \in [\![x_1^{(k-1)} + 1, x_1^{(k)}]\!]$ there are two possibilities. If $j_1 < x_1^{(k)}$ then the search will run through all $\{z_{1,j_1}\} \times Z_2 \times \cdots \times Z_n$, which contains no solution. This leads to C_k failures. If $j_1 = x_1^{(k)}$ then the search will run through $\{z_{1,j_1}\} \times Z_2 \times \cdots \times Z_n$ until finding the solution. It amounts to $X_2^{(k)}$ failures. Finally the number of j_1 that are passed such that $j_1 < x_1^{(k)}$ is $X_1^{(k)}$. □

Proposition 4.8. *Let L/K be an extension of number fields such that $[L : K] = n$. Consider $f(X) \in L[X]$ such that $|Z_K(f)| = s$. Write $d = \deg f(X)$. Then, for $k \in [\![1, s]\!]$, the average number of "failed" decodings done in Algorithm 29 before finding k roots is*

$$\frac{2d-s+1}{2(s+1)} \sum_{j=0}^{k-1} (d-j)^{n-1} - \frac{k}{2}. \tag{4.11}$$

Proof. Using Lemma 4.3 and by linearity of the expectation, one has:

$$\forall k \in [\![1, s]\!], \mathbb{E}[Y^{(k)}] = \sum_{j=1}^{k} \left(\mathbb{E}[X_1^{(j)}](d-j+1)^{n-1} + \mathbb{E}[X_2^{(j)}] \right).$$

Let fix $j \in [\![1, k]\!]$. Recall that $X_2^{(j)}$ is the number of failures found during the search through the set $\{x_1^{(j)}\} \times Z_2 \times \cdots \times Z_n$. We know there is exactly one solution in this set. Therefore we have $X_2^{(j)} \sim NHG((d-j+1)^{n-1}, 1, 1)$ and

$$\mathbb{E}[X_2^{(j)}] = \frac{(d-j+1)^{n-1}-1}{2}.$$

Now let us determine $X_1^{(j)}$. It is the number of wrong first coordinates visited until finding $x_1^{(j)}$, and after finding $x_1^{(j-1)}$. The search is done over the set Z_1 minus the elements visited before $x_1^{(j-1)}$ included. It is a set with cardinal number

$$|Z_1| - \sum_{i=1}^{(j-1)} X_1^{(i)} - (j-1) = d-j+1-\sum_{i=1}^{(j-1)} X_1^{(i)}$$

which contains $s-(j-1)$ success elements. Therefore we have $\mathbb{E}[X_1^{(j)} \mid \sum_{i=1}^{(j-1)} X_1^{(i)} = a] \sim NHG(d-j+1-a, s-(j-1), 1)$, for all possible a. Using the law of total expectation $\mathbb{E}[X] = \mathbb{E}[\mathbb{E}[X|Y]]$ between two variables X and Y, it is easy to see that we have

$$\mathbb{E}[X_1^{(j)}] = \frac{(d-j+1) - \sum_{i=1}^{(j-1)} \mathbb{E}[X_1^{(i)}] - (s-j+1)}{s-j+2} = \frac{(d-s) - \sum_{i=1}^{(j-1)} \mathbb{E}[X_1^{(i)}]}{s-j+2}.$$

It is possible to use this recurrence relation to obtain an expression of $\mathbb{E}[X_1^{(j)}]$ in closed-form. As a matter of fact, we will prove that $\mathbb{E}[X_1^{(j+1)}] = \mathbb{E}[X_1^{(j)}]$. Indeed one has

$$\mathbb{E}[X_1^{(j+1)}] = \frac{(d-s) - \sum_{i=1}^{(j)} \mathbb{E}[X_1^{(i)}]}{s-j+1} = \frac{d-s-\mathbb{E}[X_1^{(i)}] - \sum_{i=1}^{(j-1)} \mathbb{E}[X_1^{(i)}]}{s-j+1},$$

and since

$$(s-j+2)\mathbb{E}[X_1^{(j)}] = (d-s) - \sum_{i=1}^{(j-1)} \mathbb{E}[X_1^{(i)}]$$

we obtain

$$\mathbb{E}[X_1^{(j+1)}] = \frac{d-s-\mathbb{E}[X_1^{(i)}] + (s-j+2)\mathbb{E}[X_1^{(j)}] - (d-s)}{s-j+1} = \mathbb{E}[X_1^{(j)}].$$

Then remark that $X_1^{(1)} \sim NHG(d, s, 1)$, which gives the desired result. □

Remark 28. One can see that the gain of `RelativePolynomialRoots` comparatively to `RelativePolynomialRoots_naive` increases with s. Indeed if $f(X)$ has only one root, then the expected number of decodings is the same for both methods. However, if $s = \deg f(X)$ then the expected number of decodings with the naive search is $d\frac{d(d^{n-1}-1)}{d+1}$, whereas it is $\sum_{i=0}^{d-1} \frac{(d-i)^{n-1}-1}{2}$ when using `UpdateTree`.

4.2.3 Experimental results

We will now compare the practical performances of the methods described previously with the generic algebraic method implemented in PARI/GP [76], which is the function `nfroots`.

First we will consider two versions of Algorithm 23. First is the certified version, for which we will keep writing `PolynomialRoots`. Then we will denote by `PolynomialRoots_heur` the version where `NormEvaluation_heur` is used, as well as the formulae

$$[K:\mathbb{Q}]\frac{\ln \|x\|_2^2}{2\ln(2)} + 2[K:\mathbb{Q}] \tag{4.12}$$

instead of Equation (4.6).

Then we will compare `PolynomialRoots_heur` with `nfroots` in situations where we assume that some extra information is known about the polynomial $f(X)$, or where the goal is slightly different from retrieving the full set $Z_K(f)$.

Finally, we will do the comparison with `RelativePolynomialRoots` over relative extensions L/K.

Recall that we assume that $Z_K(f) \subset \mathbb{Z}[\mathcal{B}]$, with $\mathcal{B} = (b_1, \dots, b_n)$ being some $\mathbb{Q}$-basis of K. Given $x \in K$ we will keep denoting by $x_1, \dots, x_n$ its coefficients relative to $\mathcal{B}$. In all that follows, we chose $\mathcal{B}$ to be $\mathbb{Z}[\theta]$ with $\theta = X \bmod P(X)$, where $P(X)$ is a fixed polynomial defining the field K.

Solving a polynomial equation

We are interested here in the generic problem of finding all elements of $Z_K(f)$. We wish to study the practicability of `PolynomialRoots` and `PolynomialRoots_heur` in terms of time efficiency and goal achievability, i.e. if $Z_K(f)$ can be fully recovered. We will study the impact of the different parameters of the problems which are the number field K, the shape of the polynomial $f(X)$, and the size of the roots $\log_2 \|x\|_2$. Each time we focus on a parameter, we fix the others and observe the data given by experiments. We also differentiated between number fields K such that $r_1 > 0$ or such that $r_1 = 0$, since the matrices used to decode do not have the same sizes in these two cases.

Impact of K Let us explore how the choice of a number field K and the way it is represented can impact the different methods. Classically, K is given by $P(X)$ an irreducible polynomial of $\mathbb{Q}[X]$. We decided to study the impact of two parameters linked to $P(X)$, namely $\deg P(X)$ and the size of its coefficients. Therefore we fixed the parameters of the problem linked to $f(X) \in K[X]$. More precisely we considered $f(X)$ such that:

- $\deg f(X) = 50$;
- $f(X)$ splits in K, i.e. $|Z_K(f)| = \deg f(X)$;
- $\forall x \in Z_K(f), \forall i \in [\![1, n]\!], |x_i| \leqslant 2^{10}$.

Then we considered $P(X) = p_0 + p_1X + \cdots + p_{n-1}X^{n-1} + X^n \in \mathbb{Z}[X]$ for increasing degrees n, and several coefficient sizes $s(P)$. More precisely for sizes $s(P) \in \{1, 5, 10\}$, we picked polynomials $P(X)$ such that $\forall i \in [\![0, n-1]\!], p_i \in [\![-2^{s(P)}, 2^{s(P)}]\!]$, and this for n increasing. The data obtained are shown in Figures 4.2 and 4.3.

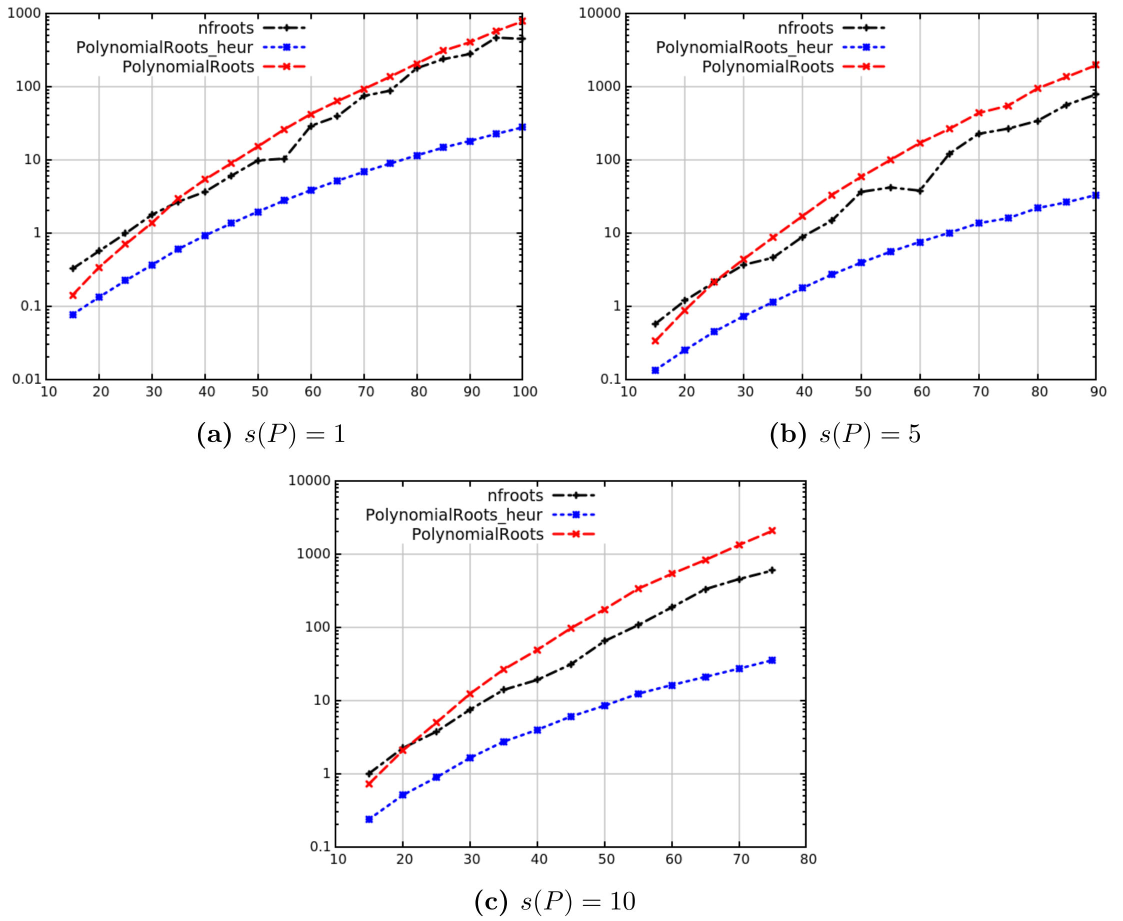
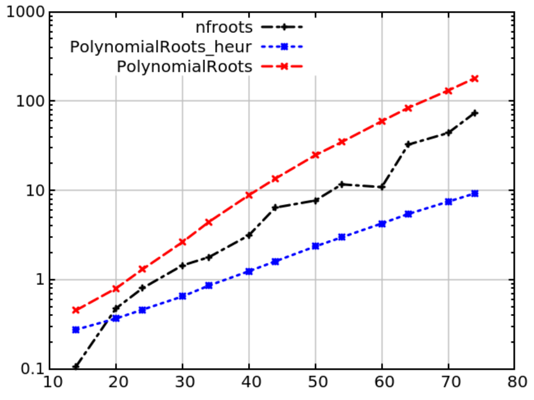
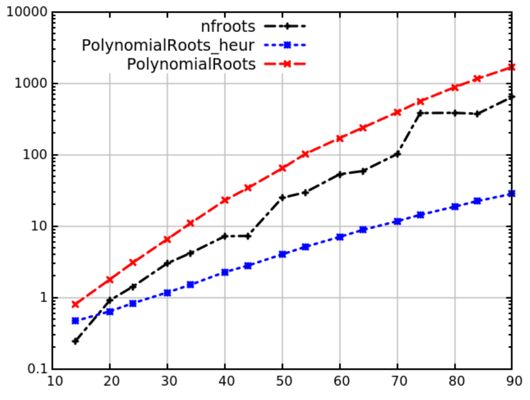


Figure 4.2: Average timings (s) of `nfroots`, `PolynomialRoots` and `PolynomialRoots_heur` plotted against $\deg P(X)$ for randomly generated $P(X)$ such that $r_1 > 0$, with $s(P) \in \{1, 5, 10\}$

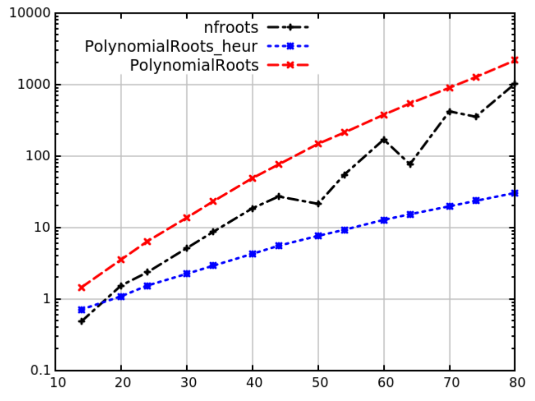
One can remark that the time efficiency of all three methods are widely influenced by the dimension $[K : \mathbb{Q}]$. This is less visible for `PolynomialRoots_heur` but it is still the case. It is easily explained by the fact that all three methods require the



(a) $s(P) = 1$



(b) $s(P) = 5$



(c) $s(P) = 10$

Figure 4.3: Average timings (s) of `nfroots`, `PolynomialRoots` and `PolynomialRoots_heur` plotted against $\deg P(X)$ for randomly generated $P(X)$ such that $r_1 = 0$, with $s(P) \in \{1, 5, 10\}$

computation of a LLL-reduced basis of a lattice with rank equal to $[K : \mathbb{Q}]$. Moreover the volume of said lattice depends also on the dimension.

The parameter $s(P)$, i.e. the coefficient size of the defining polynomial of K, also influences the performances of all three algorithms. Again our heuristic method seems to be less impacted by this parameter.

Finally, we also checked the number of roots properly retrieved by our methods. With the certified one, `PolynomialRoots`, all roots were found, but we can see from Figures 4.2 and 4.3 that it is way less efficient than `nfroots`, at least for the fixed shape of $f(X)$. Our method `PolynomialRoots_heur` – using heuristic norm evaluation and formula to compute the precision – is way more efficient than the certified version, even competing ***on average*** with `nfroots` in a number of cases. This is particularly the case when using a real embedding. However `PolynomialRoots_heur` did not retrieve all the roots. This phenomenon occurs only when $r_1 > 0$. One can find in Figure 4.4 the ratio of polynomials $f(X)$ for which all roots were retrieved.

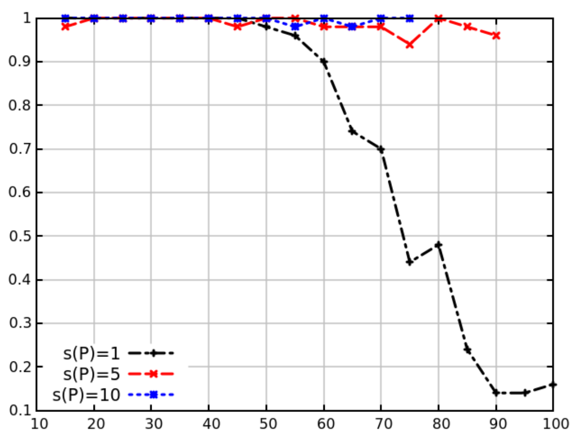


Figure 4.4: Ratios of roots retrieved by `PolynomialRoots_heur` over fields such that $r_1 = 0$ plotted against $[K : \mathbb{Q}]$, with $s(P) \in \{1, 5, 10\}$

The fact that all roots were retrieved when $r_1 = 0$ seems to indicate that the heuristic norm evaluation `NormEvaluation_heur` is good, and that the difference comes from the lattice used to decode. From the data gathered in Figure 4.4, one can see that the ratio of polynomial equations completely solved is dropping when the dimension is increasing. This indicates that the formula expressing the precision needed in function of the norm of the roots needs to be slightly bigger, and should be taking the degree $[K : \mathbb{Q}]$ more into account. For example one could consider

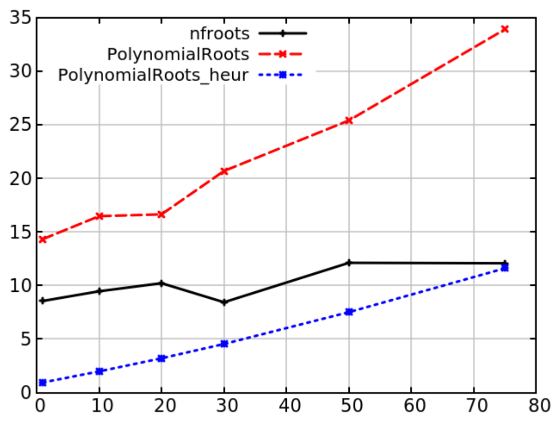
$$[K : \mathbb{Q}]\frac{\ln \|x\|_2^2}{2\ln 2} + \lceil \ln[K : \mathbb{Q}]\rceil \cdot [K : \mathbb{Q}]. \tag{4.13}$$

It would still represent a gain of $[K : \mathbb{Q}]^2$ compared to the certified precision given by Theorem 4.1. This indicates that for the fixed shape of $f(X)$ chosen, `PolynomialRoots_heur` is efficient, with respect to both time and probability of success.

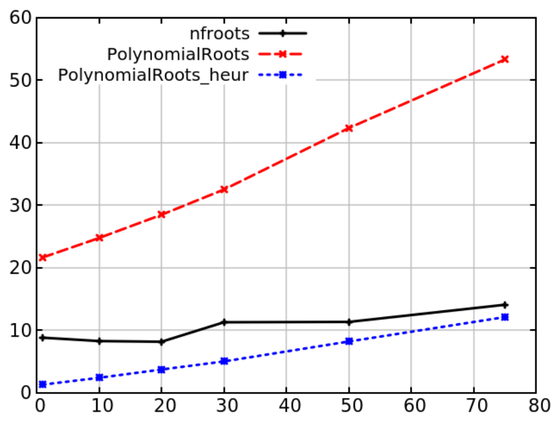
Size of roots We will now study how the size of the elements of $Z_K(f)$ impacts the performance of the different functions. To this end we fixed the parameters of the problem linked to $P(X) \in Z[X]$ defining the number field K. More precisely we considered $P(X)$ and $f(X)$ such that:

- $\deg f(X) = 50$;
- $f(X)$ splits in K, i.e. $|Z_K(f)| = \deg f(X)$;
- $s(P) = 1$;
- $\deg P(X) = 50$.

We did experiments for increasing size of roots. Let us denote by s_Z this size, i.e. $\forall x \in Z_K(f), \log_2|x_i| \in [\![-2^{s_Z}, 2^{s_Z}]\!]$. The results can be found in Figure 4.5.



(a) $[K : \mathbb{Q}] = 20$



(b) $[K : \mathbb{Q}] = 50$

Figure 4.5: Average timings (s) of `nfroots`, `PolynomialRoots` and `PolynomialRoots_heur` plotted against s_Z for randomly generated $P(X)$ such that $[K : \mathbb{Q}]$, with $r_1 > 0$ and $r_1 = 0$

One can verify that the cost of all methods looks linear in s_Z. It seems however that the slope for `nfroots` is smaller than the ones of `PolynomialRoots` and `PolynomialRoots_heur`. Again, `PolynomialRoots_heur` is competitive with `nfroots`, especially that almost all roots were retrieved. The only cases where it is not true are still when $r_1 > 0$. The ratios of roots retrieved in this case are plotted in Figure 4.6.

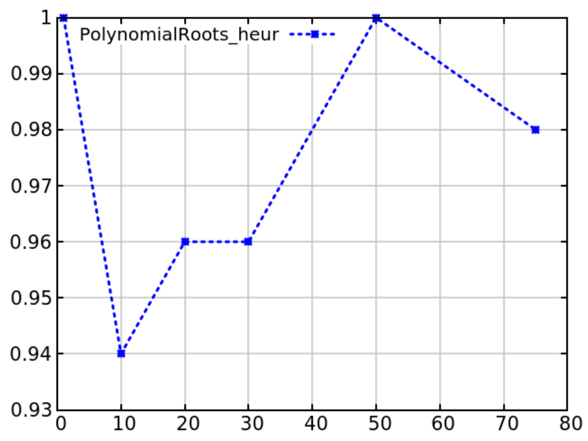


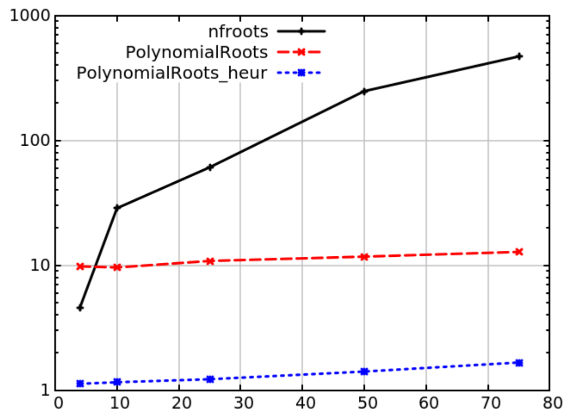
Figure 4.6: Ratios of equation for which all roots were retrieved by `PolynomialRoots_heur` over fields such that $r_1 = 0$ plotted against s_z

The ratio is always very high and does not seem to be influenced by the size of the roots as much as it is by the dimension of the field.

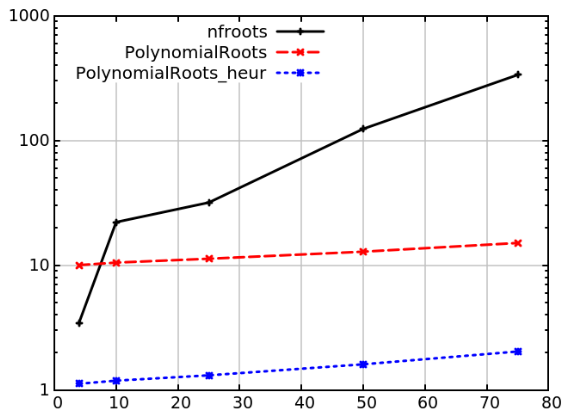
Shape of $f(X)$ Finally let us look into how the choice of $f(X)$ impacts the performance of the functions. Again we fixed the parameters linked to $P(X) \in \mathbb{Z}[X]$ defining the number field. Moreover we fixed $s_Z = 10$. Then we considered $f(X)$ with different splitting situations. More precisely, $f(X)$ and $P(X)$ are such that:

- $s(P) = 1$;
- $\deg P(X) = 51$;
- $f(X) = g(X)h(X)$ with $\deg g(X) = |Z_K(f)|$, $Z_K(g) = Z_K(f)$, $Z_K(h) = \emptyset$ and $q_f = \frac{\deg g(X)}{\deg f(X)} \in \{0.25, 0.5\}$;
- $s_Z = 10$;
- the coefficients of the polynomial $h(X)$ are drawn uniformly in $[\![-2^{10} \deg h, 2^{10} \deg h]\!]$

We did experiments for increasing degrees of $f(X)$. The results can be found in Figure 4.7

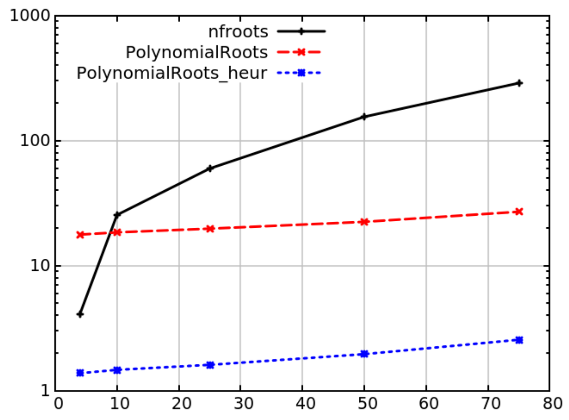


(a) $q_f = 25\%$

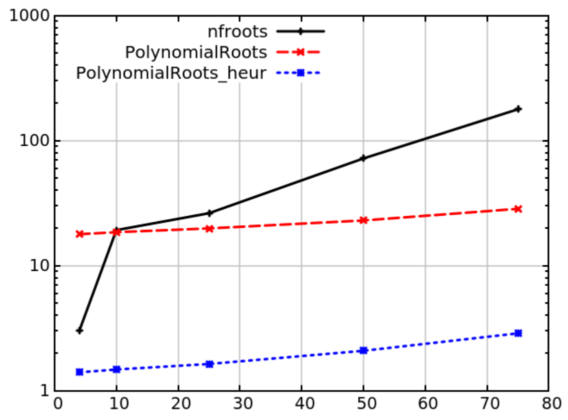


(b) $q_f = 50\%$

Figure 4.7: Average timings (s) of `nfroots`, `PolynomialRoots` and `PolynomialRoots_heur` plotted against $\deg f(X)$ for randomly generated $P(X)$ such that $r_1 > 0$, with $q_f \in \{25\%, 50\%\}$



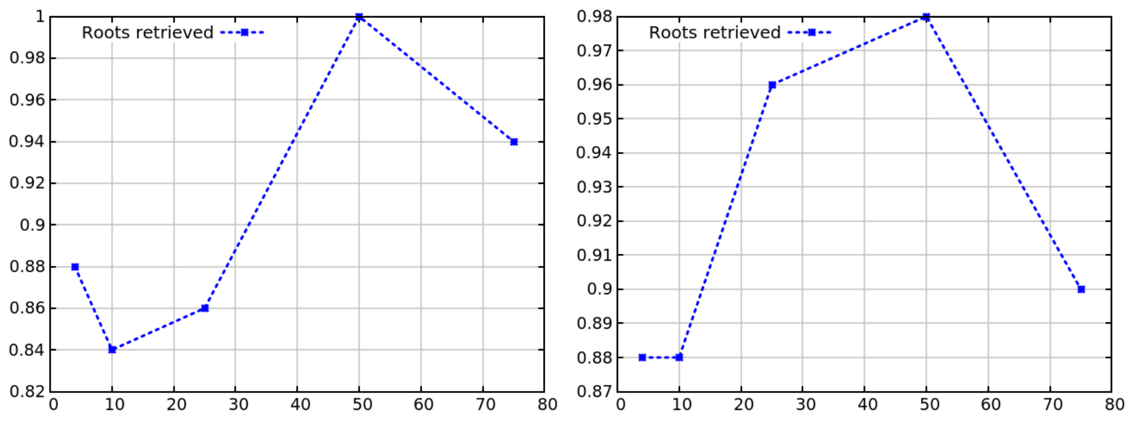
(a) $q_f = 25\%$



(b) $q_f = 50\%$

Figure 4.8: Average timings (s) of `nfroots`, `PolynomialRoots` and `PolynomialRoots_heur` plotted against $\deg f(X)$ for randomly generated $P(X)$ such that $r_1 > 0$, with $q_f \in \{25\%, 50\%\}$

We can remark that for the fixed types of number fields K and polynomial $f(X)$, our method is way more efficient than `nfroots`. This is a clear difference from the situations studied previously where $f(X)$ splitted completely, i.e. $f(X) = g(X)$. Again `PolynomialRoots_heur` does not allow the retrieval of all the roots when $r_1 > 0$, and only on this case. The ratios of polynomial $f(X)$ for which all the roots were recovered can be found in Figure 4.9. This time again, one can see that the ratios are high and do not seem to be correlated to depend on the degree of $f(X)$.



(a) $q_f = 25\%$ **(b)** $q_f = 50\%$

Figure 4.9: Ratios of roots retrieved with `PolynomialRoots_heur` plotted against $\deg f(X)$ for randomly generated $P(X)$ such that $r_1 > 0$, with $q_f \in \{25\%, 50\%\}$

Remarks From the different situations explored and the data gathered, we can conclude that the certified version of our method `PolynomialRoots` is in general less efficient than the algebraic method implemented in PARI/GP `nfroots`. However, it seems to behave better when $f(X)$ does not split completely in K. Then, the version of our method using a heuristic evaluation of the precision needed is more efficient, even competing with `nfroots` in some situations. In most cases it retrives all the roots, but fails in some circumstances. It is worth noticing that it is always when using a real embedding.

Relative extensions

Let us now consider relative extensions L/K, and study the efficiency of Algorithm 29. First we will compare the impact of our heuristic strategy. Then we will compare this method with Algorithm 23, and consider the impact of certain parameters on these algorithms.

Let us fix the notations. We will consider L/K together with $P_K(X) \in \mathbb{Q}[X]$ and $P_L(X) \in K[X]$ such that $K \cong \frac{\mathbb{Q}[X]}{(P_K(X))}$ and $L \cong \frac{K[X]}{(P_L(X))}$.

Checking one coordinate We considered extensions L/K such that $[K:\mathbb{Q}] = 30$ and $[L:K] = 3$ with:

- $s(P_K), s(P_L) \leqslant 1$;
- $\deg f(X) = 50$;
- $f(X)$ splits in K, i.e. $|Z_K(f)| = \deg f(X)$;
- $\forall x \in Z_K(f), \|x\|_\infty \leqslant 2^{10}$.

The timings obtained can be found in Figure 4.10.

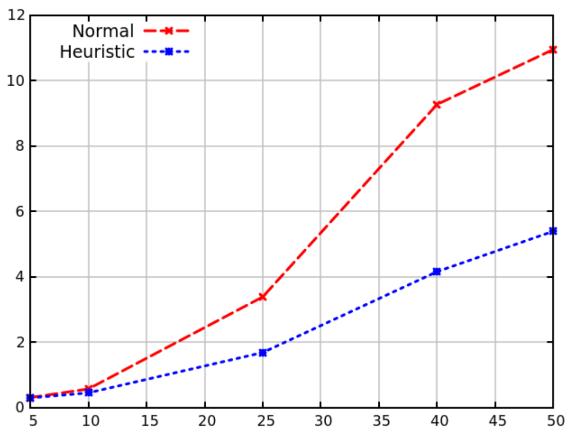


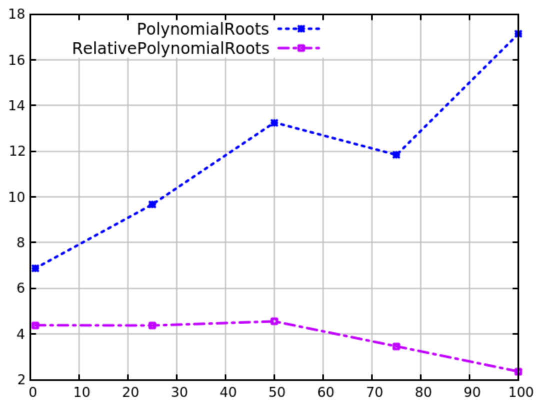
Figure 4.10: Average timings (s) of normal and heuristic versions of `RelativePolynomialRoots` plotted against $\deg f(X)$ for randomly generated $P_K(X), P_L(X)$

One can remark that using the heuristic strategy leads to computations between two times and three times faster. Moreover we can see that the timings are competitive with the ones presented in Figures 4.2 and 4.3 for example.

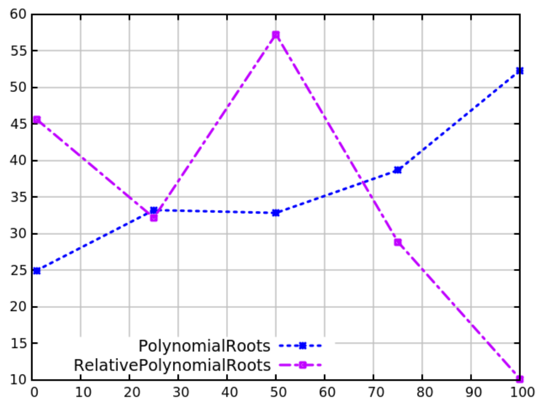
Number of roots We now study the influence of the number of roots. Here we will consider only the heuristic version of `RelativePolynomialRoots` (precision and search). We compare its performances with `PolynomialRoots_heur`. We forget about `nfroots` for now because we know from previous tests – Figures 4.7 and 4.8 – that it looses a lot of efficiency when the equation $f(X)$ is not completely split in the field.

We considered extensions L/K such that $[K:\mathbb{Q}] = 30$ and $[L:K] \in \{2,3\}$ with:

- $\|P_K\|_\infty, \|P_L\|_\infty \leqslant 2^1$;
- $\deg f(X) = 50$;
- $\forall x \in Z_K(f), \|x\|_\infty \leqslant 2^{10}$.



(a) $[L:K]=2$



(b) $[L:K]=3$

Figure 4.11: Average timings (s) of `PolynomialRoots` and `RelativePolynomialRoots` plotted against q_f for randomly generated $P_K(X), P_L(X)$ and $f(X)$, such that $[K:Q]=30$ and $\deg f(X)=50$.

The influence of the number of roots can be seen in Figure 4.11. The timings of the method `PolynomialRoots` increase with the number of roots, while the ones for the relative algorithm `RelativePolynomialRoots` globally decrease.

Kummer extensions

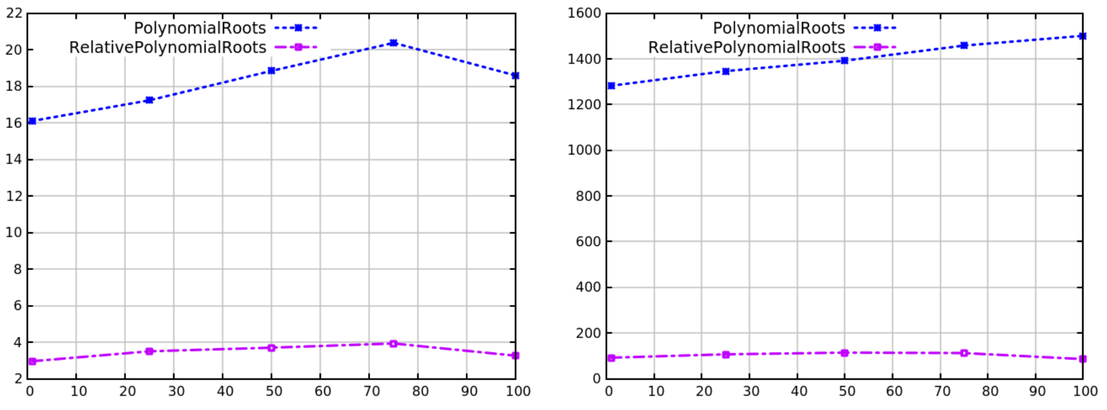
We will concentrate here on real Kummer fields of the form $L=\mathbb{Q}(\sqrt[p]{m_1},\ldots,\sqrt[p]{m_r})$, where $p>2$ is a prime integer and $(m_1,\ldots,m_r)\in\mathbb{Q}^r$ such that $[K:\mathbb{Q}]=p^r$. Fix $K=\mathbb{Q}(\sqrt[p]{m_1},\ldots,\sqrt[p]{m_{r-1}})$. Then L/K is an extension over which one can take full advantage of `RelativePolynomialRoots`. Indeed K can be embedded in $\mathbb{R}$, so that $\mathrm{Hom}(L/K,\mathbb{C})$ has one real embedding and $p-1$ complex ones. We will compare `PolynomialRoots`, `RelativePolynomialRoots` and `nfroots` of PARI/GP [76].

The first situation that we will explore will be the same as before: the number of roots. It will show how computations can be accelerated in good situations. Then we will study the special case where $f(X)$ has degree p. It is the direct generalisation of the case where $f(X)=X^p-\alpha^p$ with $\alpha\in L$, which is the type of equation that are to be solved in several tasks involving units or class group [40]. In particular, this task arose in several works these past few years [6, 64, 17] that we generalise in Chapter 5.

Remark 29 (`nfroots`)**.** Real Kummer extensions are all "bad" fields for the p-adic method. Indeed, it is mentioned by Belabas [7] that ideal lattices are usually easy to reduce – especially when one can use a pre-reduction algorithm – which occurs when the inertial degree of said ideal is large. Over real Kummer fields of the form $\mathbb{Q}(\sqrt[p]{m_1},\ldots,\sqrt[p]{m_r})$, the inertial degree cannot be larger than the exponent p.

Number of roots We keep the parameters described previously. The only modification is the shape of the extensions L/K which are Kummer fields, with exponent $p \in \{3, 5\}$.

One can find in Figure 4.12 the data collected for multicubic fields, i.e. Kummer extension with exponent 3. We considered the cases where $[K : \mathbb{Q}] \in \{27, 81\}$ and $[L : K] = 3$. Moreover each of the elements of the sequences $(m_i)_i$ defining the fields are prime integers smaller than 50.



(a) $[K : \mathbb{Q}] = 27$ **(b)** $[K : \mathbb{Q}] = 81$

Figure 4.12: Average timings (s) of `PolynomialRoots` and `RelativePolynomialRoots` plotted against q_f for randomly generated multicubic fields L/K, such that $[K : \mathbb{Q}] \in \{27, 81\}$.

We can see from Figure 4.12 that the relative method `RelativePolynomialRoots` is more efficient than `PolynomialRoots`, by a factor 5 when $[K : \mathbb{Q}] = 27$ and between 12 and 15 when $[K : \mathbb{Q}] = 81$. This illustrates the fact that with parameters similar otherwise, the advantage of reducing a basis lattice $\mathcal{L}(\mathcal{B}, \sigma, l)$ in a subfield increases with the degree of said subfield (and consequently, of the extension).

One can find in Figure 4.13 the data collected for Kummer extensions with exponent 5. We considered extensions such that $[K : \mathbb{Q}] = 25$ and $[L : K] = 5$. Again, all elements m_i defining the Kummer fields considered are prime integers smaller than 50.

We can see that `RelativePolynomialRoots` is between 1.5 and 3 times faster than `PolynomialRoots`. Moreover, we remark that when there are few roots, times for `RelativePolynomialRoots` are similar to the cases of multicubic fields with $[L : \mathbb{Q}] = 243$. This is the result of the search space being larger when $p = 5$. However one can see that the timings obtained start to drop drastically starting from $q_f = 1/2$. This drop, which is predicted by Proposition 4.8, could not be

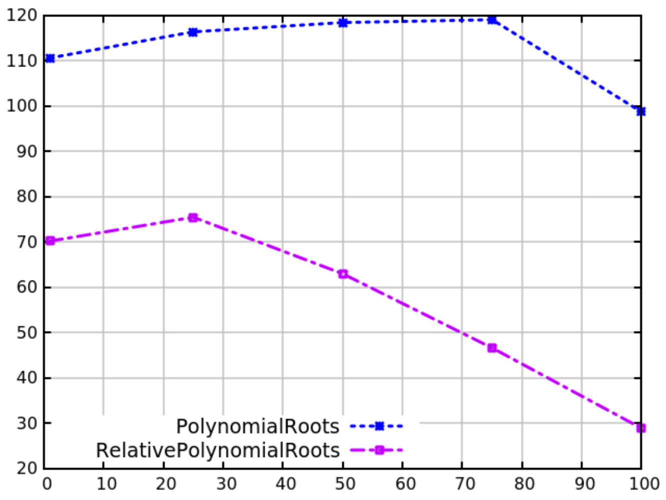


Figure 4.13: Average timings (s) of `PolynomialRoots` and `RelativePolynomialRoots` plotted against q_f for randomly generated Kummer fields L/K with exponent 5, such that $[K : \mathbb{Q}] = 25$.

observed as clearly from the data gathered previously. This is again because of the size of the search space.

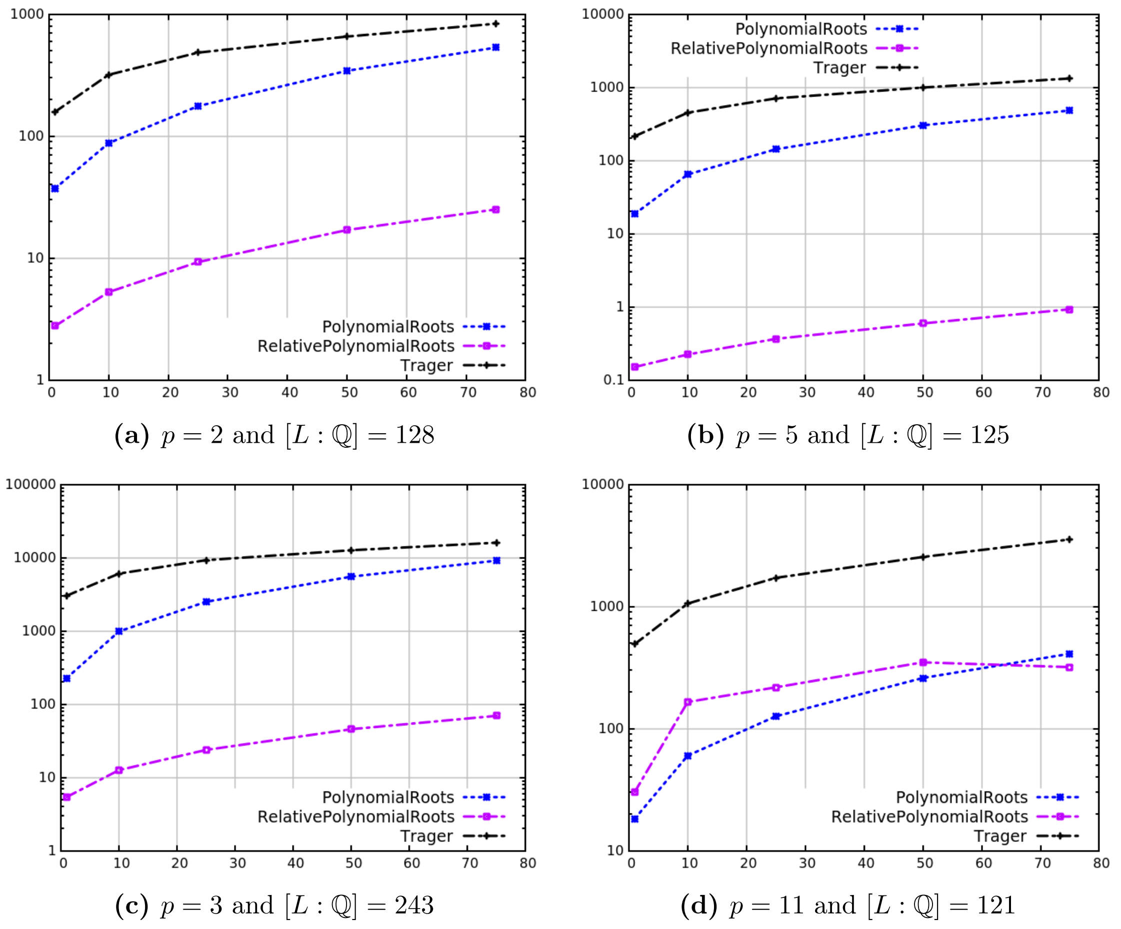
Small degree equations As mentioned we consider here polynomials $f(X)$ with small degrees, namely $\deg f(X) = p$ over real Kummer fields $\mathbb{Q}(\sqrt[p]{m_1}, \ldots, \sqrt[p]{m_r})$ of exponent p. This situation generalises the equations we need to solve to compute unit groups of such fields, and solve the PIP over them. For such degrees, the search space of Algorithm 29 is small enough to take full advantage of this method. Indeed, since the complex embeddings in $\mathrm{Hom}(L/K, \mathbb{C})$ are all conjugates (except one real embedding), the cardinality of the search space is $p^{\frac{p+1}{2}}$.

We considered Kummer fields of exponent p in $\{2, 3, 5, 11\}$. For this study, we compared our functions `PolynomialsRoots` and `RelativePolynomialRoots` as before, and we add `nfroots` as well.

Remark 30 (`nfroots`)**.** In this configuration, `nfroots` does not follow the method described before. Indeed, when $3 \deg f(X) < [L : \mathbb{Q}]$, the implementation of PARI/GP uses Trager's method [99, 7] for factorising polynomials. We will therefore refer to it as `Trager`, to differentiate it from the p-adic method described at the beginning of the section.

Here are how the different objects are drawn.

- Each m_i is a prime number smaller than 30.
- $\deg f(X) = p$ and $|Z_K(f)| = 1$.



(a) $p = 2$ and $[L : \mathbb{Q}] = 128$ (b) $p = 5$ and $[L : \mathbb{Q}] = 125$

(c) $p = 3$ and $[L : \mathbb{Q}] = 243$ (d) $p = 11$ and $[L : \mathbb{Q}] = 121$

Figure 4.14: Average timings (s) of `PolynomialRoots` and `RelativePolynomialRoots` and `Trager` plotted against s_Z for randomly generated Kummer fields L/K with exponent $p \in \{2, 3, 5, 11\}$ and $\deg f(X) = p$.

The data gathered can be found in Figure 4.14.

In most cases our relative method is way more efficient than the two others. It can go up to 500 times faster for $p = 5$, and up to 100 times faster for $p = 2$, 3. The method `Trager` implemented in PARI/GP is always worse than both our algorithms.

If one compares the data gathered in Figures 4.14a and 4.14b for $p = 2$ and $p = 5$ respectively – for which the degrees $[L : \mathbb{Q}]$ and the size of the search space are similar – one can see that the timings for `RelativePolynomialRoots` are around 10 times lower for $p = 5$. This is due to the fact that the dimension of the subfield K over which decodings are made is smaller in this case. These observations are confirmed with the timings gathered in Figure 4.14c.

Finally, one can remark in Figure 4.14d that the size of the search space is important, as `RelativePolynomialRoots` is less efficient than `PolynomialRoots` for $p = 11$. However in this case as for the others, it seems that the size of the roots is less of a problem for the relative method than for `PolynomialRoots`.

4.2.4 Conclusion

Algorithm 23 is a heuristic polynomial algorithm to retrieve the roots of a polynomial in a number field. Its certified version is slower than the classical algebraic method implemented in PARI/GP, called `nfroots`. However we saw that simple heuristic observations regarding the evaluation of the precision needed for the computation allows our method to be competitive, and even be better *on average.*

It is important to have in mind that it is only on average. Indeed we mentioned that `nfroots` is not stable. First, there are number fields for which its running time explodes (with constant parameters), most likely because it cannot find a good prime ideal $\mathfrak{p}$ (with large ramification index), so reducing the basis of said ideal is not efficient. Moreover, we saw that `nfroots` does not cope properly with several situations, namely when the size of the polynomial defining the number field is large and over equations which are not split.

Finally our algorithm which takes advantage of a relative extension structure L/K can improve greatly the running times in some cases, namely when the relative degree $[L : K]$ is small and the degree of the equation considered is also small.

There are several things that could be explored further. Regarding the algebraic method `nfroots`, the following points could be assessed.

- One could try to find a heuristic way of evaluating the volume needed to ensure the correctness of the decoding, as we did for the precision of our method.
- It would be good to study the statistics concerning bad and good number fields for this method.
- Finally, one can wonder if there is a way of using the structure of a relative extension, similar to `RelativePolynomialRoots`.

Concerning our methods, we plan to work on the following directions.

- The main drawback of our method is the time needed to reduce our basis. It might be possible to speed-up this computation using the special form of our basis lattice (which is not random).
- Another direction would be to improve the decoding phase. Using Babaï's nearest plane algorithm (Alg. 8) instead of Kannan's embedding technique should save some steps for each decoding. It would require computing the GSO of the basis lattice, but this can be done during the reduction of said lattice. Indeed, Kannan's technique requires the use of LLL (Alg. 3) for each decoding,

which computes at least partially the GSO. Finally, using this version to decode could allow the improvement of the function `Check1Coord`, and make it more robust. These improvements would be of more benefit to our relative method `RelativePolynomialRoots`.

Chapter 5

Real Kummer extensions

Our contribution: As mentioned earlier, the SPIP is shown to be solvable with quantum computers over cyclotomic fields [34], and experimental data from [6] indicates that it is also the case over multiquadratic fields. However the methods are not similar. Over the last family, the algorithms use the strong structure of the set of subfields and the Galois group. The authors of [6] show that the unit group of high degree number fields can be computed in a reasonable amount of time (polynomial in the degree for a wide range of number fields), as well as generators of principal ideals. We generalised this work to real Kummer extensions of exponent a prime integer p, i.e. generated by p-th roots of integers. We also considered real Kummer extensions of $\mathbb{Q}$ with two exponents – generated by p-th and q-th roots of integers where p and q are prime integers – in order to break the structure and see if one can still solve the SPIP with a good probability. Moreover our implementation allows us to study the Log-unit lattice of these fields and classify them with respect to their security level. In this chapter we:

- describe algorithms to compute the unit group and solve the PIP of Kummer extensions;
- study the hardness of solving the SPIP over real Kummer fields using our implementation of these algorithms.

In particular we are able to evaluate the probability of success of shortening a generator with the Log-unit lattice, and study the quality of the basis obtained for this lattice. Moreover our implementation allows us to study high dimensional fields, and the data gathered highlights the need for considering such fields to draw conclusions on asymptotic behaviours. We therefore divide them in two categories: fields of degree less than 100 are called *low dimensional fields* and the others are called *high dimensional fields*. One can find in Table 5.1 a summary of the results obtained from our computations.

Table 5.1: Summary of the data obtained, where is shown: the probability of shortening a generator, the quality of the basis obtained, and for which category of fields the data is available

Field	Dimension achieved	Probability of shortening	Quality of the basis
Cyclotomics [34]	High	High	Good
Multiquadratics [6]	High	High	Good
Multicubics [64]	High	High	Good
NTRU Prime [11]	Low	High	Good
Simple Kummer of exponent p	Low	High	Good
Most of Kummer of exponent p	High	High	Good
Kummer with two exponents p, q	Low	High	Good
Kummer of degree p^2 defined by small integers (2,3)	High	Low	Bad

Conclusion: From the experimental data that we computed, general Kummer extensions of $\mathbb{Q}$ with only one exponent seem to show the same properties as multiquadratic fields. In particular we obtain high probabilities to retrieve private keys for a wide range of fields. However, within this family of fields, we are able to identify a subcategory over which solving the SPIP is more difficult than over other fields. Indeed the probability of success of solving the SPIP is smaller for fields with degree p^2 and defined by small integers, especially (2,3). Moreover the data computed on the key and the basis of the Log-unit lattice show that the quality of the basis obtained is not as good as over cyclotomic fields, and cannot be used to solve the SPIP over high dimensional fields. This can indicate than Kummer fields with degree p^2 and defined by small integers could be an alternative to cyclotomic fields for cryptography.

We also stress that these observations can be made only because we are able to compute the units of such fields for dimensions larger than 121, where significant differences between the type of fields truly appear. This leads us to think that one should always consider high dimensional fields (if the computational power at hand allows it) when studying problems such as the SPIP or the ISVP.

We were not able to compute as much data for Kummer extensions with two exponents, particularly for high dimensional fields. The data gathered seems to show that these extensions have the same global properties as Kummer fields with one exponent, despite behaviours which are less consistent. Better algorithms or implementations could be necessary to confirm it. One should remark that conclusions cannot be drawn for Kummer fields of degree p or NTRU Prime fields either, since we are able to do computations only for low dimensional number fields.

Related work: Biasse et al. generalised in [19] the approach of [6, 17, 64] to compute the unit group, S-units and the class group to normal fields. They give necessary and sufficient conditions for the existence of what they call *norm relations*, which allow the design of algorithms based on reduction to computations into subfields, in order to compute several number theoretical objects such as the maximal order, S-units and the class group. The algorithms we designed can be seen as a specialisation of their work. We still present them for simplicity and completeness purposes.

Future work: Further work can consists in studying other important tasks of computational number theory over these fields such as computing the class group and S-units. The authors of [17] provide polynomial time algorithms for these over multiquadratic fields. It could be possible to implement the algorithms presented and studied in [79, 9] to solve the *Ideal Shortest Vector Problem* (ISVP), and compare its performance over Kummer extensions and cyclotomic fields. Moreover the work of Biasse et al. [19] could be used to extend these considerations to a variety of other number fields.

Organisation of the chapter: The rest of the chapter is organised as follows, given L/K a Kummer extension of exponent p and degree p^2.

- In Section 5.1 we describe the number field extensions we are interested in, and we provide general recursive algorithms to compute $\mathcal{O}_L^\times$ and solve the PIP, following the framework of the ones in [6].

- In Section 5.2 we describe heuristic algorithms to compute $\mathcal{O}_L^\times$ and solve the PIP in $\mathrm{Poly}(\ln |D_L|)e^{\tilde{O}((\ln P)^{2/3})}$ and $\mathrm{Poly}(\ln |D_L|, \ln \mathrm{N}(I))e^{\tilde{O}((\ln P)^{2/3})}$ respectively, when L is generated over $\mathbb{Q}$ by p-th roots and q-th roots of integers, with p and q prime, where P depends only on the field K and $\mathrm{N}(I)$ is the algebraic norm of I. We also give details on some of the auxiliary procedures used in our implementation, such as extraction of p-th roots.

- We provide data gathered from our implementation in Section 5.3 and study the possibility of solving the SPIP over real Kummer extensions. In particular we are able to evaluate the probability that an attack is successful where Kannan's embedding technique is used for step 2., and compute several parameters linked to the basis of the Log-unit to evaluate its quality. We also compare these values to the ones obtained for cyclotomic fields and fields used in the NTRU Prime cryptosystem [11], which is has been proposed as an alternative to cyclotomic fields.

5.1 Structure of Kummer extensions

Notation. Given p a prime integer, we will denote by τ_p a generator of the Galois group of the cyclotomic field $\mathbb{Q}(\zeta_p)$.

Definition 5.1. A number field extension L/K is called a *Kummer extension of exponent* n if $\zeta_n \in K$ and there are elements $m_1, \ldots, m_r$ of K such that $L = K(\sqrt[n]{m_1}, \ldots, \sqrt[n]{m_r})$.

Remark 31. In our work we relax this definition to allow ζ_n to not belong to L. We will also only consider extensions of prime exponents p. First let us recall some facts and fix some notations about the structure of Kummer extensions, and $\mathrm{Hom}(L/K, \mathbb{C})$. We refer the reader interested in a more general and in-depth presentation of Kummer extensions to [30].

5.1.1 Complex field embeddings and Galois closure

Simple extensions:

Definition 5.2. Consider L/K an extension of number fields, and prime number p. Then L/K is called a *simple Kummer extension of exponent* p if there is $m \in K$ such that $\sqrt[p]{m} \notin K$ and $L = K(\sqrt[p]{m})$.

Proposition 5.1. *Consider $L = K(\sqrt[p]{m})$ a simple Kummer extension. Then the following properties are true.*

1. *L/K is a field extension of degree p.*
2. *The elements of the set $\mathrm{Hom}(L/K, \mathbb{C})$ can be fully described by their action on $\sqrt[p]{m}$ as $\sigma^{(i)} : \sqrt[p]{m} \longmapsto \zeta_p^i \sqrt[p]{m}, i \in [\![0, p-1]\!]$.*
3. *If $\zeta_p \in L$ then L/K is Galois. If $\zeta_p \notin K$ then the Galois closure of L/K is $\widetilde{L} = L(\zeta_p)$ and if p is odd then $\mathrm{Gal}(\widetilde{L}/K) = \langle \tau_p \rangle \ltimes \langle \sigma \rangle$ where σ is the extension of the complex embedding $\sigma^{(1)}$ which acts trivially on ζ_p. If p is 2 then L is Galois.*

Proposition 5.2. *Let $L = K(\sqrt[p]{m})$ be a simple Kummer extension of exponent p, and $n \in K$. Then $L = K(\sqrt[p]{n})$ if, and only if, there is $a \in K$ such that $n = ma^p$.*

General extensions:

The properties described for simple Kummer extensions can be extended to general extensions.

Proposition 5.3. *Consider $L = K(\sqrt[p]{m_1}, \ldots, \sqrt[p]{m_r})$ a Kummer extension. Then the following assertions are equivalent.*

1. *$[L : K] = p^r$;*
2. *$(\forall \alpha \in \mathbb{Z}^r)$, $m_1^{\alpha_1} m_2^{\alpha_2} \cdots m_r^{\alpha_r} \in (K^*)^p \iff \forall i \in [\![1, r]\!], p \mid \alpha_i$.*

Definition 5.3. Given a prime p, an integer $r \in \mathbb{N}^*$ and a sequence m of rational numbers $m_1, \ldots, m_r$ we will say that m is *p-reduced for K* if it satisfies the condition of Proposition 5.3.

Proposition 5.4. *Consider p a prime number and $L = K(\sqrt[p]{m_1}, \ldots, \sqrt[p]{m_r})$ a Kummer extension of exponent p. Then L can be described as $K(\sqrt[p]{n_1}, \ldots, \sqrt[p]{n_s})$ with $n = (n_1, \ldots, n_s)$ being a p-reduced sequence.*

From now on all Kummer extensions are considered to be generated by reduced sequences.

Notation. Consider $m = (m_1, \ldots, m_r) \in K^r$ such that $L = K(\sqrt[p]{m_1}, \ldots, \sqrt[p]{m_r})$ is an extension of degree p^r. For $i \in [\![1, r]\!]$ the field $L_{m_i} = K(\sqrt[p]{m_i})$ is a simple Kummer extension of K of exponent p. Given any $j \in [\![0, p-1]\!]$, write $\sigma_{m_i}^{(j)}$ the complex embeddings of L_{m_i} following the notation described previously and $\sigma_{m_i}^{j}$ the corresponding element of $\mathrm{Gal}(\widetilde{L_{m_i}}/K)$.

The simple extensions of a Kummer extension L/K are important as they allow the full description of L/K, as we will see later.

Proposition 5.5. *Consider L/K which satisfies the equivalent assertions of Proposition 5.3. Then the following assertions are true.*

1. *L/K has exactly $\frac{p^r-1}{2}$ simple subextensions of degree p over K and they are of the form $L_\alpha := L(\prod_{i=1}^r \sqrt[p]{m_i}^{\alpha_i})$ with $\alpha \in [\![0, p-1]\!]^r$. Moreover L_α and L_β are equal if, and only if, there is an integer λ such that $\alpha = \lambda \cdot \beta \pmod p$.*

2. *Any subextension of L/K can be written as $K(\sqrt[p]{M_1}, \ldots, \sqrt[p]{M_{r'}})$ where $0 \leqslant r' \leqslant r$ and $M_j = \prod_{i=1}^r \sqrt[p]{m_i}^{\alpha_i^{(j)}}$ with $\alpha^{(j)} \in [\![0, p-1]\!]^r$ for any $j \in [\![1, r']\!]$.*

The set of complex embeddings of L and the Galois group of $\widetilde{L}/\mathbb{Q}$ can also be fully described with the ones of the subfields L_{m_i}.

Proposition 5.6. *Consider L/K which satisfies the equivalent assertions of Proposition 5.3. Then the following assertions are true.*

- $\mathrm{Hom}(L/K, \mathbb{C}) \cong \bigotimes_{i=1}^r \mathrm{Hom}(L_{m_i}/K, \mathbb{C}) = \{\otimes_{i=1}^r \sigma_{m_i}^{(\beta_i)} \mid \beta \in [\![0, p-1]\!]^r\}$.

- *$L(\zeta_p)/K(\zeta_p)$ is abelian with Galois group isomorphic to $\langle \sigma_{m_1} \rangle \times \cdots \times \langle \sigma_{m_r} \rangle$; if $\zeta_p \in K$ then the previous extension is L/K.*

- *If $\zeta_p \notin K$ then $L(\zeta_p)/K$ is Galois with Galois group isomorphic to $\langle \tau_p \rangle \ltimes \langle \sigma_{m_1} \rangle \times \cdots \times \langle \sigma_{m_r} \rangle$.*

Notation. Given a tuple β we will write $\sigma^{(\beta)}$ the complex embedding $\otimes_{i=1}^r \sigma_{m_i}^{(\beta_i)}$ and σ^β its extension in $\mathrm{Gal}(\widetilde{K}/\mathbb{Q})$. Given a subset S of $\mathrm{Hom}(K, \mathbb{C})$ we will denote by $\widetilde{S}$ the subset of $\mathrm{Gal}(\widetilde{K}/\mathbb{Q})$ whose elements are the direct extension of elements of S.

Twisted Fourier transform:

Consider the relative Minkowski's embedding $\sigma_{L/K}$. In the case of Kummer extensions this link can be expressed as a *twisted Fourier transform.* A $K-$basis of L is $(\prod_{i=1}^{n} \sqrt[p]{m_i}^{\alpha_i})_{\alpha\in[\![0,p-1]\!]^n}$. Then the image of the basis element $b_\alpha = \prod_{i=1}^{n} \sqrt[p]{m_i}^{\alpha_i}$ by the complex embedding $\sigma^{(\beta)}$ is $\zeta_p^{(\alpha|\beta)} b_\alpha$. Let $x = \sum_{\alpha\in[\![0,p-1]\!]^n} x_\alpha b_\alpha$ an element of K. It can be expressed as the vector $\mathbf{y} = (x_\alpha b_\alpha)_{\alpha\in[\![0,p-1]\!]^n}$. The image of x by Minkowski's embedding is then the result of the multiplication of $\mathbf{y}$ by the matrix

$$\left[\zeta_p^{(\alpha|\beta)}\right]_{\substack{\alpha\in[\![0,p-1]\!]^n \\ \beta\in[\![0,p-1]\!]^n}} = \begin{bmatrix} 1 & 1 & 1 & \dots & 1 \\ 1 & \zeta_p & \zeta_p^2 & \dots & \zeta_p^{p-1} \\ \vdots & \vdots & \vdots & & \vdots \\ 1 & \zeta_p^{p-1} & \zeta_p^{2(p-1)} & \dots & \zeta_p^{(p-1)(p-1)} \end{bmatrix}$$

which is the matrix of a discrete Fourier transform. Therefore the vector $\mathbf{y}$ can be retrieved from the vector of complex embeddings of x by multiplying by the matrix of the inverse Fourier transform. This shows that a twisted Fourier transform links the vector of coefficients and the Minkowski embedding of x and one can efficiently go from one representation to another.

5.1.2 Structural result

The main brick of the efficient algorithms in [6, 17, 64] are structural results which express a power of any field element as a product of relative norms over several subfields. As the fields studied here are the generalisation of multiquadratic and multicubic fields, the same structural result appears.

Notation. Given an integer k and a subset S of a field F we will denote by S^k the set $\{x^k \mid x \in S\}$.

Proposition 5.7. *Let p be an odd prime number. Consider $L = K(\sqrt[p]{m_1}, \sqrt[p]{m_2})$ a Kummer extension such that $[L : K] = p^2$. Let u and v be two elements of* $\mathrm{Hom}(L/K, \mathbb{C})$ *such that their extensions $\tilde{u}$ and $\tilde{v}$ are independent. Then the following properties are true.*

1. $L^p \subset L^u L^{uv} \dots L^{u^{p-1}v} L^v$;
2. $(\mathcal{O}_L^\times)^p \subset \mathcal{O}_{L^u}^\times \mathcal{O}_{L^{uv}}^\times \dots \mathcal{O}_{L^{u^{p-1}v}}^\times \mathcal{O}_{L^v}^\times$.

Proof. The proof is similar to the ones of the corresponding results in [6]. Let $x \in L^*$ and u, v be two elements of $\mathrm{Hom}(L/K, \mathbb{C})$ such that $\tilde{u}$ and $\tilde{v}$ are independent. Then we have:

$$x^p = \frac{\prod_{i=0}^{p-1}\prod_{j=0}^{p-1}(\tilde{u}\tilde{v}^i)^j(x)}{\prod_{i=0}^{p-1}\prod_{j=1}^{p-1}(\tilde{u}\tilde{v}^i)^j(x)} = \frac{\prod_{i=0}^{p-1} \mathrm{N}_{\tilde{L}/\tilde{L}^{\tilde{u}\tilde{v}^i}}(x)}{\prod_{j=1}^{p-1} \tilde{u}^j \left(\prod_{i=0}^{p-1} \tilde{v}^{ij}(x)\right)}. \tag{5.1}$$

For any $j \in [\![1, p-1]\!]$ the sets $\{i \mid i \in [\![0, p-1]\!]\}$ and $\{ij \mid i \in [\![0, p-1]\!]\}$ are the same, therefore:

$$x^p = \frac{\prod_{i=0}^{p-1} \mathrm{N}_{\tilde{L}/\tilde{L}^{\tilde{u}\tilde{v}^i}}(x)}{\prod_{j=1}^{p-1} \tilde{u}^j \left(\mathrm{N}_{\tilde{K}/\tilde{L}^{\tilde{v}}}(x)\right)} = \frac{\prod_{i=0}^{p-1} \mathrm{N}_{\tilde{L}/\tilde{L}^{\tilde{u}\tilde{v}^i}}(x)}{\mathrm{N}_{\tilde{L}/\tilde{L}^{\tilde{v}}} \left(\prod_{j=1}^{p-1} \tilde{u}^j(x)\right)}. \tag{5.2}$$

Now let us assume first that $\zeta_p \in K$. Then L/K is Galois, $u = \tilde{u}$, $v = \tilde{v}$ and Equation (5.2) can be written as

$$x^p = \frac{\prod_{i=0}^{p-1} \mathrm{N}_{L/L^{uv^i}}(x)}{\mathrm{N}_{L/L^v} \left(\prod_{j=1}^{p-1} u^j(x)\right)}.$$

For any morphism w the relative norm $\mathrm{N}_{L/L^w}(x)$ is an element of L^w and if x is an integer (resp. a unit) then its relative norms are also integers (resp. units). Therefore one has

$$x^p \in L^u L^{uv} \dots L^{u^{p-1}v} L^v \tag{5.3}$$

and if $x \in \mathcal{O}_L^\times$ we can replace the fields by their unit groups. Finally 5.3 is true for any x different from 0, but it is obviously correct for 0 as well, which proves that the claimed results are true if $\zeta_p \in K$. Now assume that $\zeta_p \notin K$. Then for all $i, j \in [\![0, p-1]\!]$ the action of $(\tilde{u}\tilde{v}^i)$ on x is the same as the action of $u^{(j)} \otimes v^{(j)}$. Therefore for all $i \in [\![0, p-1]\!]$ the relative norm $\mathrm{N}_{\tilde{L}/\tilde{L}^{\tilde{u}\tilde{v}^i}}(x)$ is equal to $\mathrm{N}_{L/L^{u \otimes v^{(i)}}}(x)$ which is an element of $K^{u \otimes v^{(i)}}$. The statements about integers and units are again true. In Equation (5.2) we know that x^p belongs to L as well as the numerator, so the denominator belongs to $\tilde{L}^{\tilde{v}} \cap L = L^v$. Finally the claimed results are also true if $\zeta_p \notin K$. $\square$

If one removes zero from all of the sets, then the set inclusions in Proposition 5.7 become group inclusions. In fact remark that $U = \mathcal{O}_{L^u}^\times \mathcal{O}_{L^{uv}}^\times \dots \mathcal{O}_{L^{u^{p-1}v}}^\times \mathcal{O}_{L^v}^\times$ is a full-rank subgroup of $O_L^\times$ such that $(O_L^\times)^p < U < O_L^\times$.

Corollary 5.1. *Let p be an odd prime number. Consider $L = K(\sqrt[p]{m_1}, \dots, \sqrt[p]{m_r})$ a Kummer extension such that $[L : K] = p^r$. Then the following hold:*

1. $L^{p^{r-1}} \subset \prod_\alpha L_\alpha$;
2. $(\mathcal{O}_L^\times)^{p^{r-1}} < \prod_\alpha \mathcal{O}_{L_\alpha}^\times$.

Definition 5.4. Given a Kummer extension $L = K(\sqrt[p]{m_1}, \dots, \sqrt[p]{m_r})$ we will call *simple units of L/K* and denote by $\mathrm{SU}(L/K)$ the subgroup of $\mathcal{O}_L^\times$ defined by the following equation:

$$\mathrm{SU}(L/K) = \prod_\alpha \mathcal{O}_{L_\alpha}^\times.$$

5.1.3 General algorithms

The general procedures follow the same shape as the ones in [6, 64]. The algorithms rely on two tasks:

1. detecting non trivial p-powers in the subgroup of L^* generated by a given set S;

2. computing the roots of the detected powers.

We will write `DetectPowers` for the first procedure and `ElementsFromPower` the second. The procedure which finds a basis of a subgroup of $\mathcal{O}_L^\times$ given a generating family by reducing through the Log-embedding will be written `BasisFromGeneratingSet`. They will be described as general procedures in this section and in Algorithms 30 and 31, but we will describe more thoroughly in Subsection 5.2.4 how we implemented them in the case of real Kummer extensions.

Detecting powers

In the general case one can use the *Saturation technique* mentioned in [12, 17]. For any prime ideal $\mathfrak{Q}$ such that $p \mid \mathrm{N}(\mathfrak{Q}) - 1$ one can construct a "character" $\chi_{\mathfrak{Q}} : S \to F_{\mathfrak{Q}}^*/(F_{\mathfrak{Q}}^*)^p$ where $F_{\mathfrak{Q}}$ is the residue class field. If $u \in S$ is a $p-$power then $\chi_{\mathfrak{Q}}(u)$ is trivial but the inverse is not true in general. In order to detect proper powers, one only has to intersect $\ker \chi_{\mathfrak{Q}}$ for sufficiently many $\mathfrak{Q}$. If $r = |S|$ then the rank of $S/(S \cap (L^*)^p)$ is $r' \leqslant r$. If we consider the $\chi_{\mathfrak{Q}}$ to be uniformly distributed in the dual of $S/(S \cap (L^*)^p$ then one can adapt Lemma 8.2 of [25] to show that $r' + s$ characters generate the dual – so the intersection of their kernels is $S \cap (L^*)^p$ – with probability at least $1 - p^{-s}$. If B is a bound on the size of the basis elements generating S then `DetectPowers` can be computed in $\mathrm{Poly}(B, \max_{\mathfrak{Q}} \ln(\mathrm{N}(\mathfrak{Q})), r'+s)$. For the case of multiquadratic fields, the authors of [6] give a practical way of computing these characters and a precise analysis of the cost of the overall procedure, that we refer to. It can be generalised to the real Kummer extensions that we will study below.

Computing units

Algorithm 30 describes the recursive algorithm which can be used to compute the unit group of a Kummer extension L/K. It is the generalisation of the ones for multiquadratic fields or multicubic fields presented in [6, 64]. We denote by `UnitGroup` the general procedure computing the unit group of a number field as input. Depending on the number field, different algorithms can be used.

Algorithm 30 Compute the unit group of a Kummer extension L/K of exponents p. – `KE_Units`

```
Require: A Kummer extension $L = K(\sqrt[p]{m_1}, \ldots, \sqrt[p]{m_r})$.
Ensure: A basis of the unit group $\mathcal{O}_L^\times$
 1: if $(r = 1)$ then
 2:     return UnitGroup($L$).
 3: else
 4:     Choose $u, v$ two independent elements of $\widetilde{\mathrm{Hom}(L/K)}$.
 5:     Recursively compute a basis of $U = \mathcal{O}_{L^u}^\times \mathcal{O}_{L^{uv}}^\times \ldots \mathcal{O}_{L^{u^{p-1}v}}^\times \mathcal{O}_{L^v}^\times$
 6:     $V \leftarrow$ DetectPowers($U, p$)
 7:     $V \leftarrow$ ElementsFromPower($V, p$)
 8:     $U \leftarrow$ BasisFromGeneratingSet($\langle U, V\rangle$)
 9:     return $U$
10: end if
```

Proposition 5.8. *Given a Kummer extension L/K, Algorithm 30 is correct, and returns a basis with probability at least $1-p^{-[L:\mathbb{Q}]}$ provided that one computes* Poly($[L : \mathbb{Q}]$) *characters for each subfield L' reached during the algorithm, and that the characters are uniformly distributed.*

Proof. By Proposition 5.7 the subgroup U of step 5 is such that $(\mathcal{O}_L^\times)^p < U < \mathcal{O}_L^\times$. Therefore $\mathcal{O}_L^\times$ is isomorphic to $U \times \frac{U \cap (\mathcal{O}_L^\times)^p}{U^p}$. The only part left to verify is the validity of the recursion. Clearly each of the fields L_i is a Kummer extension of K but such that $[L_i : K] = p^{r-1}$ so the algorithm can be applied to it. Since the dimension is strictly decreasing, after $r-1$ recursion steps the algorithm reaches simple extensions of L, i.e. the case $r = 1$. Then following the analysis done during the proof of Theorem 4.6 in [17], the probability of success is at least $(1-p^{-(s)})^{[L:K]} \geqslant 1-2^{[L:K]}/p^s$ where s characters are computed for each field. Therefore if $s \in$ Poly($[L : \mathbb{Q}]$) one can reach the desired probability of success. □

We will only do an analysis of complexity for the real Kummer extensions that we consider latter.

Solving the Principal Ideal Problem:

In order to solve the Principal Ideal Problem, i.e. retrieve a generator of a principal ideal I, we do as follows. First compute the relative ideal norm of I over subfields of K. Then recursively compute a generator of these ideals. By using Proposition 5.7 it is easy to see that a combination of these elements is a generator h of I^p (see [6, 64]). The final steps are finding a unit u such that hu is a p-power and computing its p-th root. This is summarised in Algorithm 31. The relative norm computations are polynomial with respect to the dimension and the size of the ideal. Moreover

Algorithms 30 and 31 are very similar in shape. One can easily deduce that the validity and complexity analysis are also similar. One can see that Algorithm 31 will go through subextensions of L/K down to simple subextensions, where it will call the procedure `Generator` to solve the PIP. As for `UnitGroup`, different algorithms depending on the field can be used. In the general case one might need to compute the class group of the field so one cannot hope better than a sub-exponential complexity.

Algorithm 31 Solve the PIP in a Kummer extension of exponent p – `KE_PIP`

```
Require: A principal ideal $I$ of a Kummer extension $L = K(\sqrt[p]{m_1}, \dots, \sqrt[p]{m_r})$, the unit group $\mathcal{O}_l^\times$
Ensure: A generator $g$ of $I$.
1: if $(r = 1)$ then
2:     return Generator(I).
3: else
4:     Choose $u, v$ two independent elements of $\widetilde{\mathrm{Hom}(L/K)}$.
5:     Recursively compute generators of $\mathrm{N}_{L^u}(I), \mathrm{N}_{L^{uv}}(I), \dots, \mathrm{N}_{L^{u^{p-1}v}}(I), \mathrm{N}_{L^v}(I)$ and use Equation 5.2 to compute $h$ a generator of $I^p$.
6:     $h \leftarrow$ DetectPowers($\mathcal{O}_L^\times \cup \{h\}, p$).
7:     return ElementsFromPower($h, p$).
8: end if
```

5.2 Real Kummer extensions

In this section we will focus on real Kummer extensions. More precisely we are interested in fields of the form $L = K(\sqrt[p]{m_1}, \dots, \sqrt[p]{m_r})$ with $K = \mathbb{Q}(\sqrt[q]{n_1}, \dots, \sqrt[q]{n_s})$ with p and q prime integers. We always consider $\sqrt[p]{m_i}$ and $\sqrt[q]{n_j}$ to be the real roots of the polynomials $X^p - m_i$ and $X^q - n_j$ respectively. Then K is a real Kummer extension of $\mathbb{Q}$ of exponent q and L is a real Kummer extension of K of exponent p. We will call such fields *real Kummer extensions of exponents* p, q. For the particular case of $s = 0$ the field K is $\mathbb{Q}$. Multiquadratic and multicubic fields studied in [6, 64] fall in this category. We will call these fields *real Kummer extensions with exponent* p.

5.2.1 Field structure

First let us describe the structure of considered extensions.

Proposition 5.9. *Consider a Kummer extension L/K as before. Then the following assertions are true.*

1. $\mathrm{Hom}(L, \mathbb{C}) \cong \bigotimes_{i=1}^r \mathrm{Hom}(L_{m_i}, \mathbb{C}) \bigotimes_{j=1}^s \mathrm{Hom}(K_{n_j}, \mathbb{C}) = \{\otimes_{i=1}^r \sigma_{m_i}^{(\beta_i)} \otimes_{j=1}^s \sigma_{n_j}^{(\gamma_j)} \mid \beta \in [\![0, p-1]\!]^r, \gamma \in [\![0, q-1]\!]^s\}$.

2. *$L(\zeta_p)/K(\zeta_p)$ is abelian with Galois group isomorphic to $\langle\sigma_{m_1}\rangle \times \cdots \times \langle\sigma_{m_r}\rangle$.*

3. *$L(\zeta_p)/\mathbb{Q}(\zeta_p)$ is abelian with Galois group isomorphic to $\langle\sigma_{m_1}\rangle \times \cdots \times \langle\sigma_{m_r}\rangle \times \langle\sigma_{n_1}\rangle \times \cdots \times \langle\sigma_{n_s}\rangle$.*

4. *$L(\zeta_p)/K$ is Galois with Galois group isomorphic to a subgroup of $\tau_p \ltimes \langle\sigma_{m_1}\rangle \times \cdots \times \langle\sigma_{m_r}\rangle$.*

Notation. Given tuples β and γ we will denote by $\sigma^{(\gamma,\beta)}$ the complex embedding $\otimes_{i=1}^{r}\sigma_{m_i}^{(\beta_i)} \otimes_{j=1}^{s} \sigma_{n_j}^{(\gamma_j)}$. Given a subset S of $\mathrm{Hom}(L,\mathbb{C})$ we will write $\widetilde{S}$ for the subset of $\mathrm{Gal}(\widetilde{L}/\mathbb{Q})$ whose elements are the direct extension of elements of S.

Galois correspondence: The Galois correspondence shows there is a bijection between the set of subfields of $\widetilde{L}/K$ and the subgroups of its Galois group. Moreover for the fields which are considered, a subextension M/K of $\widetilde{L}/K$ is a subextension of L/K if, and only if, the group associated by the Galois correspondence contains τ_p. Therefore when considering a subextension M/K of L/K and their sets of complex embeddings it is equivalent to consider their extensions $\widetilde{\mathrm{Hom}(M/K)}$ and $\widetilde{\mathrm{Hom}(L/K)}$. We can therefore "forget" about the complex part.

Twisted Fourier transform: Given an Kummer extension L/K of exponents p, q one can see that a twisted Fourier transform links the coefficient of any element of L to their coefficients in the $\mathbb{Q}$-basis. This transformation is expressed as the tensor product of the one for L/K and the one for $K/\mathbb{Q}$.

5.2.2 Basis and discriminant

We will establish some facts about $\mathbb{Q}$-bases of real Kummer extensions considered, as well as their discriminants. Knowing the discriminant of a number field is important as it is a measure of the size of the ring of integers, and one usually express complexities of algorithms in terms of the discriminant. It can be difficult to find a formulae for it. However it can be done over multiquadratic fields.

Moreover we wish to exhibit a simple $\mathbb{Q}-$basis of real Kummer extensions L such that $[L : \mathbb{Q}]\mathcal{O}_L$ is included in the order generated by this basis. Again it can be found over multiquadratic fields.

Extensions with one exponent

First we will study fields of the form $K = \mathbb{Q}(\sqrt[p]{m_1}, \ldots, \sqrt[p]{m_r})$.

Notation. Given a tuple $m = (m_1, \ldots, m_r)$, we will write $\mathcal{P}(m)$ the set $\{p \in \mathcal{P}, p \mid \prod_{i=1}^{r} m_i\}$. Given $m \in \mathbb{Q}$ and an integer n we will denote by $PF(m, n)$

the rational number $\prod_{p\in\mathcal{P}} m^{v_p(m) \pmod n}$. Similarly if $m \in \mathbb{Q}^r$ then $PF(m,n) = (PF(m_1,n),\ldots,PF(m_r,n))$. We extend $PF(\cdot,p)$ to elements in $\mathbb{Q}^{1/p}$ and sequences in $\mathbb{Q}^{1/p}$ with $PF(x,p) = PF(x^p,p)^{1/p}$. Finally, given a tuple $m \in \mathbb{Q}^r$ and $\alpha \in \mathbb{Z}^r$, we will write m^α to designate the product $\prod_{i\in[\![1,r]\!]} m_i^{\alpha_i}$.

A canonical $\mathbb{Q}$−basis of K One can define two fairly natural bases of K. One has already been mentioned earlier.

Definition 5.5. Let $K = \mathbb{Q}(\sqrt[p]{m_1},\ldots,\sqrt[p]{m_r})$ be a real Kummer field. Then the *naive basis of K relative to m* is $(\prod_{i=1}^r m_i^{\alpha_i/p})_{\alpha\in[\![0,p-1]\!]^r}$. It will be denoted by $\mathfrak{B}(p,m)$. The *power-free basis of K relative to m* is $PF(\mathfrak{B}(p,m),p)$. It will be denoted by $\mathfrak{IB}(p,m)$.

Remark 32. Both bases were considered in several work on Kummer fields such as [17, 100].

The first property that can be proven is that $\mathfrak{IB}(p,m)$ is somehow independent of the choice of m.

Lemma 5.1. *Let K be a real Kummer field. Consider m and n two sequences defining K. Then $\mathfrak{IB}(p,m)$ and $\mathfrak{IB}(p,n)$ are equal as sets.*

Proof. Consider $q \in \mathcal{P}(m)$. First let us prove that if $q \notin \mathcal{P}(n)$ then $v_q(m^\alpha) \equiv 0 \pmod p$ for all $\alpha \in [\![0,p-1]\!]^r$. Let us fix such α. Since m and n define the same field K, one can use the simple subfields and conclude that $\mathbb{Q}(\sqrt[p]{m^\alpha}) = \mathbb{Q}(\sqrt[p]{n^\beta})$ for some β. This is equivalent to $m^\alpha = n^{j\beta}a^p$ for some $j \in [\![0,p-1]\!]$ and $a \in \mathbb{Q}$. Then we obtain the equality

$$v_q(m^\alpha) = \sum_{i=1}^r \alpha_i v_q(m_i) = \sum_{i=1}^r j\beta_i v_q(n_i) + p v_q(a) \tag{5.4}$$

and taking it modulo p gives $\sum_{i=1}^r \alpha_i v_q(m_i) = 0 \pmod p$, since $v_q(n_i) = 0$, for all $i \in [\![1,r]\!]$. This is true for all α. Now, fix i_0 such that $q \mid m_{i_0}$. Then, the equality applied with α such that $\alpha_i = 1$ if $i = i_0$ and $\alpha_i = 0$ otherwise gives $v_q(m_{i_0}) = 0 \bmod p$. Thus we obtain that none of $q \in \mathcal{P}(m) \cup \mathcal{P}(n) \setminus (\mathcal{P}(m)\cap\mathcal{P}(n))$ can be found in $\mathfrak{IB}(p,m)$ nor $\mathfrak{IB}(p,m)$.

Now let us consider only $q \in \mathcal{P}(m)\cap\mathcal{P}(n)$. Let $\alpha \in \mathbb{F}_p^r \setminus \{0\}$. Then for all $q \in \mathcal{P}(m)\cap\mathcal{P}(n)$ and all $j \in [\![1,p-1]\!]$, $(v_q(m^{j\alpha})) = jv_q(m^\alpha) \pmod p$. Following Equation (5.4), if β is such that n^β defines the same simple field as m^α, then $(v_q(m^\alpha))_q = j(v_q(n^\beta))_q$ for some $j \in [\![1,p-1]\!]$. Therefore the sets $\{(v_q(m^{j\alpha}))_q \mid j \in [\![1,p-1]\!]\}$ and $\{(v_q(n^{j\beta}))_q \mid j \in [\![1,p-1]\!]\}$ are identical. Finally if α and α'

define distinct simple subfields of K then $(v_q(m^\alpha))_q$ and $(v_q(m^{\alpha'}))_q$ are not colinear modulo p. □

The equality given by Lemma 5.1 shows that the set of power-free basis of a real Kummer field is a canonical choice of a $\mathbb{Q}$-basis of K.

Definition 5.6. Let K be a real Kummer field with one exponent p defined by a sequence m. *The power-free basis* of K is the unordered sequence set $\mathfrak{IB}(p, m)$. It will be denoted $\mathfrak{IB}(K)$.

Now let us prove another simple result on defining sequences, that will be used later.

Lemma 5.2. *Let $m \in \mathbb{Q}^r$ be a sequence defining a real Kummer extension K with one exponent p, and $i_0 \in [\![1, r]\!]$. Consider $q \in \mathcal{P}(m)$ such that $\exists i \in [\![1, r]\!], v_q(m_i) \not\equiv 0 \pmod p$. Then there is $m' \in \mathbb{Q}^r$ defining K such that*

$$\forall i \in [\![1, r]\!], q \mid m'_i \iff i = i_0.$$

Proof. One can always assume $v_q(m_{i_0}) \not\equiv 0 \pmod 0$, modulo a permutation on m. Then fix $m'_{i_0} = PF(m_{i_0}, p)$, and $m'_i = PF(m_i, p)$ for all $i \in [\![1, r]\!]$ such that $q \nmid m_i$. Finally consider $i \in [\![1, r]\!]$ such that $q \mid m_i$. Let $e_i \geqslant 0$ such that $e_i \equiv -v_q(m_i)v_q(m_{i_0})^{-1} \pmod p$. Then fix $m'_i = PF(m_i m_{i_0}^{e_i}, p)$. □

We will now determine the discriminant of $\mathfrak{IB}(K)$.

Theorem 5.1. *Let $K = \mathbb{Q}(\sqrt[p]{m_1}, \dots, \sqrt[p]{m_r})$ be a real Kummer field, and q a prime integer. Moreover write $b = \delta_{(p \in \mathcal{P}(m))}$. Then the discriminant $D_K(\mathfrak{IB}(K))$ satisfies the following:*

$$v_q(D_K(\mathfrak{IB}(K))) = \begin{cases} p^{r-1}(p-1), & \text{if } q \in \mathcal{P}(m) \setminus \{p\}, \\ p^{r-1}(p-1) \times b + rp^r, & \text{if } q = p. \end{cases} \tag{5.5}$$

Proof. Given a sequence m such that $K = \mathbb{Q}(\sqrt[p]{m_1}, \dots, \sqrt[p]{m_r})$, let us denote by M_m the matrix $(\sigma_i(\mathfrak{IB}(p, m)_j))_{i,j}$, where as usual $\sigma_i : \sqrt[p]{m_i} \longmapsto \zeta_p \sqrt[p]{m_i}$. Moreover we will write M_B the matrix $[\sigma_i(b_j)]_{i,j}$ for any $\mathbb{Q}-$basis $B = (b_1, \dots, b_{p^r})$.

First remark that $D_K(\mathfrak{IB}(K)) = D_K(\mathfrak{IB}(p, m))$ for any sequence m defining K. Indeed, considering different sequences amounts to applying permutations on the rows and columns of a fixed matrix M_m. Moreover, if $m' = (m_1, \dots, m_{r-1})$ then $\mathfrak{B}(p, m) = \mathfrak{B}(m_r) \otimes \mathfrak{B}(m')$ and $\mathfrak{IB}(p, m) = PF(\mathfrak{IB}(p,m_r) \otimes \mathfrak{IB}(p,m'), p)$. Let us denote by $\mathfrak{B}$ the basis $\mathfrak{IB}(p,m_r) \otimes \mathfrak{IB}(p,m')$.

Now let us start with the proof per se. Let m be a defining sequence of K. One can assume it is p-reduced and composed of integers. Let $q \in \mathcal{P}(m) \setminus \{p\}$. Following Lemma 5.1 and Lemma 5.2, one can also assume that $q \mid m_r$ and for all $i < r, q \nmid m_i$. We mentioned that $\mathfrak{IB}(p, m) = PF(\mathfrak{IB}(m_r) \otimes \mathfrak{IB}(m'), p)$. The action of PF amounts to dividing elements of the basis by an integer. Let us denote by $c_1, \dots, c_{p^r}$ these integers. Since q divides only m_r and $\mathfrak{IB}(m_r)$ is already reduced, none of said coefficients is divided by q. Now remark that $M_m = M_{\mathfrak{IB}(p,m)} = [\frac{C_1(M_{\mathfrak{B}})}{c_1}|\dots|\frac{C_{p^r}(M_{\mathfrak{B}})}{c_{p^r}}]$, where $C_j(M_{\mathfrak{B}})$ is the j-th column of $M_{\mathfrak{B}}$. Therefore we have $\det M_m = \frac{\det M_{\mathfrak{B}}}{c_1 c_2 \dots c_{p^r}}$. Consequently we obtain $v_q(\det M_m) = v_q(\det M_{\mathfrak{B}})$, and we can consider the discriminant of the basis $\mathfrak{B}$. Now let us denote by $b_1, \dots, b_p$ the elements of $\mathfrak{IB}(m_r)$. Then we have $\mathfrak{B} = [\mathfrak{IB}(m')b_1|\mathfrak{IB}(m')b_2|\dots|\mathfrak{IB}(m')b_p]$ and for $\beta \in [\![0, p-1]\!]^r$, $\sigma^{(\beta)} = \sigma_1^{(\beta_1)} \otimes \dots \otimes \sigma_r^{(\beta_r)}$ acts on $\mathfrak{IB}(m')b_i$ as

$$\sigma_1^{(\beta_1)} \otimes \dots \otimes \sigma_{r-1}^{(\beta_{r-1})} (\mathfrak{IB}(m'))\, \sigma_r^{(\beta_r)}(b_i).$$

Thus we obtain that $M_{\mathfrak{B}}$ is equal to

$$\left[\begin{array}{c|c|c|c}
b_1 M_{\mathfrak{IB}(m')} & b_2 M_{\mathfrak{IB}(m')} & \dots & b_p M_{\mathfrak{IB}(m')} \\ \hline
\sigma_r^{(1)}(b_1) M_{\mathfrak{IB}(m')} & \sigma_r^{(1)}(b_2) M_{\mathfrak{IB}(m')} & \dots & \sigma_r^{(1)}(b_p) M_{\mathfrak{IB}(m')} \\ \hline
\vdots & \vdots & & \vdots \\ \hline
\sigma_r^{(p-1)}(b_1) M_{\mathfrak{IB}(m')} & \sigma_r^{(p-1)}(b_2) M_{\mathfrak{IB}(m')} & \dots & \sigma_r^{(p-1)}(b_p) M_{\mathfrak{IB}(m')}
\end{array}\right],$$

which is $M_{m_r} \otimes M_{m'}$. Therefore, we have $\det M_{\mathfrak{B}} = \det M_{m_r}^{p^{r-1}} \det M_{m'}^{p}$, and

$$v_q(D_K(\mathfrak{IB}(K))) = p^{r-1} v_q(\det M_{m_r}^2) + p v_q(\det M_{m'}^2).$$

Westlund showed that $v_q(\det M_{m_r}^2) = p - 1$ and since $q \nmid m_i$ for all $i \in [\![1, r-1]\!]$ one has $v_q(\det M_{m'}^2) = 0$, by induction and remarking that $v_q(\det M'_m) = v_q(\det M_{m_1} \otimes \dots \otimes M_{m_{r-1}})$ [100]. Finally we obtain $v_q(D_K(\mathfrak{IB}(K))) = (p-1)p^{r-1}$.

Now consider $q = p$. As before we have $v_p(\det M_m^2) = v_p(\det M_{m_1}^2 \otimes \dots \otimes \det M_{m_r}^2)$, and $\det M_1^2 \otimes \det M_r^2 = \prod_{i=1}^r (\det M_{m_i}^2)^{p^{r-1}}$. If $p \notin \mathcal{P}(m)$ then $v_p(\det M_{m_i}^2) = p$ for all $i \in [\![1, r]\!]$ [100], so $v_p(D_K(\mathfrak{IB}(K))) = \sum_{i=1}^r p^{r-1} v_p(\det M_{m_i}^2) = rp^r$. If $p \in \mathcal{P}(m)$ then $v_p(\det M_{m_i}^2) = p$ for all $i \in [\![1, r-1]\!]$ and $v_p(\det M_{m_r}^2) = 2p - 1$ [100]. Therefore we have $v_p(D_K(\mathfrak{IB}(K))) = \sum_{i=1}^{r-1} p^{r-1} v_p(\det M_i^2) + p^{r-1}(2p-1) = rp^r + p^{r-1}(p-1)$. □

We established that $\mathfrak{IB}(K)$ is a fairly canonical basis for a real Kummer field K, and determined its discriminant. We will show that the order it generates contains $[K : \mathbb{Q}]\mathcal{O}_K$. For this we will study the discriminant of K. Indeed recall that we have the following result. Given $\mathcal{O}_1$ and $\mathcal{O}_2$ two orders of a number field, then $\mathcal{O}_1 < \mathcal{O}_2 \iff D_K(O_2) \mid D_K(O_1)$.

Lemma 5.3. *Let $K \in \mathbb{Q}(\sqrt[p]{m_1}, \dots, \sqrt[p]{m_r})$ be a real Kummer extension defined by a p-reduced sequence m. Then q ramifies in $K/\mathbb{Q}$ if, and only if, $q \in \mathcal{P}(m) \cup \{p\}$.*

Proof. We know that $K = \otimes_{i=1}^{r} \mathbb{Q}(\sqrt[p]{m_i})$, and given two linearly disjoints fields K_1 and K_2, the discriminant of their compositum $D_{K_1K_2}$ divides $D_{K_1}^{[K_2:\mathbb{Q}]} D_{K_2}^{[K_1:\mathbb{Q}]}$. Therefore we have $D_K \mid \prod_{i=1}^{r} D(\mathbb{Q}(\sqrt[p]{m_i}))^{p^{r-1}}$. Following Westlund [100], $q \mid D(\mathbb{Q}(\sqrt[p]{m_i}))$ if, and only if, $q \in \mathcal{P}(m_i) \cup \{p\}$. $\square$

In order to study the $q-$valuation of D_K, we will study the splitting of q in $K/\mathbb{Q}$. A similar approach has been done over multiquadratic fields [91] and bicubic fields [27]. We will use some results over dihedral groups, which are stated and proved in Appendix B.

Splitting of primes in K To study the splitting of primes we will use the different of the extensions (Def. 2.43). Westlund established the splitting for simple fields.

Proposition 5.10 (Westlund [100])**.** *Let $K = \mathbb{Q}(\sqrt[p]{m})$ be a simple Kummer extension and q a prime integer. Then one has the following possibilities:*

1. *$q \neq p$ and $q \mid m \implies (q) = \mathfrak{q}^p$;*
2. *$p \mid m \implies (p) = \mathfrak{p}^p$;*
3. *$p \nmid m$ and $m^{p-1} \equiv 1 \bmod p^2 \implies (p) = \mathfrak{p}^{p-1}\mathfrak{q}$;*
4. *$p \nmid m$ and $m^{p-1} \not\equiv 1 \bmod p^2 \implies (p) = \mathfrak{p}^p$.*

One can see that for simple Kummer field, the splitting of p depends on a condition satisfied by m: whether $m^{p-1} \equiv 1 \bmod p$ or not. The splitting of primes in a general number field K will then be influenced by their splitting in the simple subfields of K. We can identify different types of Kummer fields.

Lemma 5.4. *Let $K = \mathbb{Q}(\sqrt[p]{m_1}, \sqrt[p]{m_2})$ be a Kummer extension of degree p^2 such that $m_i \not\equiv 0 \bmod p$ and $m_i \not\equiv 1 \bmod p^2$, for $i \in \{1,2\}$. Then one can find m' a sequence defining K such that $m'_2 \equiv 1 \bmod p^2$.*

Proof. For $i \in \{1,2\}$, since $m_i \not\equiv 0 \bmod p$ then m_i can be seen as an element of $G = (\frac{\mathbb{Z}}{p^2\mathbb{Z}})^{\times}$. Moreover the order of m_i in G is p or $p(p-1)$, and we want to prove that we can find a defining sequence m' such that $o(m'_2) \mid p-1$. The group G is isomorphic to $\frac{\mathbb{Z}}{(p-1)\mathbb{Z}} \times \frac{\mathbb{Z}}{p\mathbb{Z}}$. Let us denote by $\phi = (\phi_1, \phi_2)$ this isomorphism. Then $\phi(m_i) = (\phi_1(m_i), \phi_2(m_2))$ with $\phi_2(m_i) \neq 0$. Let m' defined by $m'_1 = m_1$ and $m'_2 = m_1 m_2^k$ with $k \in [\![1, p-1]\!]$ such that $k\phi_2(m_2) = -\phi_2(m_1)$. Then one has $\phi_2(m'_2) = 0$ so $o(m'_2) \mid p-1$ in G. Clearly m' also defines K. $\square$

Using Lemma 5.4, we obtain only a few possibilities for general real Kummer extensions.

Proposition 5.11. *Let K be a real Kummer extension of degree p^r for an integer $r \geqslant 1$. Then one can find a sequence $m = (m_1, \dots, m_r)$ defining K and satisfying one of the following properties.*

1. *$p \notin \mathcal{P}(m)$ and $\forall i \in [\![1, r]\!], m_i^{p-1} \equiv 1 \bmod p^2$.*
2. *$p \notin \mathcal{P}(m)$, $m_1^{p-1} \not\equiv 1 \bmod p^2$ and $\forall i \in [\![2, r]\!], m_i^{p-1} \equiv 1 \bmod p^2$.*
3. *$p \mid m_1$ and $\forall i \in [\![2, r]\!], m_i^{p-1} \equiv 1 \bmod p^2$.*
4. *$p \mid m_1$, $m_2^{p-1} \not\equiv 1 \bmod p^2$ and $\forall i \in [\![3, r]\!], m_i^{p-1} \equiv 1 \bmod p^2$.*

Proof. This is just an application of Lemma 5.2 and Lemma 5.4. □

Now we can express how primes split in K depending on which type of Kummer field it is. However remark than only the splitting of p will be influenced by the types identified in Proposition 5.11. Therefore, let us start by $q \neq p$.

Proposition 5.12. *Consider $K = \mathbb{Q}(\sqrt[p]{m_1}, \dots, \sqrt[p]{m_r})$ a real Kummer extension with one exponent, and $q \in \mathcal{P}(m)$. Then q splits in K as $\mathfrak{Q}_1^p \dots \mathfrak{Q}_s^p$ for $s \geqslant 1$, and $v_q(D_K) = (p-1)p^{r-1}$.*

Proof. By Lemma 5.2, one can suppose that $\forall i \in [\![2, r]\!], q \mid m_i \iff i = 1$. Let us fix $K_1 = \mathbb{Q}(\sqrt[p]{m_1})$ and $K_2 = \mathbb{Q}(\sqrt[p]{m_2}, \dots, \sqrt[p]{m_r})$. By [100] the prime q ramifies in K_1 as $\mathfrak{q}^p$. Moreover q is unramified in K_2 so $q\mathcal{O}_{K_2} = \mathfrak{q}_1 \cdots \mathfrak{q}_s$ with $s \geqslant 1$. By multiplicativity of the ramification index, for all $i \in [\![1, s]\!]$, the ideal $\mathfrak{q}_i$ ramifies completely in K as $\mathfrak{Q}_i^p$. Therefore $q\mathcal{O}_K = (\mathfrak{Q}_1 \cdots \mathfrak{Q}_s)^p$.

$$\begin{tikzcd}
 & K \quad (\mathfrak{Q}_1\cdots\mathfrak{Q}_s)^p \arrow[ddl, dash] \arrow[dr, dash] & \\
 & & K_2 \quad \mathfrak{q}_1\cdots\mathfrak{q}_s \arrow[ddl, dash] \\
\mathfrak{q}^p \quad K_1 \arrow[dr, dash] & & \\
 & \mathbb{Q} & \\
 & (q) &
\end{tikzcd}$$

Now recall that the different of $K/\mathbb{Q}$ satisfies $\mathfrak{D}(K/\mathbb{Q}) = \prod_{\mathfrak{Q}} \mathfrak{Q}^{s_{\mathfrak{Q}}}$ where the product is over the prime ideals of $\mathcal{O}_K$ which are ramified over $\mathbb{Q}$. Thus the part of $\mathfrak{D}(K/\mathbb{Q})$ above q is $\prod_{i=1}^s \mathfrak{Q}_i^{s_i}$ for some integers s_i. For all $i \in [\![1, s]\!]$ we know that $e(\mathfrak{Q}_i|q) = p$

and q are coprime. Therefore s_i is equal to $e(\mathfrak{Q}_i|q) - 1 = p - 1$. Thus one has for the discriminant

$$v_q(D_K) = v_q(\mathrm{N}_{K/\mathbb{Q}}(\mathfrak{D}(K/\mathbb{Q}))) = v_q(\mathrm{N}_{K_2/\mathbb{Q}}(\mathrm{N}_{K/K_2}(\prod_{i=1}^{s} \mathfrak{q}_i^{p-1}))).$$

Finally since $\mathrm{N}_{K/K_2}(\mathfrak{Q}_i) = \mathfrak{q}_i$ we obtain

$$v_q(D_K) = v_q(\mathrm{N}_{K_2/\mathbb{Q}}(\prod_{i=1}^{s} \mathfrak{q}_i^{p-1})) = v_q(\mathrm{N}_{K_2/\mathbb{Q}}(q\mathcal{O}_{K_2})^2) = (p-1)p^{r-1}.$$

□

With Proposition 5.12 one is able to prove the result we were looking for.

Theorem 5.2. *Let $K = \mathbb{Q}(\sqrt[p]{m_1}, \ldots, \sqrt[p]{m_r})$ be a real Kummer extension, and denote by $\mathcal{O}$ the order $\mathbb{Z}[\mathfrak{IB}(K)]$. Then the following propositions are true:*

- *$\forall q \in \mathcal{P}(m) \setminus \{p\}$, $\mathcal{O}$ is $q-$maximal;*
- *$[K : \mathbb{Q}]\mathcal{O}_K < \mathcal{O}$.*

Proof. Proposition 5.12 and Theorem 5.1 show that $v_q(\mathcal{O}) = v_q(\mathcal{O}_K)$ for all $q \in \mathcal{P}(m) \setminus \{p\}$, so $\mathcal{O}$ is indeed $q-$maximal. Concerning p one has

$$v_p(D_K([K : \mathbb{Q}]\mathcal{O}_K)) \geqslant 2[K : \mathbb{Q}]v_q([K : \mathbb{Q}]) = 2rp^r \geqslant rp^r + p^{r-1}(p-1)$$

so the second property is also true. □

Despite the fact that Theorem 5.2 shows that $\mathfrak{IB}(K)$ is a basis satisfying the properties we were looking for, we can still study further the splitting of p in K in each of the four types of real Kummer fields established in Proposition 5.11. It allows us to have a finer knowledge of D_K. First let us establish a result concerning extensions of number fields such that the Galois group of their Galois closure is dihedral.

Lemma 5.5. *Let L/K be an extension of number fields. Suppose additionally that $\mathrm{Gal}(\tilde{L}/K)$ is isomorphic to $\langle\tau\rangle \ltimes \langle\sigma\rangle$, with $\langle\tau\rangle \cong \frac{\mathbb{Z}}{(p-1)\mathbb{Z}}$ and $\langle\sigma\rangle \cong \frac{\mathbb{Z}}{p\mathbb{Z}}$ for some prime integer p. Any prime ideal $\mathfrak{p}$ of $\mathcal{O}_K$ satisfies*

$$\mathfrak{p}\mathcal{O}_{\tilde{L}} = (\mathfrak{P}_1 \ldots \mathfrak{P}_p)^{p-1} \implies \mathfrak{p}\mathcal{O}_L = \mathfrak{p}_1\mathfrak{p}_2^{p-1},$$

where each $\mathfrak{P}_i$ is a prime ideal of $\tilde{L}$ and each $\mathfrak{p}_i$ is a prime ideal of L.

It is similar to part of the proof for Proposition 10.1.26 of Cohen's book [30]. In fact several facts and their proofs that we will establish are generalisations of this Proposition.

Proof. Let $G = \mathrm{Gal}(\tilde{L}/K)$. By hypothesis we are in the following situation :

$$\begin{tikzcd}
& \tilde{L} & (\mathfrak{P}_1\cdots\mathfrak{P}_p)^{p-1} \\
L \arrow[ur, dash, "p-1", "\tau"'] & & \\
& \tilde{K} \arrow[uu, dash, "\sigma", "p"'] & \\
K \arrow[uu, dash, "p"] \arrow[ur, dash, "p-1"'] & & \\
\mathfrak{p} & &
\end{tikzcd}$$

The group G acts transitively on the $\mathfrak{P}_i$ and by conjugation on the inertia groups $I(\mathfrak{P}_i \mid \mathfrak{p})$ for $i \in [\![1,p]\!]$. Clearly one has $|I(\mathfrak{P}_i/\mathfrak{p})| = p-1$. By Lemma B.3 there are p distinct subgroups of G of order $p-1$. Moreover they are of the form $\langle\tau\sigma^b\rangle$ with $b \in [\![0,p-1]\!]$. Therefore the action of G on the set of such subgroups is transitive. Thus there is a unique $i_0 \in [\![1,p]\!]$ such that $I(\mathfrak{P}_{i_0} \mid \mathfrak{p}) = \langle\tau\rangle$, and $I(\mathfrak{P}_{i_0} \mid \mathfrak{P}_{i_0} \cap \mathcal{O}_L) = I(\mathfrak{P}_{i_0} \mid \mathfrak{p}) \cap \mathrm{Gal}(\tilde{L}/L) = \langle\tau\rangle$. Therefore $e(\mathfrak{P}_{i_0} \mid \mathfrak{P}_{i_0} \cap \mathcal{O}_L) = p-1$ so by multiplicativity $e(\mathfrak{P}_{i_0} \cap \mathcal{O}_L \mid \mathfrak{p}) = 1$. Now consider $i \neq i_0$. Then $I(\mathfrak{P}_i \mid \mathfrak{p}) = \langle\tau\sigma^b\rangle$ for some $b \in [\![1,p-1]\!]$, and $I(\mathfrak{P}_i \mid \mathfrak{P}_{i_0} \cap \mathcal{O}_L) = I(\mathfrak{P}_{i_0} \mid \mathfrak{p}) \cap \mathrm{Gal}(\tilde{L}/L) = \langle 1\rangle$. Therefore again by multiplicativity of the ramification index, $e(\mathfrak{P}_i \cap \mathcal{O}_L \mid \mathfrak{p}) = p-1$. □

Theorem 5.3. *Consider $K = \mathbb{Q}(\sqrt[p]{m_1}, \dots, \sqrt[p]{m_r})$ a real Kummer extension with exponent p. Then depending on the type of field as described in Proposition 5.11 the splitting of p in K and $v_p(D_K)$ are as follows :*

1. *$(p) = \mathfrak{p}(\mathfrak{p}_1 \dots \mathfrak{p}_s)^{p-1}$ for $s = \frac{p^r-1}{p-1}$, and $v_p(D_K) = \frac{p^r-1}{p-1}(p-2)$;*
2. *$(p) = \mathfrak{p}^p(\mathfrak{p}_1 \dots \mathfrak{p}_s)^{p(p-1)}$ for $s = \frac{p^{r-1}-1}{p-1}$, and $v_p(D_K) = p^r + \frac{p^{r-1}-1}{p-1}(p-2)$;*
3. *$(p) = \mathfrak{p}^p(\mathfrak{p}_1 \dots \mathfrak{p}_s)^{p(p-1)}$ for $s = \frac{p^{r-1}-1}{p-1}$, and $v_p(D_K) = p^{r-1}(2p-1) + \frac{p^{r-1}-1}{p-1}(p-2)$.*

Remark 33. We were not able to prove similar results for the fourth type of field for a general exponent p. However we did so for $p = 3$ in [64].

Proof. We will prove the results one type of fields after another.

Fields of the first type. We will prove the factorisation by induction. For $r = 1$ K is a simple Kummer extension. Then the splitting is correct, following Westlund [100]. Now consider $r \geqslant 1$ and assume the result is true for r. Let $K = \mathbb{Q}(\sqrt[p]{m_1}, \dots, \sqrt[p]{m_{r+1}})$ a real Kummer field such that for all $i \in [\![1, r+1]\!], m_i^{p-1} \equiv 1 \bmod p^2$. Let us fix $K_1 = \mathbb{Q}(\sqrt[p]{m_1}, \dots, \sqrt[p]{m_r})$ and $K_2 = \mathbb{Q}(\sqrt[p]{m_{r+1}})$. If one denotes $\frac{p^r-1}{p-1}$ by s, one has the following decompositions by using the induction hypothesis, where the numbers are the dimensions of the respective extensions.

$$\begin{tikzcd}
 & & & \widetilde{K} \arrow[dll, dash, "p"'] \arrow[d, dash, "p"'] \arrow[ddrrr, dash, "p^r"] & & & \\
 & \widetilde{K_1} \arrow[ddr, dash, "p-1"] & & K \arrow[ddl, dash, "p"] \arrow[ddrr, dash, "p^r"] & & & \\
 & & & & & & \widetilde{K_2} \arrow[dl, dash, "p-1"'] \\
\mathfrak{p}\mathfrak{p}_1^{p-1}\cdots\mathfrak{p}_s^{p-1} & & K_1 \arrow[drr, dash, "p^r"'] & & & K_2 \arrow[dl, dash, "p"] & \mathfrak{q}_1\mathfrak{q}_2^{p-1} \\
 & & & & \mathbb{Q} & & \\
 & & & & (p) & &
\end{tikzcd}$$

Moreover p is totally ramified as $\mathfrak{a}^{p-1}$ in $k = \mathbb{Q}(\zeta_p)$. First we consider the splitting of p in the Galois closure $\widetilde{K_1}$ and $\widetilde{K_2}$. We focus on $\widetilde{K_2}$, and the situation in $\widetilde{K_1}$ is similar. First remark that $\widetilde{K_2}/\mathbb{Q}$ is Galois with dimension $[\widetilde{K_2} : \mathbb{Q}] = p(p-1)$ so the decomposition of p satisfies $efg = p(p-1)$ with the functions $e(\cdot|p)$ and $f(\cdot|p)$ being constant – equal to e and f respectively – over prime ideals $\widetilde{\mathfrak{q}}$ of $\widetilde{K_2}$ such that $\widetilde{\mathfrak{q}} \mid (p)$. Considering the factorisation $p\mathcal{O}_{K_2} = \mathfrak{q}_1\mathfrak{q}_2^{p-1}$ we obtain $p-1 \mid e(\widetilde{\mathfrak{q}}|p)$. Moreover for the decomposition of $\mathfrak{q}_\mathfrak{i}$ in $\widetilde{K_2}$, since $\widetilde{K_2}/K_2$ is Galois, we have $e_i f_i g_i = p-1$. Since $p-1 \mid e$ and $e = e_1$ we have $e_1 = p-1$, $f_1 = 1$ and $g_1 = 1$. Therefore $\mathfrak{q}_1\mathcal{O}_{\widetilde{K_2}} = \widetilde{\mathfrak{q}}^{p-1}$. Moreover $f = f_1 = f_2$ and $e = (p-1)e_2$ so $e_2 = 1$ and $g_2 = p-1$. Thus $\mathfrak{q}_2$ splits completely in $\widetilde{K_2}$ as $\widetilde{\mathfrak{q}_1}\widetilde{\mathfrak{q}_2}\dots\widetilde{\mathfrak{q}_{p-1}}$. Finally we obtain the factorisation $(p) = \widetilde{\mathfrak{q}}^{p-1}\widetilde{\mathfrak{q}_1}^{p-1}\widetilde{\mathfrak{q}_2}^{p-1}\dots\widetilde{\mathfrak{q}_{p-1}}^{p-1}$ in $\widetilde{K_2}$. Similarly we have $\mathfrak{p}\mathcal{O}_{\widetilde{K_1}} = \widetilde{\mathfrak{p}}^{p-1}$ and $\mathfrak{p}_i$ splits completely in $\widetilde{K_1}$ for all $i \in [\![1, s]\!]$. Therefore the factorisations of (p) in $\widetilde{K_1}$ and $\widetilde{K_2}$ are as follows:

$$(p) = \begin{cases} \widetilde{\mathfrak{q}}^{p-1}(\widetilde{\mathfrak{q}_1}\widetilde{\mathfrak{q}_2}\dots\widetilde{\mathfrak{q}_{p-1}})^{p-1}, & \text{in } \widetilde{K_2}/\mathbb{Q}, \\ \widetilde{\mathfrak{p}}^{p-1}(\widetilde{\mathfrak{p}_1}\widetilde{\mathfrak{p}_2}\dots\widetilde{\mathfrak{p}_s})^{p-1}, & \text{in } \widetilde{K_1}/\mathbb{Q}. \end{cases}$$

Consequently the splitting of $\mathfrak{a}$ in the same two fields is

$$(\mathfrak{a}) = \begin{cases} \widetilde{\mathfrak{q}}\widetilde{\mathfrak{q}_1}\widetilde{\mathfrak{q}_2}\dots\widetilde{\mathfrak{q}_{p-1}}, & \text{in } \widetilde{K_2}/\mathbb{Q}, \\ \widetilde{\mathfrak{p}}\widetilde{\mathfrak{p}_1}\widetilde{\mathfrak{p}_2}\dots\widetilde{\mathfrak{p}_s}, & \text{in } \widetilde{K_1}/\mathbb{Q}. \end{cases}$$

Remark that the residual degree is 1 everywhere. We will now consider the decomposition of p in $\tilde{K}$. We will in fact look at the decomposition of $\mathfrak{a}$. Consider $\tilde{\mathfrak{P}}$ a prime ideal of $\widetilde{K}$ above p. Remark it is also above $\mathfrak{a}$ in $\widetilde{K}/k$. For $i \in \{1,2\}$, denote by D_i the decomposition group $D(\tilde{\mathfrak{P}} \mid \tilde{\mathfrak{P}} \cap \mathcal{O}_{\tilde{K}_i})$. Let us write $G = \mathrm{Gal}(\widetilde{K}/k)$, $G_1 = \mathrm{Gal}(\widetilde{K}/\widetilde{K}_1)$ and $G_2 = \mathrm{Gal}(\widetilde{K}/\widetilde{K}_2)$. Each D_i is a subgroup of $G_i < G$. Moreover recall that $G_1 \cong \langle \sigma_{r+1} \rangle$, $G_2 \cong \langle \sigma_1 \rangle \times \cdots \times \langle \sigma_r \rangle$ and $G \cong G_1 \times G_2$. Remark also that $|D_1| = |D_2|$. Since $|G_1| = p$ then one has $D_1 = \langle 1 \rangle$ or $D_1 = G_1$. Let us show that $D_1 = \langle 1 \rangle$. Suppose that we have $D_1 = \langle \sigma_{r+1} \rangle$. Then $|D_2| = p$ so there is $\sigma \in G_2$ such that $o(\sigma) = p$ and $D_2 = \langle \sigma \rangle$. Now, since for $i \in \{1,2\}$ we have $D_i = D(\tilde{\mathfrak{P}} \mid \mathfrak{a}) \cap G$, we obtain $\{\sigma_{r+1}\} < D(\tilde{\mathfrak{P}} \mid \mathfrak{a})$ and $\{\sigma\} < D(\tilde{\mathfrak{P}} \mid \mathfrak{a})$. Therefore, $\{\sigma_{r+1}\} \times \{\sigma\} < D(\tilde{\mathfrak{P}} \mid \mathfrak{a})$ which implies that $ef = |D(\tilde{\mathfrak{P}} \mid \mathfrak{a})| \geqslant p^2$. However if we consider the splitting of $\mathfrak{a}$ in $\widetilde{K}_1$ and $\widetilde{K}$, we have $e_1 = f_1 = 1$ in $\widetilde{K}_1/k$ and $[\widetilde{K} : \widetilde{K}_1] = p$, so $ef \leqslant p$ in $\widetilde{K}/k$. Thus we have an absurdity so D_1 is trivial as announced, $D_1 = D_2 = \langle 1 \rangle$ and $\mathfrak{a}$ splits completely in $\widetilde{K}/k$. Finally $\mathfrak{p}$ splits in $\widetilde{K}/K_1$ as

$$(\widetilde{\mathfrak{P}}\widetilde{\mathfrak{P}} \ldots \widetilde{\mathfrak{P}}_p)^{p-1},$$

and

$$\mathrm{Gal}(\widetilde{K}/K_1) \cong \langle \tau_p \rangle \ltimes \langle \sigma_{r+1} \rangle \cong \frac{\mathbb{Z}}{(p-1)\mathbb{Z}} \ltimes \frac{\mathbb{Z}}{p\mathbb{Z}}.$$

We see that $\mathfrak{p}$ and $\tilde{K}/K_1$ satisfy the hypothesis of Lemma 5.5, so $\mathfrak{p}$ splits in K/K_1 as

$$\widetilde{\mathfrak{P}}\widetilde{\mathfrak{P}}_1^{p-1}.$$

Moreover, for each $i \in [\![1, s]\!]$, the ideal $\mathfrak{p}_i$ splits completely in $\widetilde{K}$ so it splits completely in K. We obtain the final decomposition for p in $K/\mathbb{Q}$ as

$$(p) = \mathfrak{P}(\mathfrak{P}_1 \ldots \mathfrak{P}_t)^{p-1}$$

with $t = 1 + sp = 1 + \frac{p^r-1}{p-1}p = \frac{p^{r+1}-1}{p-1}$. Thus the decomposition is correct for $r+1$, which ends the proof by induction. Let us now fix $K = \mathbb{Q}(\sqrt[p]{m_1}, \ldots, \sqrt[p]{m_r})$ and look at the p-valuation of D_K. Remark that $\gcd(1,p) = \gcd(p-1,p) = 1$, and that for any prime ideal $\mathfrak{Q}$ of K above p we have $e(\mathfrak{Q} \mid p) = 1$ or $e(\mathfrak{Q} \mid p) = p-1$. Therefore the part of $\mathfrak{D}(K/\mathbb{Q})$ above p is

$$\prod_{i=1}^{s} \mathfrak{P}_i^{p-2}$$

where $s = \frac{p^r-1}{p-1}$. Since the inertial degree $f(\mathfrak{P}_i \mid p) = 1$, we have $\mathrm{N}_{K/\mathbb{Q}}(\mathfrak{P}_i) = p$ for all $i \in [\![1, s]\!]$. Thus we obtain

$$v_p(D_K) = v_p\left(\mathrm{N}_{K/\mathbb{Q}}(\mathfrak{D}(K/\mathbb{Q}))\right) = v_p\left(\prod_{i=1}^{s} \mathrm{N}_{K/\mathbb{Q}}(\mathfrak{P}_i)^{p-2}\right) = (p-2)s,$$

which is the required value.

Fields of the second type: Let us now consider a field $K = \mathbb{Q}(\sqrt[p]{m_1}, \dots, \sqrt[p]{m_r})$ such that $p \notin \mathcal{P}(m)$, $m_1^{p-1} \not\equiv 1 \bmod p^2$ and $\forall i \in [\![2, r]\!], m_i^{p-1} \equiv 1 \bmod p^2$. The proof is simpler in this case. Fix $K_1 = \mathbb{Q}(\sqrt[p]{m_1})$ and $K_2 = \mathbb{Q}(\sqrt[p]{m_2}, \dots, \sqrt[p]{m_r})$. Remark that K_2 is a real Kummer field of the first type. Therefore, following Westlund [100] and the previous result, for $s = \frac{p^{r-1}-1}{p-1}$ we obtain the following situation.

$$\begin{tikzcd}
 & & K \arrow[ddl, dash, "p^{r-1}"'] \arrow[dr, dash, "p"] & & \\
 & & & K_2 \arrow[ddl, dash, "p^{r-1}"] & \mathfrak{q}\mathfrak{q}_1^{p-1}\cdots\mathfrak{q}_s^{p-1} \\
\mathfrak{p}^p & K_1 \arrow[dr, dash, "p"'] & & & \\
 & & \mathbb{Q} & & \\
 & & (p) & &
\end{tikzcd}$$

By multiplicativity of the ramification index, for any $\mathfrak{P}$ above p in K, one has $p \mid e(\mathfrak{P} \mid p)$. Thus the splitting of p in K is as follows:

$$(p) = \mathfrak{P}^p(\mathfrak{P}_1 \dots \mathfrak{P}_s)^{p(p-1)}.$$

Now let us find $v_p(D_K)$. We have

$$D_K = D_{K_1}^{[K:K_1]}\mathrm{N}_{K_1/\mathbb{Q}}(\mathfrak{d}(K/K_1)) = D_{K_1}^{[K:K_1]}\mathrm{N}_{K_1/\mathbb{Q}}(\mathrm{N}_{K/K_1}(\mathfrak{D}(K/K_1)))$$

and the part of $\mathfrak{D}(K/K_1)$ over $\mathfrak{p}$ is $(\mathfrak{P}_1 \cdots \mathfrak{P}_s)^{p-2}$. Indeed p is coprime to 1 and $p-1$. We know by [100] that $v_p(D_{K_1}) = p$ so

$$v_p(D_K) = [K : K_1]p + v_p\left(\mathrm{N}_{K/\mathbb{Q}}((\mathfrak{P}_1 \cdots \mathfrak{P}_s))^{p-2}\right).$$

Since the inertial degree is trivial everywhere, $\mathrm{N}_{K/\mathbb{Q}}(\mathfrak{P}_i) = p$ for all $i \in [\![1, s]\!]$. Finally we obtain

$$v_p(D_K) = p^r + v_p(p^{s(p-2)}) = p^r + s(p-2).$$

Fields of the third type: Let us now consider a field $K = \mathbb{Q}(\sqrt[p]{m_1}, \dots, \sqrt[p]{m_r})$ such that $p \in \mathcal{P}$, and $\forall i \in [\![2, r]\!], m_i^{p-1} \equiv 1 \bmod p^2$. Again fix $K_1 = \mathbb{Q}(\sqrt[p]{m_1})$ and $K_2 = \mathbb{Q}(\sqrt[p]{m_2}, \dots, \sqrt[p]{m_r})$. Remark that K_2 is a real Kummer field of the first type. Therefore, following Westlund [100] and the previous result, for $s = \frac{p^{r-1}-1}{p-1}$ we obtain the decomposition as the previous case. Therefore the proof is identical. The only thing which changes is $v_p(D_{K_1})$. It is equal to $2p-1$ in this case [100]. □

Extensions with two exponents

We were not able to prove similar results for general Kummer extensions with two exponents, but only on a restricted family of them.

Definition 5.7. Let L/K be a real Kummer extension with two exponents p, q. We will call a *power-free basis of* L/K and denote by $\mathfrak{IB}(L/K)$ the basis $\mathfrak{IB}(M) \otimes \mathfrak{IB}(K)$.

Proposition 5.13. *Let* $L = K(\sqrt[p]{m_1}, \dots, \sqrt[p]{m_r})$ *with* $K = \mathbb{Q}(\sqrt[q]{n_1}, \dots, \sqrt[q]{n_s})$ *be a real Kummer extension with two exponents. Let* $a \in \mathcal{P}(m) \cup \mathcal{P}(n)$. *Write* $\delta_m = \delta_{(a \in \mathcal{P}(m))}$ *and* $\delta_n = \delta_{(a \in \mathcal{P}(n))}$. *If* $a \notin \{p, q\}$, *then one has*

$$v_a(D_L(\mathfrak{IB}(L/K))) = [L : \mathbb{Q}] \left(\frac{p-1}{p}\delta_m + \frac{q-1}{q}\delta_n \right).$$

If $a \in \{p, q\}$ *then one has*

$$v_a(D_L(\mathfrak{IB}(L/K))) = \begin{cases} [L : \mathbb{Q}] \left(r + \frac{p-1}{p}\delta_m + \frac{q-1}{q}\delta_n \right) & \text{if } a = p, \\ [L : \mathbb{Q}] \left(s + \frac{p-1}{p}\delta_m + \frac{q-1}{q}\delta_n \right) & \text{if } a = q. \end{cases}$$

Proof. With the notations used during the proof of Theorem 5.1, remark that $M_{\mathfrak{IB}(L/K)} = M_{\mathfrak{IB}(L')} \otimes M_{\mathfrak{IB}(K)}$ where $L' = \mathbb{Q}(\sqrt[p]{m_1}, \dots, \sqrt[p]{m_r})$. Then apply v_a to $\det M^2_{\mathfrak{IB}(L/K)}$ in the different cases. □

Remember that to prove Theorem 5.2, one only has to study the splitting of primes different from the exponent p, as the p-valuation of the discriminant of the order generated by $\mathfrak{IB}(K)$ is automatically smaller than the one of the discriminant of $[K : \mathbb{Q}]\mathcal{O}_K$. We will see that it is not as simple over extensions with two exponents.

Proposition 5.14. *Let* $L = K(\sqrt[p]{m_1}, \dots, \sqrt[p]{m_r})$ *with* $K = \mathbb{Q}(\sqrt[q]{n_1}, \dots, \sqrt[q]{n_s})$ *be a real Kummer extension with two exponents. Let* $a \in \mathcal{P}(m) \cup \mathcal{P}(n) \setminus \{p, q\}$. *Then the splitting of* a *in* $L/\mathbb{Q}$ *and* $v_a(D_L)$ *satisfy the following:*

1. $a \in \mathcal{P}(m) \setminus \mathcal{P}(n) \implies \exists t \geqslant 1, (a) = (\mathfrak{a}_1 \dots \mathfrak{a}_t)^p$ *and* $v_a(D_L) = [L : \mathbb{Q}]\frac{p-1}{p}$;
2. $a \in \mathcal{P}(n) \setminus \mathcal{P}(m) \implies \exists t \geqslant 1, (a) = (\mathfrak{a}_1 \dots \mathfrak{a}_t)^p$ *and* $v_a(D_L) = [L : \mathbb{Q}]\frac{q-1}{q}$;
3. $a \in \mathcal{P}(m) \cap \mathcal{P}(n) \implies \exists t \geqslant 1, (a) = (\mathfrak{a}_1 \dots \mathfrak{a}_t)^{pq}$ *and* $v_a(D_L) = [L : \mathbb{Q}]\frac{pq-1}{pq}$.

Proof. The proof is quite similar to the one of Proposition 5.12. Using Lemma 5.2, one can assume that there is at most one $i_0 \in [\![1, r]\!]$ such that $a \mid m_{i_0}$ and at most one $j_0 \in [\![1, s]\!]$ such that $a \mid n_{j_0}$. Assume also that i_0 and j_0 are equal to 1 when they exist. Fix l the field equal to the compositum of the simple subfields of L' and K

generated by m_{i_0} and n_{i_0}. Depending on the cases, l is equal to $\mathbb{Q}(\sqrt[p]{m_1})$, $\mathbb{Q}(\sqrt[q]{n_1})$ or $\mathbb{Q}(\sqrt[p]{m_1})\mathbb{Q}(\sqrt[q]{n_1})$. Now let k be the field such that $lk = L$. Now it is easy to see that a completely ramifies in l and does not ramify in k. Thus there is $t \geqslant 1$ such that the splitting of a is as follows.

$$\begin{tikzcd}
& L \quad (\mathfrak{a}_1 \cdots \mathfrak{a}_t)^{[l:\mathbb{Q}]} \arrow[dl, dash] \arrow[dr, dash] & \\
\mathfrak{p}^{[l:\mathbb{Q}]} \quad l \arrow[dr, dash] & & k \quad \mathfrak{p}_1 \cdots \mathfrak{p}_t \arrow[dl, dash] \\
& \mathbb{Q} &
\end{tikzcd}$$

(a)

Since $a \notin \{p, q\}$, $\gcd(a, [l : \mathbb{Q}]) = 1$, therefore the part of the different $\mathfrak{D}(L/\mathbb{Q})$ above a is equal to

$$\prod_{i=1}^{t} \mathfrak{a}^{[l:\mathbb{Q}]-1}.$$

One can conclude by using the same arguments than in the proof of Proposition 5.12.

□

Remark 34. One can remark from Proposition 5.13 and Proposition 5.14 that if $a \in \mathcal{P}(m) \cap \mathcal{P}(n) \setminus \{p, q\}$ then $v_a(D_K) \geqslant v_a(\mathbb{Z}[\mathfrak{IB}(L)])$. Therefore if $\mathcal{P}(m) \cap \mathcal{P}(n) \setminus \{p, q\} \neq \emptyset$ then the counterpart of Theorem 5.2 for Kummer extension with two exponents does not hold.

Theorem 5.4. *Let $L = K(\sqrt[p]{m_1}, \ldots, \sqrt[p]{m_r})$ with $K = \mathbb{Q}(\sqrt[q]{n_1}, \ldots, \sqrt[q]{n_s})$ be a real Kummer extension with two exponents. Denote by $\mathcal{O}$ the order $\mathbb{Z}[\mathfrak{IB}(L)]$, and $A = (\mathcal{P}(m) \cap \mathcal{P}(n)) \setminus \{p, q\}$ and $P_A = \prod_{a \in A} a$. Then the following properties are true.*

- *$\forall a \in \mathcal{P}(m) \cup \mathcal{P}(n) \setminus (A \cup \{p, q\})$, $\mathcal{O}$ is $a-$maximal.*
- *$P_A[L : \mathbb{Q}]\mathcal{O}_L < \mathcal{O}$.*

Proof. Let $a \in \mathcal{P}(m) \cup \mathcal{P}(n) \setminus (A \cup \{p, q\})$. From Proposition 5.13 and Proposition 5.14, $v_a(D_L(\mathcal{O})) = v_a(D_L(\mathcal{O}_L))$ so $\mathcal{O}$ is indeed $a-$maximal. Consider $a \in A$. Then we have $v_a(D_L(\mathcal{O})) = [L : \mathbb{Q}]\left(\frac{p-1}{p} + \frac{q-1}{q}\right)$, and

$$v_a\left(D_L(P_A[L : \mathbb{Q}]\mathcal{O}_L)\right) = v_a(P_A^{2[L:\mathbb{Q}]} D_L) = 2[L : \mathbb{Q}] + v_a(D_L).$$

Since $v_a(D_L) = [L : \mathbb{Q}]\frac{pq-1}{pq}$, we obtain

$$v_a\left(D_L(P_A[L : \mathbb{Q}]\mathcal{O}_L)\right) = [L : \mathbb{Q}](\frac{2pq + pq - 1}{pq}) \geqslant v_a(D_L(\mathcal{O})).$$

Now consider $a \in \{p, q\}$. Since the situation is the same for p or q, we can choose $a = p$ for example. First assume that $p \notin \mathcal{P}(m) \cup \mathcal{P}(n)$. Then again from Proposition 5.13 we have $v_p(D_L(\mathcal{O})) \leqslant r[L : \mathbb{Q}]$. Moreover since $v_p([L : \mathbb{Q}]) = r$ we get

$$v_p\left(D_L(P_A[L : \mathbb{Q}]\mathcal{O}_L)\right) \geqslant 2r[L : Q] \geqslant r[L : \mathbb{Q}].$$

Now let us assume that $p \in \mathcal{P}(m) \cup \mathcal{P}(n)$. Then we have

$$v_p(D_L(\mathcal{O})) = [L : \mathbb{Q}]\left(r + \frac{p-1}{p} + \frac{q-1}{q}\right) \leqslant [L : \mathbb{Q}](r + 2).$$

Since $p \in \mathcal{P}(m) \cup \mathcal{P}(n)$, there is a subfield l of L of the form $\mathbb{Q}(\sqrt[p]{\prod_i m_i})$ (resp. $\mathbb{Q}(\sqrt[q]{\prod_i n_i})$), such that $p \mid m$ (resp. $p \mid n$). Consequently, p ramifies completely in l and we know that $v_p(D_l) = 2p - 1$ (resp. $v_p(D_l) = p$). Recall that $D_L = D_l^{[L:l]}\mathrm{N}_{l/\mathbb{Q}}(\mathfrak{d}(L/l)) \geqslant D_l^{[L:l]}$. Thus we obtain

$$v_p\left(D_L(P_A[L : \mathbb{Q}]\mathcal{O}_L)\right) \geqslant 2r[L : Q] + [L : \mathbb{Q}] \geqslant [L : \mathbb{Q}](r + 2).$$

□

5.2.3 Geometry under Log_L

Lemma 5.6. *Consider K_1 and K_2 two number fields, and $K = K_1K_2$ their compositum. Assume that $\mathrm{Hom}(K, \mathbb{C}) \cong \mathrm{Hom}(K_1, \mathbb{C}) \otimes \mathrm{Hom}(K_2, \mathbb{C})$. Then one has*

$$\forall (x_1, x_2) \in K_1 \times K_2, (\mathrm{Log}_K(x_1) \mid \mathrm{Log}_K(x_2)) = \ln|\mathrm{N}_{K_1/\mathbb{Q}}(x_1)| \ln|\mathrm{N}_{K_2/\mathbb{Q}}(x_2)|.$$

In particular $\mathrm{Log}_K(\mathcal{O}_{K_1}^\times)$ is orthogonal to $\mathrm{Log}_K(x_2)$ for any $x_2 \in K_2$.

Proof. Let us denote by H, H_1 and H_2 the sets $\mathrm{Hom}(K, \mathbb{C})$, $\mathrm{Hom}(K_1, \mathbb{C})$ and $\mathrm{Hom}(K_2, \mathbb{C})$ respectively. Moreover we will write S for $(\mathrm{Log}_K(x_1) \mid \mathrm{Log}_K(x_2))$. Then we have

$$S = \sum_{\sigma \in H} \ln|\sigma(x_1)| \ln|\sigma(x_2)| = \sum_{\sigma_1 \in H_1} \sum_{\sigma_2 \in H_2} \ln|\sigma_1 \otimes \sigma_2(x_1)| \ln|\sigma_1 \otimes \sigma_2(x_2)|.$$

Then for $i \in \{1, 2\}$ we get $\sigma_1 \otimes \sigma_2(x_i) = \sigma_i(x_i)$. Thus we obtain

$$S = \sum_{\sigma_1 \in H_1} \sum_{\sigma_2 \in H_2} \ln|\sigma_1(x_1)| \ln|\sigma_2(x_2)| = \sum_{\sigma_1 \in H_1} \ln|\sigma_1(x_1)| \sum_{\sigma_2 \in H_2} \ln|\sigma_2(x_2)|$$

which gives the first result. The statement about the orthogonality of the units follows from the fact that their algebraic norm is ± 1. □

Corollary 5.2. *Let $K = \mathbb{Q}(\sqrt[p]{m_1}, \ldots, \sqrt[p]{m_r})$ be a real Kummer field with one exponent. Then we have*

$$\mathrm{Log}_K(\mathrm{SU}(K)) = \overset{\perp}{\bigoplus_{\alpha \in \frac{\mathbb{F}_p^r \setminus \{0\}}{\sim}}} \mathrm{Log}_K(\mathcal{O}_{K_\alpha}^\times) \tag{5.6}$$

Proof. Just remark that for any pair $(\alpha, \beta) \in \mathbb{F}_p^r \setminus \{0\}$ such that $\alpha \not\sim \beta$ we can apply Lemma 5.6 to K_α and K_β. □

We know that $\mathrm{SU}(K)$ is a full-rank subgroup of $\mathcal{O}_K^\times$ following Corollary 5.1, and equivalently $\mathrm{Log}_K(\mathrm{SU}(K))$ is a full-rank sublattice of $\mathrm{Log}_K(\mathcal{O}_K^\times)$. In the case of multiquadratic and multicubic fields, one can see from Corollary 5.2 that each set of fundamental units $\{\epsilon_\alpha \mid \alpha \in \frac{\mathbb{F}_p^r \setminus \{0\}}{\sim}\}$ is sent by Log_K to an orthogonal basis of this sublattice. This is the best situation possible when it comes to solving lattice problems. In particular one could hope to decode respectively to $\mathrm{Log}_K(\mathrm{SU}(K))$, and use enumerations like over cyclotomic fields in [34]. However as mentioned in [6] the index $[\mathcal{O}_K^\times : \mathrm{SU}(K)]$ is too large for this strategy to be efficient. On the other hand, Algorithm 30 shows that one can obtain $\mathrm{Log}_K(\mathcal{O}_K^\times)$ from $\mathrm{Log}_K(\mathrm{SU}(K))$ by doing simple operations on vectors: additions and division by a scalar (2 or 3 depending on the case).

For Kummer extensions with one exponent $p > 3$, we obtain blocks of size $\frac{p-1}{2}$ orthogonal one to each other, i.e. if we consider a basis matrix M of $\mathrm{Log}_K(\mathrm{SU}(K))$ then its Gram matrix $MM^\intercal$ is a block diagonal matrix

$$\begin{bmatrix} G_\alpha & 0 & \ldots & 0 \\ 0 & G_\beta & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ 0 & \ldots & 0 & G_\gamma \end{bmatrix}$$

with the diagonal blocks being of the form $M_\alpha M_\alpha^\intercal$, with $M_\alpha = \mathrm{Log}_K(\mathcal{O}_{K_\alpha}^\times)$. The basis from which we construct the unit group is therefore not orthogonal anymore. One can wonder whether it has an impact on the quality of the basis obtained for $\mathrm{Log}_K(\mathcal{O}_K^\times)$ and on the performance of the SPIP procedure.

For Kummer extensions with two exponents, we cannot apply Lemma 5.6 to the minimal subfields reached by the recursion of the version of Algorithm 30 adapted to these type of extensions, i.e. Algorithm 35. Indeed, we will see that they are of the form $\mathbb{Q}(\sqrt[p]{M_\alpha}\sqrt[q]{N_\beta})$, which do not satisfy the required properties of Lemma 5.6. The reunion of their unit groups will still generate a full-rank sublattice, but not as a direct sum anymore. Thus we obtain a situation more entangled than with real Kummer extensions with one exponent. Again one may ask how it impacts the possibility of recovering a short generator through the Log-unit lattice.

5.2.4 Auxiliary algorithms

First we will describe the procedures used in Algorithm 30 when applied to real Kummer extensions, as well as how we compute the final reduction step to solve the SPIP. In the following we will denote by N the absolute dimension of L. As in [6, 64] we will always assume that an element x is represented together with an approximation of $\mathrm{Log}_L(x)$, that we will denote by $\mathrm{ApproxLog}_L(x)$. Moreover, we used the *power-free basis* defined and studied in Subsection 5.2.2 to represent elements x. This way we know that there is a coefficient d_L such that the coefficients of $d_L x$ are integers.

Finding Good Primes

As in [6] we will need to be able to find primes satisfying fixed conditions with respect to the m_i's.

Definition 5.8. Consider $m = (m_1, \dots, m_r)$, $C = (c_1, \dots, c_r) \in \{0,1\}^n$ and a prime number p. A *good prime relative to* (m, C, p) is a prime Q such that:

$$\forall i \in [\![1, r]\!], \exists a_i \mid m_i \equiv a_i^p \bmod Q \iff c_i = 1.$$

In particular we need to find good primes Q for the condition sequence $(1, \dots, 1)$ in order to construct morphisms from K^* into finite fields $\mathbb{F}_Q$. Remark that the primes should not divide any of the integers m_i. Now if we fix a prime $Q > 3$ we have the following situation:

- if $Q \equiv 1 \bmod p$ then $\mathbb{F}_Q$ contains a primitive p-th root of unity and $\frac{\mathbb{F}_Q^*}{(\mathbb{F}_Q^*)^p} \simeq \mathbb{F}_p$;
- if $Q \not\equiv 1 \bmod p$ then $\mathbb{F}_Q$ does not contain a primitive p-th root of unity and $\frac{\mathbb{F}_Q^*}{(\mathbb{F}_Q^*)^p} \simeq \{1\}$.

Therefore we can have different strategies depending on our goal. If we want the condition $(1, \dots, 1)$ to be satisfied we might consider primes which are not congruent to 1 modulo p as long as we do not need a non-trivial p-th root of 1 to be in the field $\mathbb{F}_Q$.

Let us now describe how the algorithm operates to find a good prime $Q \equiv 1 \bmod p$. First we have to draw a prime Q and verify that it is congruent to 1 modulo p. This happens with probability $\frac{1}{p-1}$. Then we have to check whether the sequence of conditions C is satisfied by $(m_1, \dots, m_r)$ and Q. We know that $m_i^{\frac{Q-1}{p}} \bmod Q$ has order 1 or p which is equivalent to m_i being a power or not. We have therefore Algorithm 32 where we make use of two functions: `CheckPowerCondition` which has been explained, and `DrawPrime` which corresponds to the way we select the candidates for the prime numbers. One can follow [6] and generate a random prime number in a range given as argument. We could also generate a random prime first and then draw the next prime each time we need a new one.

Algorithm 32 Finding a good prime for a sequence $\underline{d}$ and a condition sequence C - `OneGoodPrime`

Require: A reduced sequence $(m_1, \dots, m_r)$, $C = (c_1, \dots, c_r) \in \{0, 1\}^r$ and a prime p

Ensure: A good prime Q relative to (m, C, p) which does not divide any of the m_i.

```
1: $b \leftarrow 0$
2: while $b = 0$ do
3:     $Q \leftarrow$ DrawPrime
4:     while $Q \not\equiv 1 \bmod p$ do
5:         $Q \leftarrow$ DrawPrime
6:     end while
7:     $b \leftarrow \prod_{i=1}^{r}$ CheckPowerCondition$(m_i, c_i, Q, p)$
8: end while
9: return $Q$
```

For a random prime $Q \equiv 1 \bmod p$ the probability that the power condition is true is equal to $\frac{p-1}{p}$ if $c_i = 0$ and $\frac{1}{p}$ if $c_i = 1$. Therefore if $\mathrm{Hw}(C)$ designates the Hamming weight of C we have

$$\mathbb{P}\left(\prod_{i=1}^{r} \texttt{CheckPowerCondition}(m_i, c_1, Q, p) = 1\right) = (\frac{1}{p})^{\mathrm{Hw}(C)} \times (\frac{p-1}{p})^{r-\mathrm{Hw}(C)}.$$

On average the algorithm will try $\frac{p^r}{(p-1)^{r-\mathrm{Hw(C)}}}$ primes before finding one satisfying the condition sequence C. In particular the probability that each m_i is equal to a p-th power in $\mathbb{F}_Q$ is $\frac{1}{p^r}$ and the algorithm will try $O(p^r)$ primes before finding one satisfying the condition sequence $C = (1, \dots, 1)$. Moreover we check if a m_i is a power or not modulo Q by doing a modular exponentiation. Therefore if Q is polynomial in N as it is expected, the complexity of `CheckCubeCondition` will be polynomial in $\log(N)$.

If we need to find good primes for a given sequence C – as it will be the case to detect non trivial cubes of units – we repeat Algorithm 32 until obtaining enough primes. The only thing to be careful with is the function `DrawPrime` in the case we generate random primes in a given range. It needs to be large enough so that the time taken before generating the desired number of "good" primes is low enough. If `DrawPrime` generates primes by finding the next one then we repeat this process.

Complexity : We obtain a complexity in $\tilde{O}(N)$.

Detecting powers

As mentioned earlier the authors of [6] showed how to realise the characters in the case of multiquadratic fields. It can be adapted to real Kummer extensions, as we did to multicubic fields in [64]. Consider L/K a Kummer extension of exponents p, q and $S = \langle s_1, \dots, s_n \rangle$ a subgroup of L^*. In order to obtain a non trivial character $\chi_Q : S \to \mathbb{F}_p$ one can do as follows. First select a prime Q such that one can construct a ring morphism from $\mathbb{Z}[\sqrt[p]{m_1}, \dots, \sqrt[p]{m_r}, \sqrt[q]{n_1}, \dots, \sqrt[q]{n_s}]$ to $\mathbb{F}_Q$. The prime Q must be such that for all $i \in [\![1, r]\!]$ the rational m_i has a p-th-root in $\mathbb{F}_Q$, and that for all $j \in [\![1, s]\!]$ the rational n_i has a q-th-root in $\mathbb{F}_Q$. Moreover since the character needs to be non trivial, $\mathbb{F}_Q$ has to contain a primitive p-th root of unity, i.e. $Q = 1 \pmod{p}$. After the reduction modulo Q, one can verify if $\phi_Q(s_i)$ is a p-power by computing an exponentiation with exponent $\frac{p-1}{Q}$. The composition of this and ϕ_Q will be the character χ_Q. Following the analysis of [6], such a prime Q can be found in time Poly(N) so finding R good primes can be done in Poly(NR) with the maximum of the Q to be also in Poly(NR). Finally if B is an upper bound for the size of the coefficients of $s_1, \dots, s_n$ then one can construct and apply the characters in time Poly$(BNRn)$. Then detecting the powers can be done using Algorithm 33 in polynomial time with respect to the entries.

Algorithm 33 Compute non trivial p-powers of a subgroup of K^* – `DetectPowers`

```
Require: A real Kummer extension $L/K$ of exponents $p, q$, $S = \langle s_1, \dots, s_n \rangle$ a
    subgroup of $K^*$
Ensure: $\lambda_1, \dots, \lambda_{n'} \in [\![0, p-1]\!]^n$ such that $\prod_{i=1}^n s_i^{\lambda_{j,i}}$ is a $p$-power in $K$, for all
    $j \in [\![1, n']\!]$.
 1: Generate sufficiently enough characters $\chi_{Q_1}, \dots, \chi_{Q_R}$.      ▷ Use OneGoodPrime
 2: $M \leftarrow [\chi_{Q_j}(s_i)]_{i,j} \in M_{n,R}(\mathbb{F}_p)$
 3: $N \leftarrow \ker(M)$                                              ▷ Left Kernel in $\mathbb{F}_p$
 4: return $N$ as a matrix in $\mathbb{Z}$
```

Remark that Algorithm 33 returns exponents corresponding to true $p-$powers with probability at least $1 - p^{-(R-n)}$ under assumption that the characters constructed are uniformly distributed in the dual of $S/(S \cap K^p)$. We never encountered failure during our computations.

Heuristic 5.1. *Let $S < K^*$ with L/K a real Kummer extension of exponents p, q. Then the characters χ_Q described previously are uniformly distributed in* $\mathrm{Hom}(S/(S\cap K^p), \mathbb{F}_p)$.

Reducing a basis subgroup

In order to find a basis of a subgroup $U < \mathcal{O}_K^\times$ one can use Pohst's modified LLL [84] algorithm on the matrix $\mathrm{ApproxLog}_L(U)$. In order to find a transformation matrix with small coefficients, one can follow [6] and compute a LLL on a matrix of the form $\Big[\mathrm{Id}|C\cdot \mathrm{ApproxLog}_L(U)\Big]$. This leads to a reduction in $\mathrm{Poly}(NB)$ if B is a bound on the size of the elements of $\mathrm{ApproxLog}_L(U)$, as we take C with size polynomial in N. Moreover, the use of a reducing algorithm allows us to find a basis of better quality. One can choose to use another reducing algorithm such as BKZ [94].

Reducing an element with respect to a lattice

In order to retrieve a short generator g of a principal ideal from another generator h, we mentioned that one can try to solve a CVP with respect to the Log-unit lattice. In order to do so, we followed [64] and computed the result of Babaï's nearest plane algorithm using Kannan's embedding technique. This technique can be used more generally to reduce an element $[h, \mathrm{ApproxLog}_L(h)]$ with respect to a sublattice $\mathrm{ApproxLog}_L(U)$ of $\mathrm{ApproxLog}_L(\mathcal{O}_L^\times)$, in order to control the size of the elements which are handled. Recall that if B is an upper bound of the norm of the vectors of the basis of $\mathrm{ApproxLog}_L(U)$ then one can consider the matrix

$$\left[\begin{array}{c|c} \mathrm{ApproxLog}_L(U) & \mathbf{0} \\ \hline \mathrm{ApproxLog}_L(h) & B \end{array}\right] = \left[\begin{array}{c|c} \mathrm{ApproxLog}_L(u_1) & 0 \\ \mathrm{ApproxLog}_L(u_2) & 0 \\ \vdots & \vdots \\ \mathrm{ApproxLog}_L(u_m) & 0 \\ \hline \mathrm{ApproxLog}_L(h) & B \end{array}\right].$$

Reducing it with a LLL algorithm is expected to reduce the last row to the Log-embedding of a shorter element in the same coset. In order to obtain again a transformation matrix with small coefficients, we consider a matrix of the form

$$\left[\mathrm{Id} \mid \begin{array}{c|c} C\times \mathrm{ApproxLog}_K(U) & \mathbf{0} \\ \hline C\times \mathrm{ApproxLog}_K(h) & B \end{array}\right].$$

We will denote by `RKEBabai`(U, h) this procedure.

Computing p-th roots

The authors of [6] were able to exhibit a recursive algorithm in order to compute square roots in a multiquadratic field. The method cannot be adapted to Kummer extensions of exponents bigger than 3. We then developed the method using the LLL algorithm that we presented in Chapter 4. Fix $y = x^p$ in a real Kummer extension L/K. Remark that since L/K is *real*, $\zeta_p \notin L$ and the polynomial $X^p - y$ has exactly one root in L. Moreover in a given extension we usually have to compute several roots. Thus we can use the same reduced basis lattice L_l for several elements or update the precision as needed. This strategy is efficient because one can evaluate the logarithm of $\|x\|_2$ quite accurately (experimentally) with the formulae $\texttt{EvaluteNorm}(x) = \frac{\ln\|y\|_2}{p}$. Therefore one can evaluate the norms of all the roots to be computed, and sort the elements by increasing norms. Let us denote by `Sort` this sorting procedure.

Finally, in order to reduce the time of computation, one can try to bound the norm of the powers. Let y be one of the powers outputted by `DetectPowers`, and $S = \langle s_1, \dots, s_n\rangle$ the subgroup of K^* given as input. Then one can reduce y with respect to ApproxLog(S^p) using `Kannan` as explained above. Experimentally, it allows the computations to run considerably faster.

Implementing these ideas, we obtain Algorithm 34 which computes roots of powers such as outputted by `DetectPowers`. In this context, we will write `InitBasisLatt` and `UpdateBasisLatt` the procedures which respectively initialise and update to a larger precision the basis lattice matrix of L/K (as defined in Section 4.2).

Algorithm 34 Compute the p-th roots in L/K – `ElementsFromPower`

```
Require: A Kummer extension $L = K(\sqrt[p]{m_1}, \dots, \sqrt[p]{m_r})$ with $K = \mathbb{Q}(\sqrt[q]{n_1}, \dots, \sqrt[q]{n_s})$, a subgroup $S = \langle s_1, \dots, s_n\rangle$ of $K^*$ and $V = \langle y_1, \dots, y_t\rangle < S^p$
    non-trivial $p$-th powers
Ensure: A basis $\langle x_1, \dots, x_t\rangle$ of $V^{1/p}$
 1: $Y \leftarrow \texttt{RKEBabai}(S^p, V)$          ▷ Reduce in the Log-representation
 2: $Y \leftarrow \texttt{Sort}(X)$
 3: $X \leftarrow \emptyset$
 4: $[L, U, l] \leftarrow \texttt{InitBasisLatt}(L/K)$
 5: for $i = 1$ to $n$ do
 6:     $[L, U, l] \leftarrow \texttt{UpdateBasisLatt}(L/K, \texttt{PrecisionEvaluation}(y_i), U)$
 7:     $x \leftarrow [x_i]_l$
 8:     $x \leftarrow \texttt{TestDecode}(L, t)$
 9:     $X \leftarrow X \cup \{x\}$
10: end for
11: return $X$
```

5.2.5 Computing the unit group:

In order to compute the unit group of a real Kummer extension of exponents p, q we are able to use Algorithm 30 several times. Indeed if L/K is a Kummer extension of exponents p, q then each of the minimal subextensions $L(\sqrt[p]{M_\alpha})$ can be written as $\mathbb{Q}(\sqrt[p]{M_\alpha})(\sqrt[q]{n_1}, \ldots, \sqrt[q]{n_s})$, i.e. a Kummer extension of $\mathbb{Q}(\sqrt[p]{M_\alpha})$ of exponent q. Therefore if one applies Algorithm 30 to L/K, when it reaches the simple subextensions $L(\sqrt[p]{M_\alpha})$ in step 2, one can again apply `KE_Units` instead of `UnitGroup`. This leads to Algorithm 35.

Algorithm 35 Compute the unit group of a Kummer extension L/K of exponents p, q. – `RKE_Units`

```
Require: A Kummer extension $L = K(\sqrt[p]{m_1}, \ldots, \sqrt[p]{m_r})$ with $K = \mathbb{Q}(\sqrt[q]{n_1}, \ldots, \sqrt[q]{n_s})$.
Ensure: A basis of the torsion-free part of the unit group $\mathcal{O}_K^\times$.
 1: if ($r = 1$ and $s \leqslant 1$) then
 2:     return UnitGroup($L$).
 3: end if
 4: if ($r = 1$ and $s > 1$) then
 5:     return KE_Units($L/\mathbb{Q}(\sqrt[p]{m_1})$).      ▷ Compute a basis of $U = O_L^\times$ by
    considering $L$ as a Kummer extension of $\mathbb{Q}(\sqrt[p]{m_1})$.
 6: else
 7:     Choose $u, v$ two independent elements of $\widetilde{\mathrm{Hom}(L/K)}$.
 8:     Recursively compute a basis of $U = \mathcal{O}_{L^u}^\times \mathcal{O}_{L^{uv}}^\times \ldots \mathcal{O}_{L^{u^{p-1}v}}^\times \mathcal{O}_{L^v}^\times$
 9:     $V \leftarrow$ DetectPowers($U, p$)
10:     $V \leftarrow$ ElementsFromPower($V, p$)
11:     $U \leftarrow$ BasisFromGeneratingSet($\langle U, V \rangle$)
12:     return $U$
13: end if
```

Theorem 5.5. *Consider $L = K(\sqrt[p]{m_1}, \ldots, \sqrt[p]{m_r})$ with $K = \mathbb{Q}(\sqrt[q]{n_1}, \ldots, \sqrt[q]{n_s})$ a real Kummer extension with p and q prime integers such that $[L : \mathbb{Q}] = p^r q^s$. Under the assumption of Heuristic 5.1 and GRH Algorithm 35 heuristically computes $\mathcal{O}_L^\times$ in $\mathrm{Poly}(\ln(|D_L|))L_P(2/3 + \epsilon, c)$ for some $c > 0$ and $\epsilon > 0$ as small as desired, with probability at least $1 - (pq)^{-N}$, where P is the product of all primes dividing the m_i and n_j.*

Proof. Let us study the running time of the algorithm. The analysis is very similar to the one of the $S-$units computations done in [17]. Assume we obtained the unit groups of the $p + 1$ subfields, with elements given in compact representation. Denote by $U_1, \ldots, U_{p+1}$ these groups. Their union is a generating family of U. Then in order to compute $\mathcal{O}_L^\times$, one needs to reduce this family to a basis of U, apply the characters and detect powers, compute the associated roots and finally reduce the obtained family to a basis of $\mathcal{O}_L^\times$.

First let us remark that from the simple subextensions of L/K, there are only $\mathrm{Poly}(\ln(|D_L|)$ calls to `BasisFromGeneratingSet`. Each implies a loss in precision of $\mathrm{Poly}(NB)$ where B is the actual precision in the Log representation. Therefore given we obtain a certain precision in $\mathrm{Poly}(\ln|D_L|)$ for the approximate logarithm of the units of the simple subextensions of L/K it is possible to finish the algorithm with also a precision in $\mathrm{Poly}(\ln|D_L|)$ for the Log-unit lattice. Since the reasoning will be the same for the recursive part of step 5, we can assume that at each step the precision in the Log representation is in $\mathrm{Poly}(\ln|D_L|)$.

Now assume that we reduce a family of elements which are all in compact representation. As we saw the coefficients of the transformation matrix of the LLL will all have their logarithms in $\mathrm{Poly}(\ln|D_L|)$. Moreover the rank of the families we will reduce will also be in $\mathrm{Poly}(\ln|D_L|)$, as well as the length of the product defining each element. Therefore one can compute the compact representation of the elements of the basis in $\mathrm{Poly}(\ln|D_L|)$.

Once we have a basis $\langle u_1,\dots,u_n\rangle$ of U in compact representation, we need to apply `DetectPowers`. But each u_i is given as $u_{i,0}u_{i,1}^{p}\dots u_{i,k}^{p^k}$ with each $u_{i,j}$ of size polynomial in the logarithm of the discriminant. Thus the image of u_i by a character χ_Q is $\chi_Q(u_{i_0})$, which can be computed in polynomial time. We saw in the discussion for the general case that we ensure the searched probability of success by choosing a number of characters polynomial in N, assuming Heuristic 5.1. Therefore the cost of applying is in $\mathrm{Poly}(\ln|D_L|)$.

The exponents found by `DetectPowers` are less than $p-1$ so one can easily compute the compact representation of any of the powers detected. Given a power $v = v_0v_1^{p}\dots v_k^{p^k}$ in compact representation, its p-th root is $\sqrt[p]{v_0}v_1v_2^{p}\dots v_k^{p^{k-1}}$. One only has to compute the root of v_0 whose size is in $\mathrm{Poly}(\ln|D_L|)$. This will be done in time $\mathrm{Poly}(\ln|D_L|)$. Since there are only $O(N)$ roots to compute, the overall running time of the roots extraction together with the last call to `BasisFromGeneratingSet` is also in $\mathrm{Poly}(\ln|D_L|)$. During the descent to the simple subextension of L/K the algorithm reaches $O([L:K])$ number of subextensions so the cost of the algorithm will be in $O([L:K])\max_\alpha T_U(L_\alpha) + O([L:K])\mathrm{Poly}(\ln|D_L|)$, where the L_α are the simple subextensions of L/K and $T_U(L_\alpha)$ is the running time of Algorithm 35 when applied to L_α. The whole analysis done above can been applied to this part of the algorithm. Therefore we obtain a complexity in $\mathrm{Poly}(\ln D_L)\times S_U$ where S_U designates the maximum running time on the minimal subfields reached by the algorithm to compute the unit group. These fields are of the form $\mathbb{Q}(\sqrt[p]{M})\otimes\mathbb{Q}(\sqrt[q]{N})$ with M being a product of m_i and N a product of n_j. Given the discriminant of such fields [100], one obtains S_U to be in $e^{\tilde{O}((\ln P)^{2/3})}$ by applying the algorithm of Biasse and Fieker [15]. $\square$

5.2.6 Solving the PIP

As we saw the procedure to find a generator of a principal ideal is very similar to the one to compute the unit group. Therefore we obtain easily Algorithm 36. The analysis of the running time is similar to the one of Algorithm 35 which gives the same complexity since solving the PIP on the subfields of dimension pq is also sub-exponential.

Theorem 5.6. *Consider $L = K(\sqrt[p]{m_1}, \ldots, \sqrt[p]{m_r})$ with $K = \mathbb{Q}(\sqrt[q]{n_1}, \ldots, \sqrt[q]{n_s})$ a real Kummer extension with p and q prime integers such that $[L : \mathbb{Q}] = p^r q^s$ and a principal ideal I. Under the assumption of Heuristic 5.1 and GRH Algorithm 36 heuristically computes a generator of I in $\mathrm{Poly}(\ln(\mathrm{N}_{L/\mathbb{Q}}(I)), \ln(|D_L|))L_P(2/3 + \epsilon, c)$ for some $c > 0$ and $\epsilon > 0$ as small as desired, with probability at least $1 - (pq)^{-N}$, where P is the product of all primes dividing the m_i and n_j.*

Algorithm 36 Solve the PIP in a Kummer extension of exponents p, q – `RKE_PIP`

```
Require: A principal ideal $I$ of a Kummer extension $L = K(\sqrt[p]{m_1}, \ldots, \sqrt[p]{m_r})$ with
    $K = \mathbb{Q}(\sqrt[q]{n_1}, \ldots, \sqrt[q]{n_s})$, the unit group $\mathcal{O}_L^\times$.
Ensure: A generator $g$ of $I$.
 1: if ($r = 1$ and $s \leqslant 1$) then
 2:     return Generator(I).
 3: end if
 4: if ($r = 1$ and $s > 1$) then
 5:     return KE_PIP($L/\mathbb{Q}(\sqrt[p]{m_1})$).  ▷ Compute a generator of $I$ by considering $L$
    as a Kummer extension of $\mathbb{Q}(\sqrt[p]{m_1})$.
 6: else
 7:     Choose $u, v$ two independent elements of $\widetilde{\mathrm{Hom}(L/K)}$.
 8:     Recursively compute generators of $\mathrm{N}_{L^u}(I)\mathrm{N}_{L^{uv}}(I), \ldots, \mathrm{N}_{L^{u^{p-1}v}}(I), \mathrm{N}_{L^v}(I)$
    and use Equation 5.2 to have $h$ a generator of $I^p$.
 9:     return ElementsFromPower($[\mathcal{O}_L^\times, h], p$).
10: end if
```

5.3 Experimental results

We implemented the algorithms for real Kummer extensions in MAGMA [22], with the procedures described but without the compact representation of elements, which leads to exponential algorithms.

- We study in Subsection 5.3.1 the probability to retrieve a short generator of a principal ideal through an attack using the algorithms presented in Section 5.2 ; we computed data for Kummer extensions with one and two exponents, and compare the results to the ones of [6, 64]. This allows us to identify Kummer fields with degree p^2 and defined by small integers to be fields over which the SPIP is more difficult to solve.

- Finally in Subsection 5.3.2 we study further the geometrical situation. In particular we compute the size of the target vector normalised by the volume of the Log-unit lattice and the quality of the basis obtained for the Log-unit lattice through Algorithm 35. We focus on Kummer extensions with one exponent with degree p^2 and compare them with other number fields.

5.3.1 Probability of solving the SPIP

The first way we studied the possibility of real Kummer extensions was to launch attacks with Algorithm 36. As a matter of fact, we did not do proper attacks because computing ideal norms can be quite long even though the theoretical complexity is polynomial. However the knowledge of the secret key allows us to compute the HNF of the norms efficiently, and the rest of the attack is unchanged. We tried to retrieve generators of principal ideals (g) such that the coefficients of the generators g are drawn uniformly in $\{-1, 0, 1\}$. The previous observations in [6, 64] seemed to show two phenomena. Given a sequence $m = (m_1, \dots, m_r)$ defining the fields, the probability of retrieving a generator increased when :

- the length r of the sequence defining the field was increasing;
- the global size of the entries of the sequence, i.e. $\prod_{i=1}^{r} m_i$, was increasing,

Part of our work has been to verify that it happens on all Kummer extensions.

Kummer extensions with one exponent

First let us consider fields of the form $K = \mathbb{Q}(\sqrt[p]{m_1}, \dots, \sqrt[p]{m_r})$. We present the results obtained in Tables 5.2, 5.3, 5.4 and 5.5. There is one table for each exponent p defining the field, except for Table 5.5 which presents the results for the three exponents (11, 13, 17). For each exponent we computed attacks for fields defined by sequences of increasing length and increasing coefficients ; moreover the coefficients are consecutive prime numbers. For each field we provide the probability of retrieving a generator when LLL or BKZ$_{20}$ is used to reduce the different bases during the algorithms.

Table 5.2: Experimental results for Kummer extension of $\mathbb{Q}$ with exponent 3

(a) $r = 2$ and 3

Sequence length r	2					3				
Rank of the lattice $r_1 + r_2 - 1$	4					13				
First coefficient m_1	2	3	5	7	11	2	3	5	7	11
Success LLL (%)	32	89.2	98	97.6	99.99	43.6	98	100	100	100
Success BKZ (%)	29.4	92.2	98.4	98	100	47.4	99.4	100	100	100

(b) $r = 4$ and 5

Sequence length r	4					5				
Rank of lattice $r_1 + r_2 - 1$	40					121				
First coefficient m_1	2	3	5	7	11	2	3	5	7	11
Success LLL (%)	58.6	100	100	100	100	77.6	100	100	100	100
Success BKZ (%)	64.6	100	100	100	100	74.3	100	100	100	100

Table 5.3: Experimental results for Kummer extension of $\mathbb{Q}$ with exponent 5

Sequence length r	2					3				
Rank of lattice $r_1 + r_2 - 1$	12					62				
First coefficient m_1	2	3	5	7	11	2	3	5	7	11
Success LLL (%)	53.6	74.8	100	100	100	71.6	97.2	100	100	100
Success BKZ (%)	54.6	69.6	100	100	100	68.6	95.8	100	100	100

Table 5.4: Experimental results for Kummer extension of $\mathbb{Q}$ with exponent 7

Sequence length r	2					3				
Rank of lattice $r_1 + r_2 - 1$	24					171				
First coefficient m_1	2	3	5	7	11	2	3	5	7	11
Success LLL (%)	86.6	100	100	100	100	80.6	100	100	100	–
Success BKZ (%)	84.9	100	100	100	100	98.7	100	100	100	–

Table 5.5: Experimental results for Kummer extension of $\mathbb{Q}$ with degree p^2 and exponents 11, 13 and 17.

Field exponent	11					13					17	
Rank of lattice	60					84					144	
First coefficient m_1	2	3	5	7	11	2	3	5	7	11	2	3
Success LLL (%)	77.6	100	100	100	100	19.3	99.6	100	100	100	0	29.1
Success BKZ (%)	90.7	100	100	100	100	70.0	100	100	100	100	12.4	100

We can remark that the two phenomena described before seem to be true for all exponents p. Moreover the probability of success seems to converge quickly to one. For similar degrees and rank of $\mathrm{Log}_K(\mathcal{O}_K)$ we can remark that we obtain a better probability of success with fields defined by longer sequences and smaller exponents. Compare for instance fields of degree 7^3 in Table 5.4 and fields of degree 13^2 or 17^2 in Table 5.5.

Fields with degree p^2 : Let us now focus our attention on the subclass of fields of the form $K = \mathbb{Q}(\sqrt[p]{m_1}, \sqrt[p]{m_2})$. First we see that again the probability of success converges quickly to 1 when m_1 increases. Now fix $(m_1, m_2) = (2, 3)$ and let p vary. One can find the percentages of success plotted in Figure 5.1.

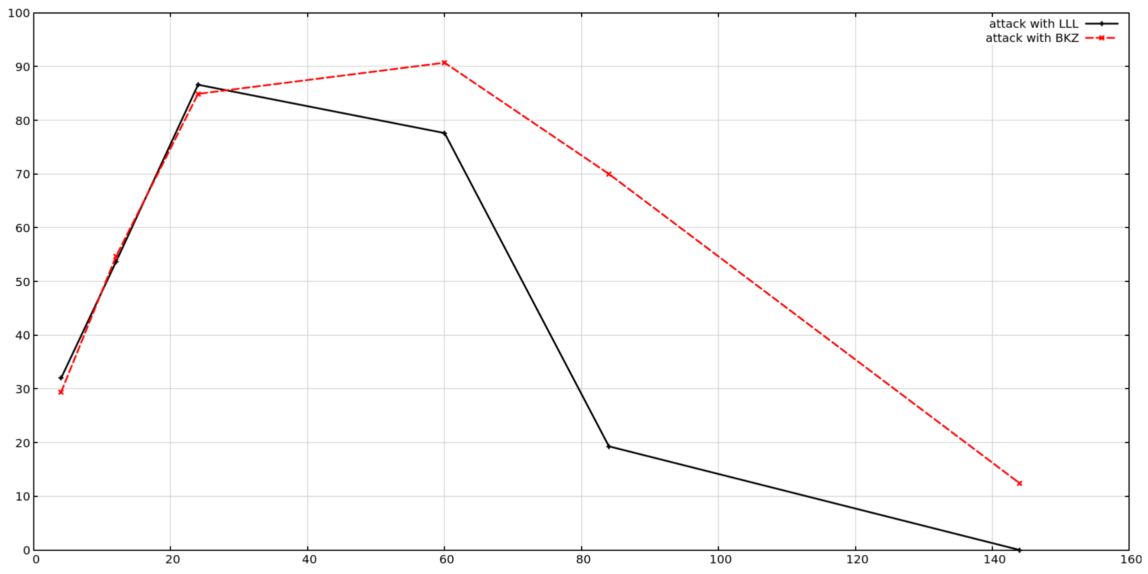


Figure 5.1: Percentage of success of an attack with LLL or BKZ for fields $K = \mathbb{Q}(\sqrt[p]{2}, \sqrt[p]{3})$ plotted against the rank $r_1 + r_2 - 1$ of $\mathrm{Log}_K(\mathcal{O}_K)$

We can notice that for one or the other method used as a reduction algorithm throughout the procedures, the probability of retrieving a short generator starts to increase but decreases when p is larger than 11. It converges to 0 when using LLL and is bigger when using BKZ_{20} but is still quickly decreasing.

Remark 35 (Importance of studying high degree number fields)**.** One important observation is that computations on high degree number fields were required to observe meaningful data. Indeed when restricted to fields with degree less than 121, i.e. to primes strictly smaller than 11, the probability of success of an attack is quickly increasing and there is no difference between using LLL or BKZ_{20}. *This highlights the need to work over high degree number fields.*

Finally one could consider Kummer fields of degree p^2 defined by small integers as an alternative to number fields already used in cryptography such as cyclotomic fields. Indeed, in addition to the data gathered here, their structure could be used to build an efficient arithmetic as done over multiquadratic fields in [6]. One could also consider Kummer fields of degree p if the pattern concerning the probability of success (decreasing with the length of the sequence) is still valid. However we cannot confirm or invalidate it. We only have access to the classical algorithms to do computations on these fields, thus preventing examining fields with high degree.

Kummer extensions with two exponents

Consider now real Kummer extensions of the form $L = K(\sqrt[p]{m_1}, \ldots, \sqrt[p]{m_r})$ with $K = \mathbb{Q}(\sqrt[q]{n_1}, \ldots, \sqrt[q]{n_s})$. We tried to verify whether the phenomena observed in [6, 64] and mentioned earlier were still true over such fields or not. To do so, we computed data for several fixed ground fields K and varying parameters for the extension L. Because of efficiency reasons, we were restricted in our choice of parameters. Indeed, our implementation is way slower over Kummer extensions with two exponents than extensions with one exponent. We only present the probabilities with LLL because the ones with BKZ_{20} are very similar, due to the fact that the ranks of the Log-unit lattices manipulated are small.

Simple Kummer field as ground field: First let us consider fields such that K is a simple Kummer field $\mathbb{Q}(\sqrt[q]{n})$ and $L = K(\sqrt[p]{p_1}, \sqrt[p]{p_2})$ with p_1, p_2 being consecutive prime numbers. The data gathered can be found in Tables 5.6, 5.7 and 5.8.

Table 5.6: Success of an attack (in %) over Kummer extensions of the form $L = K(\sqrt[p]{p_1}, \sqrt[p]{p_2})$ with $K = \mathbb{Q}(\sqrt[2]{n})$

Exponent p	3					5					7				
$r_1 + r_2 - 1$	9					25					49				
Coefficient p_1	2	3	5	7	11	2	3	5	7	11	2	3	5	7	11
$n = 2$	–	59	71.6	66.2	58	–	68.6	74.6	72.8	65.6	–	82.6	77.6	71.2	71.9
$n = 5$	10	56.3	–	47.7	51.7	13.7	59	–	63.4	54.7	74	65	–	62	52
$n = 13$	27.3	80.4	90.3	88.7	87.7	–	68.4	86.3	87.3	90.6	83.3	90.3	89	86	79.8

Table 5.7: Success of an attack (in %) over Kummer extensions of the form $L = K(\sqrt[p]{p_1}, \sqrt[p]{p_2})$ with $K = \mathbb{Q}(\sqrt[3]{n})$

Exponent p	5				
$r_1 + r_2 - 1$	37				
Coefficient p_1	2	3	5	7	11
$n = 2$	–	78.1	82	79.3	81.3
$n = 5$	35.4	98	–	100	98.6
$n = 13$	55.4	78	98.3	99.7	–

Table 5.8: Success of an attack (in %) with over Kummer extensions of the form $L = K(\sqrt[p]{p_1}, \sqrt[p]{p_2})$ with $K = \mathbb{Q}(\sqrt[5]{n})$

Exponent p	3				
$r_1 + r_2 - 1$	22				
Coefficient p_1	2	3	5	7	11
$n = 2$	–	77	78.6	73.9	69.9
$n = 5$	50.7	95.3	–	98.3	98
$n = 13$	55	93	98	99.7	100

We can see that the results are different for these fields than for Kummer extensions with one exponent. For each pair (p, q) it seems that the probability of success does not converge to 1 when the coefficients (p_1, p_2) increase ; for some pairs the probability is even decreasing. We are still able to retrieve a high percentage of generators, but one should remark that the dimensions are all relatively low. We mentioned in Remark 35 the importance of studying high dimensional number fields i.e. with dimension at least greater than 100, and we stress that the data we were able to produce regarding Kummer extensions with two exponents do not meet this requirement. Thus the observations made from these data might not be representative of the asymptotic behaviours.

Increasing $[L : K]$ with constant exponent: Now let us consider extensions $L = K(\sqrt[p]{m_1}, \ldots, \sqrt[p]{m_r})$ with fixed K and p, with increasing length sequence r of consecutive prime numbers.

Table 5.9: Success of an attack over Kummer extensions of the form $L = K(\sqrt[3]{p_1}, \ldots, \sqrt[3]{p_r})$ with $K = \mathbb{Q}(\sqrt[q]{n})$

Length r	2		3		4	
Exponent q	2	5	2	5	2	5
$r_1 + r_2 - 1$	9	22	27	67	81	202
Success with LLL (%)	61.4	79.6	85.4	94.7	81.2	–

The data in Table 5.9 seems to show that again, the phenomena observed over Kummer fields with one exponent cannot be seen as clearly over Kummer extensions with two exponents, at least for $q = 2$.

Conclusion: The probabilities of successfully retrieving the private key seem to be smaller and to differ much more than for the previous type of fields. It could be an indication that breaking the regularity of the field structure makes the attack more difficult. However one has to remark that we lack data, and that the ones we obtained are essentially over fields with relatively low degrees.

5.3.2 Analysis of the geometrical situation

Let us now focus on Kummer extensions with one exponent, since we are able to compute data for high dimensional fields. Moreover recall that we identified Kummer extensions of degree p^2 defined by the sequence $(2, 3)$ as fields for which recovering a short generator through the Log-unit lattice could be more difficult than over other number fields. Thus all Kummer extensions considered further are defined by sequences of the first prime integers. In order to study further the situation we looked into the possibility of recovering a short generator through an enumeration process. In order to evaluate the cost of enumerations, we used the function `EnumerationCost`(L, m^2) of MAGMA. It computes an estimation of the number of nodes to visit during an enumeration process of short vectors of a lattice L within the ball $B(0, m)$. Moreover we studied the quality of the basis obtained by computing several parameters. Given a basis B (whose vectors are sorted by increasing norms), evaluating its orthogonality can be difficult. Let us denote by r and V respectively the rank and the volume of the lattice generated by B. We chose to compute:

1. the Hermite factor $\delta_0 = \frac{\|b_1\|}{\sqrt[r]{V}}$ which is used to evaluate the quality of basis reduction on random lattices;
2. the orthogonality defect $\delta = \sqrt[r]{\frac{\prod_{i=1}^{r} \|b_i\|}{V}}$ which expresses the overall orthogonality of the basis.

We gathered data of cyclotomic fields, NTRU Prime fields and Kummer fields. We computed the unit group of the first two categories using the generic algorithm of MAGMA `UnitGroup` up to degree 60. In order to obtain data on cyclotomic fields of larger degree we used the subgroup C of cyclotomic units, which has a very small index [34]. For some fields they are even equal, for example for power-of-2 cyclotomics (under GRH). Even if C is not $\mathcal{O}_K$ one can argue that it is close to it and is used by the authors of [34] to solve the SPIP over cyclotomic fields.

Norm of the target vector

One important geometrical parameter is the size of the target when compared to the volume of the Log-unit lattice, in order to know if retrieving it through a CVP computation or an enumeration process is conceivable. In addition to the size of the target vector we studied the cost one would obtain for an enumeration.

Let us recall a quick result which can be found in [9, 34].

Lemma 5.7. *Let K be a number field, H be the subspace of $\mathbb{R}^n$ orthogonal to $\mathbf{1} = (1, \ldots, 1)$ and p_H be the orthogonal projection on H. Then for any $g \in K$ one has* $\mathrm{Log}_{\mathbf{K}}(\mathbf{g}) = p_H(\mathrm{Log}_{\mathbf{K}}(\mathbf{g})) + \frac{\ln |\mathrm{N}_{K/\mathbb{Q}}(g)|}{n}\mathbf{1}$.

One can conclude from Lemma 5.7 that if g is the secret key, then the norm of the target is

$$\sqrt{\sum_{i=1}^{n} \left(\ln |\sigma_i(g)| - \frac{\|\mathrm{Log}_{\mathbf{K}}(\mathbf{g})\|_1}{n} \right)^2}.$$

For each field we computed the ratio of the norm of $p_H(\mathrm{Log}_K(g))$ by the scaled volume of the Log-unit lattice $\sqrt[r]{V_K}$ where $r = r_1 + r_2 - 1$. We computed the median value of this ratio for each set of keys, and the corresponding enumeration cost. Let us denote by M_K said median value, and EC_K the bit-size of the corresponding enumeration cost.

Prime degree fields: First consider fields with prime degree – or conductor for cyclotomic fields – smaller than 60. The data can be found in Table 5.10. We show the enumeration cost only after LLL because for those dimensions the values obtained after BKZ$_{20}$ are the same.

Table 5.10: Data concerning the target $p_H(\mathrm{Log}_{\mathbf{K}}(\mathbf{g}))$ for Kummer extensions, cyclotomic fields and NTRU Prime fields with prime degree or conductor

	Degree p	11	13	17	19	23	29	31	37	41	43	47	53	59
Kummer field $\mathbb{Q}(\sqrt[p]{2})$	M_K	0.48	0.46	0.42	0.42	0.43	0.39	0.38	0.37	0.35	0.35	0.34	0.32	0.31
	EC_K	1.172	0.89	0.61	0.55	0.82	0.47	0.80	0.77	0.53	0.85	0.81	0.49	0.89
NTRU Prime	M_K	1.34	1.32	1.17	1.11	1.05	0.97	0.96	0.91	0.88	0.84	0.84	0.77	0.74
	EC_K	4.93	5.27	4.58	4.10	3.63	2.56	2.82	2.38	1.77	2.09	2.88	2.05	2.50
Cyclotomics	M_K	0.81	0.83	0.82	0.84	0.83	0.84	0.85	0.85	0.86	0.86	0.86	0.86	0.87
	EC_K	3.14	3.55	3.89	4.19	4.39	4.90	4.77	485	5.30	5.69	5.50	5.11	6.29

From the data gathered, the targets are bigger over NTRU Prime fields than over cyclotomic or Kummer fields for very small primes, but the values decrease quickly. The targets seem to have a relatively stable size over cyclotomic fields, which ends up being the largest for bigger prime numbers. Kummer fields present the smallest target vectors. This leads to the same phenomenon for the corresponding enumeration cost. However these are still quite small but it was to be expected over lattices with small ranks. One can find the plot of the minimum and maximum value of the ratio $\left\|p_H(\mathrm{Log}_{\mathbf{K}}(\mathbf{g}))\right\|_2 / V_K^{1/r}$ in Figure 5.2. We can remark that the values over cyclotomic fields and NTRU prime fields are getting closer with p increasing. The targets over Kummer fields are again smaller. If this trend remains true asymptotically it would indicate that Kummer extensions of prime degrees are weaker than fields of the two other types. However it should be noted that the difference is quite small, as well as the rank of the lattices considered.

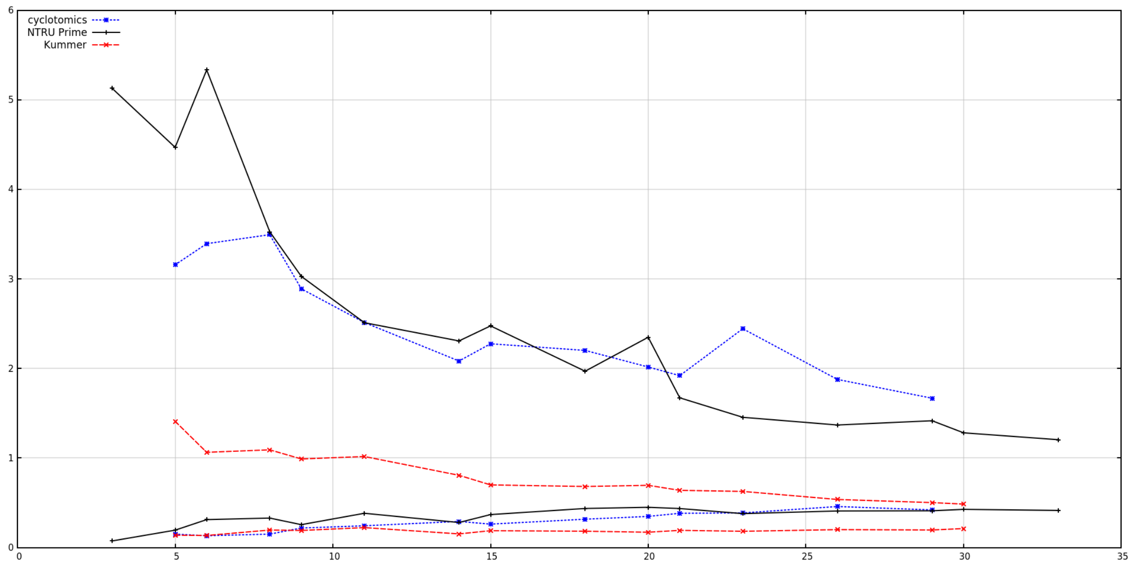


Figure 5.2: Minimal and maximal values of $\|p_H(\mathrm{Log}_{\mathbf{K}}(\mathbf{g}))\|_2 / V_K^{1/r}$ plotted against the rank of $\mathrm{Log}_K(\mathcal{O}_K^\times)$ over fields of prime degree or conductor

Kummer fields with degree p^2: The attacks showed that the SPIP seems to be more resistant over fields of the form $\mathbb{Q}(\sqrt[p]{2}, \sqrt[p]{3})$, so we will focus on them. In order to have a better idea of the situation, let us compare them with:

- cyclotomic fields of prime conductor p;
- cyclotomic fields of the form $\mathbb{Q}(\zeta_{2^n})$;
- Kummer fields of degree p^3 and Kummer fields of exponent 3 and defined by successive primes i.e. of the form $\mathbb{Q}(\sqrt[3]{2}, \sqrt[3]{3}, \ldots, \sqrt[3]{p_r})$.

Remember that in order to compute data for high degree cyclotomic fields, we considered C the subgroup of cyclotomic units. Again we computed the median values of the quotients $\|p_H(\mathrm{Log}_{\mathbf{K}}(\mathbf{g}))\|_2 / V_K^{1/r}$ and the corresponding enumeration costs. One can find the values corresponding to the first parameter plotted in Figure 5.3.

We can remark that the values for Kummer extensions of square degrees are close to the ones for cyclotomic fields, in particular the ones of the form $\mathbb{Q}(\zeta_{2^n})$. Moreover the values for cyclotomic fields with conductor of the form p^k with $k \geqslant 2$ are also similar, even if we did not plot them for clarity purposes. For Kummer fields of degree p^3, the plot suggests that the values could asymptotically be close to the ones over the previous fields. However we cannot confirm this because the state of our implementation does not allow us to compute the units for the following prime $p = 11$, which corresponds to a field of degree 1331. We can see that the size of targets over multicubic fields is decreasing quickly, which is consistent with the probability of success already observed. This also confirms the differences between fields with increasing exponents such that the defining sequence has a constant

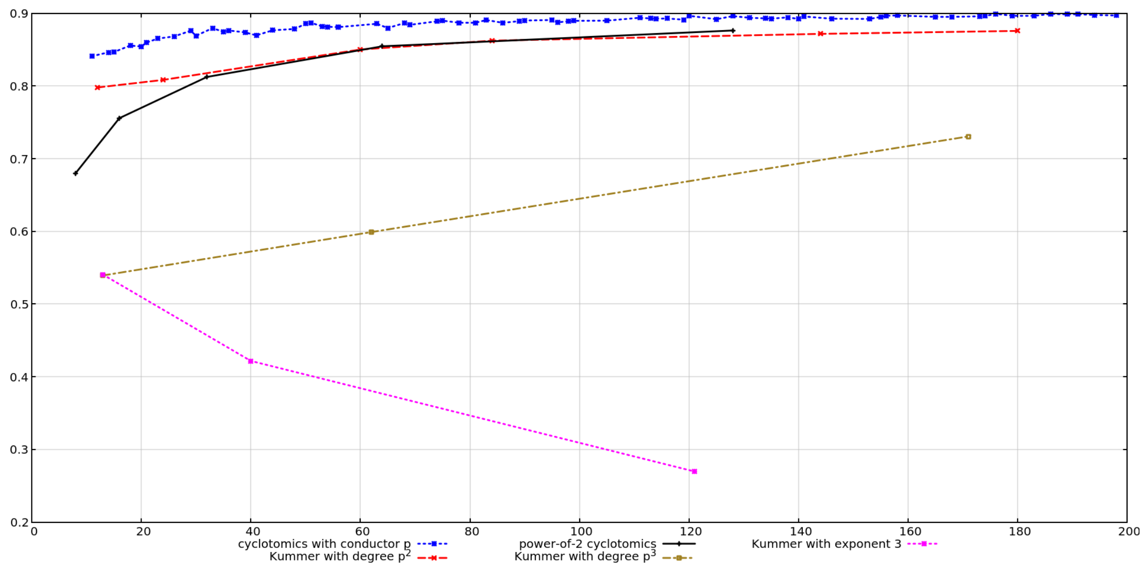


Figure 5.3: Median values M_K plotted against $r_1 + r_2 - 1$ over Kummer fields of degrees p^2 and p^3, and over different types of cyclotomic fields

length, and fields with a constant exponent such that the length of the defining sequence is increasing.

With these observations and the ones regarding number fields with prime degrees, one could expect to obtain similar enumeration costs for cyclotomic and Kummer fields. However we can see in Figures 5.4 and 5.5 – which show the corresponding enumeration costs with the use of LLL and BKZ$_{20}$ respectively – that the costs are low over cyclotomic fields (and close one to each other) but asymptotically bigger over Kummer fields of degree p^2 and p^3. Again the situation is worse for Kummer fields of degree p^2 than p^3. Regarding the influence of BKZ$_{20}$, it has again a positive and noticeable impact for ranks greater than 80 i.e. degrees greater 160, and only over Kummer fields of degree p^2. These observations coupled with the values of the enumeration cost obtained seem to indicate that Kummer extensions of degree p^2 could be better options than cyclotomic fields when it comes to building a cryptosystem whose security relies on the hardness of solving the PIP. Indeed for the field $\mathbb{Q}(\sqrt[17]{2}, \sqrt[17]{3})$, the enumeration cost after BKZ$_{20}$ is still large enough to prevent an enumeration process.

Basis of Log-unit lattice

As mentioned before, we studied further the situation by computing several parameters to evaluate the quality of the basis of $\mathrm{Log}_K(\mathcal{O}_K)$ for the fields K considered. Results of these computations are gathered in Table 5.11 for fields with prime degree or conductor less than 60, and in Figures 5.6, 5.7 and 5.8 for the same type of fields considered in the previous analysis. Again, we will denote by EC the bit-size of the

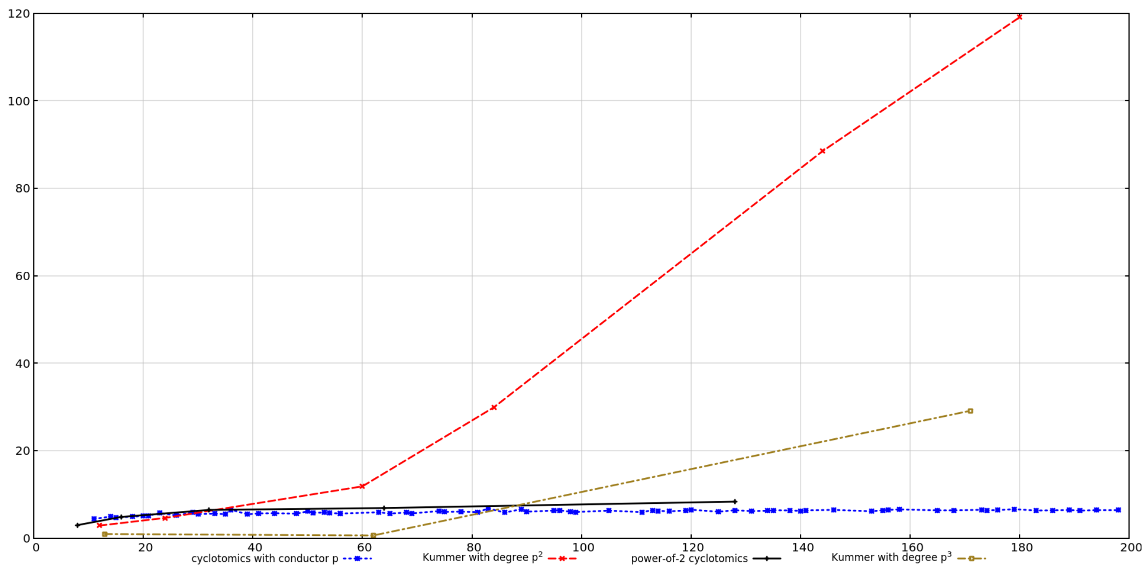


Figure 5.4: Median values of enumeration cost EC_K plotted against $r_1 + r_2 - 1$ over Kummer fields of degrees p^2 and p^3, and over cyclotomic fields, after LLL reductions

enumeration cost considered.

Table 5.11: Data concerning the Log-unit lattice of Kummer extensions, cyclotomic fields and NTRU Prime fields with prime degree or conductor

	Degree p	11	13	17	19	23	29	31	37	41	43	47	53	59
Kummer fields $\mathbb{Q}(\sqrt[p]{2})$	δ_0	0.81	0.76	0.722	0.70	0.66	0.62	0.59	0.56	0.54	0.53	0.51	0.49	0.47
	δ	1.08	1.08	1.12	1.11	1.15	1.21	1.22	1.27	1.26	1.36	1.38	1.43	1.48
	EC for $V_K^{1/r}$	4.11	4.33	4.61	4.60	4.73	4.74	4.95	5.42	5.03	5.97	6.02	6.42	7.27
NTRU Prime fields	δ_0	0.15	0.12	0.08	0.07	0.05	0.04	0.04	0.03	0.3	0.02	0.02	0.02	0.01
	δ	1.05	1.06	1.09	1.10	1.13	1.16	1.14	1.72	1.20	1.17	1.29	1.24	1.32
	EC for $V_K^{1/r}$	3.20	3.38	3.30	3.24	3.26	3.78	3.05	2.83	2.38	2.85	3.99	3.14	4.10
Cyclotomic fields $\mathbb{Q}(\zeta_p)$	δ_0	1.12	1.13	1.14	1.15	1.16	1.17	1.17	1.18	1.19	1.20	1.21	1.20	1.22
	δ	1.14	1.17	1.22	1.22	1.26	1.25	1.26	1.31	1.30	1.31	1.34	1.36	1.32
	EC for $V_K^{1/r}$	3.94	4.43	5.07	5.33	5.73	6.21	6.12	6.28	6.69	6.89	6.89	6.57	7.64

Hermite factor: One can see in Table 5.11 that fields of all three types have short smallest vector, especially NTRU Prime fields. Moreover their orthogonality defect are similarly small, indicating that the basis obtained for the Log-unit lattice is relatively well reduced. This is also supported by the fact that the values are the same with LLL or BKZ$_{20}$, which is why we present only one set of data. The situation is very similar over Kummer fields of degree p^2, cyclotomic fields of prime conductor with higher degrees and power-of-2 cyclotomics as shown in Figure 5.6, where the δ_0 is plotted. Again there is only one plot because the values are not modified by BKZ$_{20}$. We can remark that the different plots for different types of number fields look like the ones found in Figure 5.3.

Orthogonality defect: Now let us look at the orthogonality defect δ for high degree number fields. We plotted the values obtained after LLL in Figure 5.7 and

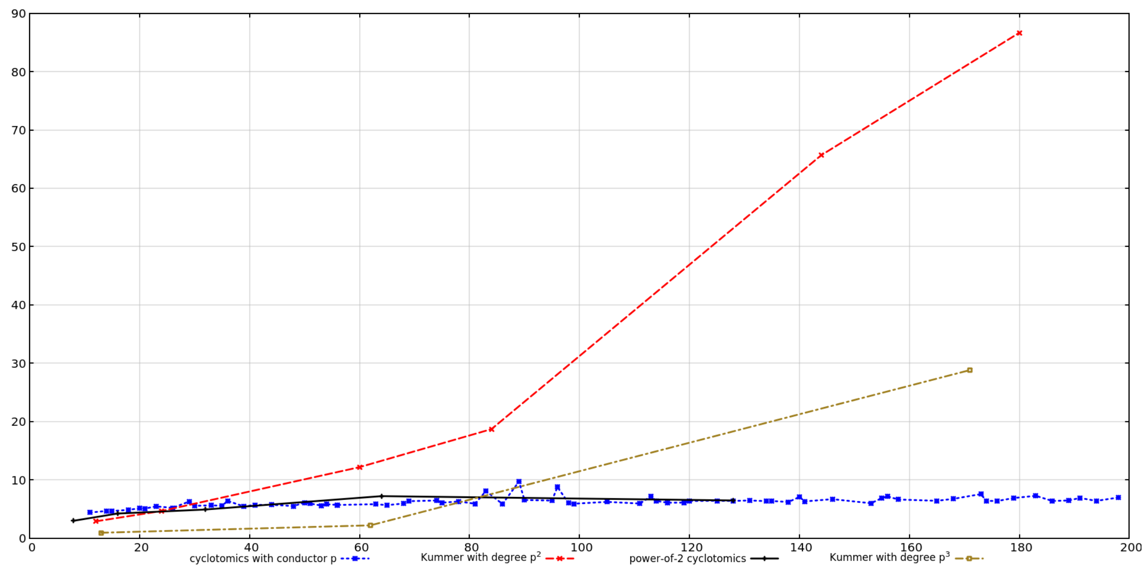


Figure 5.5: Median values of enumeration cost EC_K plotted against $r_1 + r_2 - 1$ over Kummer fields of degrees p^2 and p^3, and over cyclotomic fields after BKZ_{20} reductions

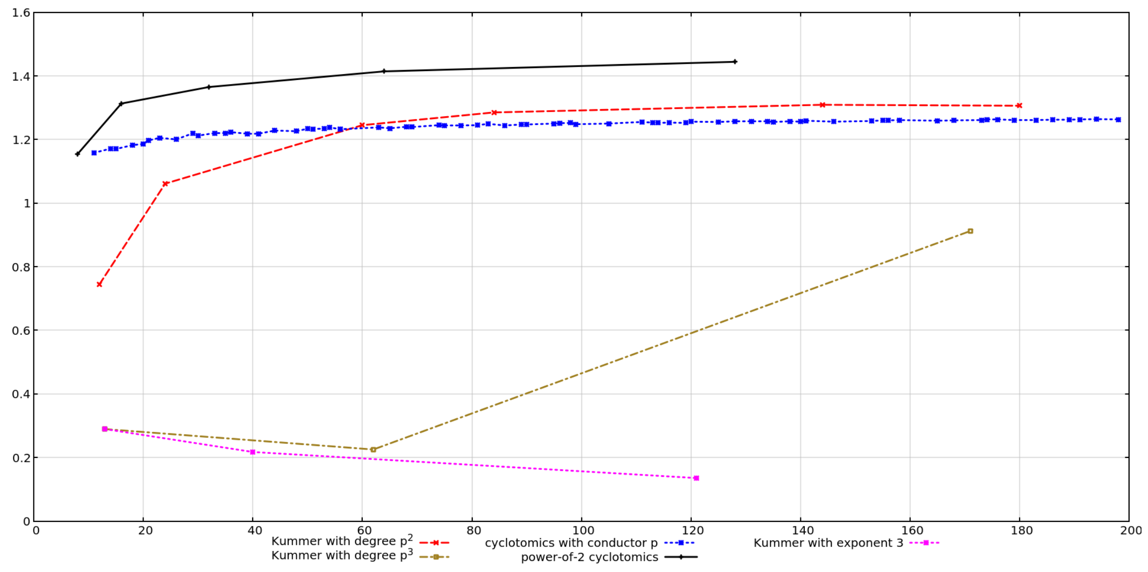


Figure 5.6: Values of δ_0 plotted against r_1+r_2-1 over Kummer fields of degrees p^2 and p^3, and over cyclotomic fields after LLL reductions

after BKZ_{20} in Figure 5.8. One can notice that the only fields for which BKZ_{20} has a significant impact are Kummer fields with degree p^2, as it was the case for the enumeration cost shown in Figures 5.4 and 5.5. This indicates that for these fields, the basis of the Log-unit lattice obtained by our procedures is not well reduced, and better reduction algorithms modify the basis. This is completely different from cyclotomic fields where the basis formed by cyclotomic units are massively orthogonal and are not modified by reduction algorithms. We can also conclude from the values for Kummer extensions of degree p^3 that it is possible to obtain reduced basis of Log-unit lattices which are not as orthogonal as over cyclotomic units, but are not reduced further by BKZ_{20}.

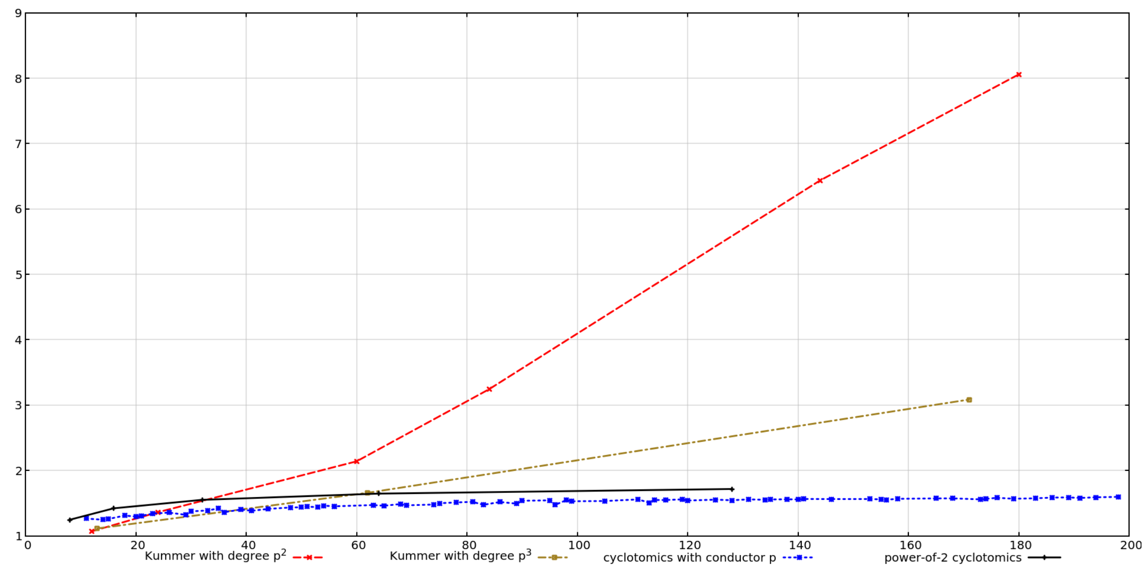


Figure 5.7: Values of δ plotted against $r_1 + r_2 - 1$ over Kummer fields of degrees p^2 and p^3, and over cyclotomic fields after LLL reductions

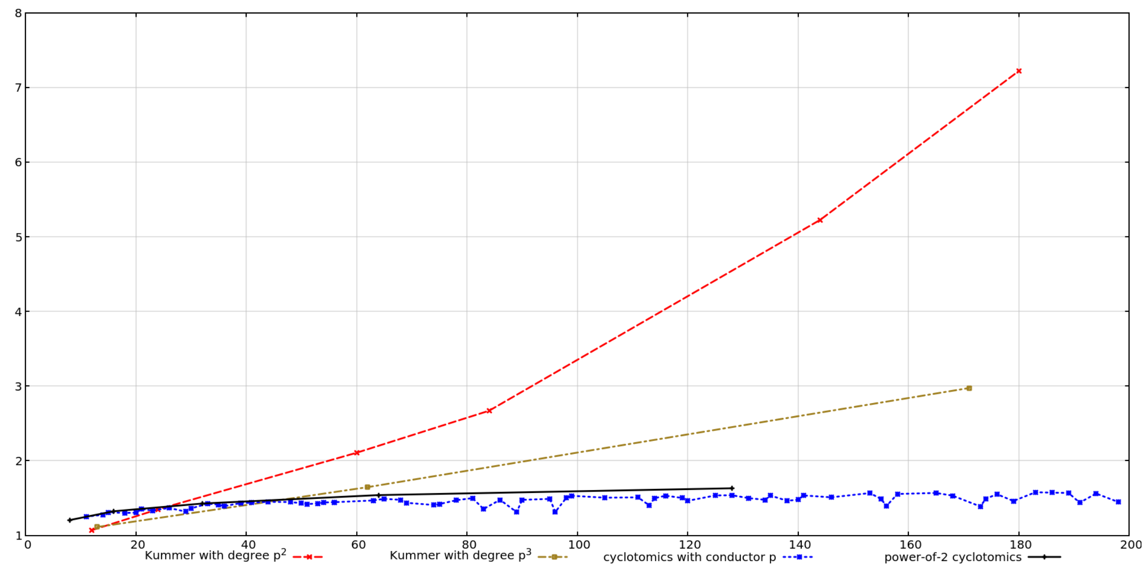


Figure 5.8: Values of δ plotted against $r_1 + r_2 - 1$ over Kummer fields of degrees p^2 and p^3, and over cyclotomic fields after BKZ reductions

Chapter 6

Conclusion

6.1 Diagonally dominant matrices

In Chapter 3 we explored the possibility of building an encryption based on diagonally dominant matrices, inspired by the signature scheme DRS [83]. To show that a correct scheme could be constructed, we studied λ_1 for c.d.d. and r.d.d. matrices and exhibited a lower bound. Moreover we showed that one could construct algorithms running in polynomial time and solving GDD_γ for γ depending on the noise matrix of the diagonally dominant matrix considered. From these, one can deduce an upper bound on the covering radius of diagonally dominant lattices.

Even if the cryptosystem DRE that we gave as an example of scheme using diagonally dominant matrices is correct, much work is left to be done. In particular, the security of such a cryptosystem needs to be assessed. Moreover, the algorithms that we described are not proven to be efficient in their current form. Therefore, a more in-depth study of these algorithms with implementation work have to be undertaken if one hopes to obtain acceptable enough efficiency.

6.2 Computations in number fields

We studied practical improvements for two tasks in Chapter 4.

We first studied two different methods to compute the norm of an ideal I relative to an extension of number fields L/K. The first one is certified to run in polynomial time when the Galois closure of the extension is small enough. It computes the norm as the product of all conjugates of the ideal I. Even if one can hasten the computations by checking at some points during the process if the norm has been reached, it is still not very efficient because of the size of the matrices handled. The second method that we studied is only heuristic and probabilistic. We were not able to find an upper bound on its running time. However, it behaves very well in practice and outperforms both our first method and the implementation of MAGMA [22].

The second task that we studied is the computation of polynomial roots in number fields. We described a method which can be linked to the paper originally describing the LLL algorithm [63]. It computes the roots of a polynomial through approximations of conjugates. We showed that this method runs in heuristic polynomial time. Moreover we described how to take advantage of the structure of an extension L/K in order to do a decoding phase with respect to K instead of L. Moreover we made several heuristic observations allowing us to speed-up both methods. Finally, experimental data shows that the absolute method can be competitive with PARI/GP [76] in some cases, and that the relative offers great speed-ups when the relative degree $[L : K]$ and the degree of the polynomial that we study are small.

Further improvements can be explored. Regarding the computation of ideal norms, one should find an upper bound on the running time of our probabilistic method. Then concerning the extraction of polynomial roots, one could try to improve the classical method implemented in PARI/GP. One could find a heuristic way of evaluating the volume needed to ensure the correctness of the decoding, as we did for the precision of our method. Moreover, finding a way of using the structure of an extension such as we described could improve massively the running time of this algorithm. Our methods could be more efficient by using Babaï's nearest plane algorithm to solve BDD instead of Kannan's as we described.

6.3 Real Kummer extensions

In Chapter 5 we studied some real Kummer extensions, namely number fields of the form $L = K(\sqrt[p]{m_1}, \ldots, \sqrt[p]{m_r})$ with $K = \mathbb{Q}$ or $K = \mathbb{Q}(\sqrt[q]{n_1}, \ldots, \sqrt[q]{n_s})$, where p, q are prime integers. Our goal was to assess the possibility of solving the SPIP through the Log-unit lattice. To this end, we generalised the work of Bauch et al. done over multiquadratic fields [6]. In particular we showed that general real Kummer fields enjoy similar properties to multiquadratic fields, which can be exploited to design efficient algorithms computing the unit group and solving the PIP over these fields. Our implementation of these algorithms allowed us to try to solve the SPIP in practice over some real Kummer extensions. Experimental data showed that the probability of success is very high over most of fields studied. However some fields seem to be more resistant, namely fields of the form $\mathbb{Q}(\sqrt[p]{m_1}, \sqrt[p]{m_2})$ with small m_1 and m_2 (especially $m_1 = 2$ and $m_2 = 3$). Moreover, we studied the geometry of the

Log-unit lattice of these real Kummer fields and compared it to the one of cyclotomic fields. We were able to highlight the fact the geometries are really different. In particular, the basis of $\mathrm{Log}(\mathcal{O}_K^\times)$ computed by our algorithms are far from orthogonal, which is completely different than for cyclotomic fields. Our work indicates that the structure of the Log-unit lattice can be different from one number field to another. In particular, the SPIP is more difficult to solve over some real Kummer fields than over cyclotomic fields. Consequently, such fields could be an alternative to cyclotomic fields when building cryptosystems based on structured lattices such as ideal lattices or module lattices. Another fact that our work highlights is the importance of studying large degree number fields. Indeed, we were able to detect that fields of the form $\mathbb{Q}(\sqrt[p]{2}, \sqrt[p]{3})$ had different properties than other Kummer fields only because we computed data for degree larger than 120.

The first direction that we could explore further would be to build and cryptanalyse a cryptosystem based on the hardness of solving the SPIP over Kummer fields. Then one could implement algorithms solving the ISVP [79, 9] over such fields, and compare the quality of the output compared to cyclotomic fields. Finally, the work of Biasse et al. [19] shows that other number fields enjoy a structure allowing us to compute the unit group and solve the PIP more efficiently than with generic algorithms. Implementing such algorithms could help in studying more number fields.

Bibliography

[1] https://csrc.nist.gov/Projects/post-quantum-cryptography/post-quantum-cryptography-standardization.

[2] M. Ajtai. “The Shortest Vector Problem in L_2 is NP-Hard for Randomized Reductions (Extended Abstract)”. In: *Proceedings of the Thirtieth Annual ACM Symposium on Theory of Computing*. STOC ’98. Dallas, Texas, USA: Association for Computing Machinery, 1998, pp. 10–19. ISBN: 0897919629. DOI: 10.1145/276698.276705. URL: https://doi.org/10.1145/276698.276705.

[3] M. Albrecht, S. Bai, and L. Ducas. “A Subfield Lattice Attack on Overstretched NTRU Assumptions”. In: *Advances in Cryptology – CRYPTO 2016*. Ed. by M. Robshaw and J. Katz. Berlin, Heidelberg: Springer Berlin Heidelberg, 2016, pp. 153–178. ISBN: 978-3-662-53018-4.

[4] E. Alkim et al. “Post-Quantum Key Exchange: A New Hope”. In: *Proceedings of the 25th USENIX Conference on Security Symposium*. SEC’16. Austin, TX, USA: USENIX Association, 2016, pp. 327–343. ISBN: 9781931971324.

[5] L. Babaï. “On Lovasz lattice reduction and the nearest lattice point problem”. In: *Combinatorica* 6 (Mar. 1986), pp. 1–13. DOI: 10.1007/BF02579403.

[6] J. Bauch et al. “Short Generators Without Quantum Computers: The Case of Multiquadratics”. In: *Advances in Cryptology – EUROCRYPT 2017*. Ed. by J.-S. Coron and J. B. Nielsen. Cham: Springer International Publishing, 2017, pp. 27–59. ISBN: 978-3-319-56620-7.

[7] K. Belabas. “A relative van Hoeij algorithm over number fields”. In: *J. Symb. Comput.* 37 (May 2004), pp. 641–668. DOI: 10.1016/j.jsc.2003.09.003.

[8] K. Belabas. “Topics in computational algebraic number theory”. en. In: *Journal de théorie des nombres de Bordeaux* 16.1 (2004), pp. 19–63. DOI: 10.5802/jtnb.433. URL: http://www.numdam.org/item/JTNB_2004__16_1_19_0.

[9] O. Bernard and A. Roux-Langlois. "Twisted-PHS: Using the Product Formula to Solve Approx-SVP in Ideal Lattices". In: *Advances in Cryptology – ASIACRYPT 2020*. Ed. by S. Moriai and H. Wang. Cham: Springer International Publishing, 2020, pp. 349–380. ISBN: 978-3-030-64834-3.

[10] D. J. Bernstein, J. Buchmann, and E. Dahmen. *Post Quantum Cryptography*. 1st. Springer Publishing Company, Incorporated, 2008. ISBN: 3540887016.

[11] D. J. Bernstein et al. "NTRU Prime: Reducing Attack Surface at Low Cost". In: *Selected Areas in Cryptography – SAC 2017*. Ed. by C. Adams and J. Camenisch. Cham: Springer International Publishing, 2018, pp. 235–260. ISBN: 978-3-319-72565-9.

[12] J.-F. Biasse and C. Fieker. "Improved techniques for computing the ideal class group and a system of fundamental units in number fields." In: *Algorithmic Number Theory, 10th International Symposium, ANTS-IX, San Diego CA, USA, July 9-13, 2012. Proceedings*. Vol. 1. Open Book Series. Mathematical Science Publishers, 2012, pp. 113–133.

[13] J.-F. Biasse and F. Song. "Efficient quantum algorithms for computing class groups and solving the principal ideal problem in arbitrary degree number fields". In: *Proceedings of the Twenty-Seventh Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2016, Arlington, VA, USA, January 10-12, 2016*. 2016, pp. 893–902. DOI: `10.1137/1.9781611974331.ch64`. URL: `http://dx.doi.org/10.1137/1.9781611974331.ch64`.

[14] J.-F. Biasse. *A fast algorithm for finding a short generator of a principal ideal of* $\mathbb{Q}(\zeta_{p^s})$. 2017. arXiv: `1503.03107 [math.NT]`.

[15] J.-F. Biasse and C. Fieker. "Subexponential class group and unit group computation in large degree number fields". In: *LMS Journal of Computation and Mathematics* 17.A (2014). DOI: `10.1112/S1461157014000345`.

[16] J.-F. Biasse, C. Fieker, and T. Hofmann. "On the computation of the HNF of a module over the ring of integers of a number field". In: *Journal of Symbolic Computation* 80 (2017), pp. 581–615. ISSN: 0747-7171. DOI: `https://doi.org/10.1016/j.jsc.2016.07.027`. URL: `https://www.sciencedirect.com/science/article/pii/S0747717116300736`.

[17] J.-F. Biasse and C. Vredendaal. "Fast multiquadratic S-unit computation and application to the calculation of class groups". In: *The Open Book Series* 2 (Jan. 2019), pp. 103–118. DOI: `10.2140/obs.2019.2.103`.

[18] J.-F. Biasse et al. "Computing Generator in Cyclotomic Integer Rings". In: *Advances in Cryptology – EUROCRYPT 2017*. Ed. by J.-S. Coron and J. B. Nielsen. Cham: Springer International Publishing, 2017, pp. 60–88. ISBN: 978-3-319-56620-7.

[19] J.-F. Biasse et al. *Norm relations and computational problems in number fields*. 2020. arXiv: `2002.12332 [math.NT]`.

[20] J. Bos et al. "CRYSTALS - Kyber: A CCA-Secure Module-Lattice-Based KEM". In: Apr. 2018, pp. 353–367. DOI: `10.1109/EuroSP.2018.00032`.

[21] J. Bos et al. "Frodo: Take off the Ring! Practical, Quantum-Secure Key Exchange from LWE". In: Oct. 2016, pp. 1006–1018. DOI: `10.1145/2976749.2978425`.

[22] W. Bosma, J. Cannon, and C. Playoust. "The Magma algebra system. I. The user language". In: *J. Symbolic Comput.* 24.3-4 (1997). Computational algebra and number theory (London, 1993), pp. 235–265. ISSN: 0747-7171. DOI: `10.1006/jsco.1996.0125`. URL: `http://dx.doi.org/10.1006/jsco.1996.0125`.

[23] Z. Brakerski, C. Gentry, and V. Vaikuntanathan. "(Leveled) Fully Homomorphic Encryption without Bootstrapping". In: *Proceedings of the 3rd Innovations in Theoretical Computer Science Conference*. ITCS '12. Cambridge, Massachusetts: Association for Computing Machinery, 2012, pp. 309–325. ISBN: 9781450311151. DOI: `10.1145/2090236.2090262`. URL: `https://doi.org/10.1145/2090236.2090262`.

[24] J. Buchmann. "A subexponential algorithm for the determination of class groups and regulators of algebraic number fields". In: 1990.

[25] J. P. Buhler, H. W. Lenstra, and C. Pomerance. "Factoring integers with the number field sieve". In: *The development of the number field sieve*. Ed. by A. K. Lenstra and H. W. Lenstra. Berlin, Heidelberg: Springer Berlin Heidelberg, 1993, pp. 50–94. ISBN: 978-3-540-47892-8.

[26] P. Campbell, M. Groves, and D. Shepherd. *Soliloquy : a cautionary tale*. 2014.

[27] A. P. Chalmeta. "On the Units and the Structure of the 3-Sylow Subgroups of the Ideal Class Groups of Pure Bicubic Fields and their Normal Closures". In: 2006.

[28] Y. Chen and P. Q. Nguyen. "BKZ 2.0: Better Lattice Security Estimates". In: *Advances in Cryptology – ASIACRYPT 2011*. Ed. by D. H. Lee and X. Wang. Berlin, Heidelberg: Springer Berlin Heidelberg, 2011, pp. 1–20. ISBN: 978-3-642-25385-0.

[29] J. Cheon, J. Jeong, and C. Lee. "An algorithm for NTRU problems and cryptanalysis of the GGH multilinear map without a low-level encoding of zero". In: *LMS Journal of Computation and Mathematics* 19 (Jan. 2016), pp. 255–266. DOI: 10.1112/S1461157016000371.

[30] H. Cohen. *Advanced Topics in Computational Number Theory.* Graduate Texts in Mathematics. Springer New York, 2012. ISBN: 9781441984890. URL: https://books.google.cz/books?id=OFjdBwAAQBAJ.

[31] H. Cohen. *A Course in Computational Algebraic Number Theory.* Berlin, Heidelberg: Springer-Verlag, 1993. ISBN: 0-387-55640-0.

[32] J. Conway and N. Sloane. *Sphere Packings, Lattices and Groups.* Vol. 290. Jan. 1988. ISBN: 978-1-4757-2018-1. DOI: 10.1007/978-1-4757-2016-7.

[33] R. Cramer, L. Ducas, and B. Wesolowski. "Short Stickelberger Class Relations and Application to Ideal-SVP". In: *EUROCRYPT.* 2017.

[34] R. Cramer et al. "Recovering Short Generators of Principal Ideals in Cyclotomic Rings". In: *Advances in Cryptology – EUROCRYPT 2016.* Ed. by M. Fischlin and J.-S. Coron. Berlin, Heidelberg: Springer Berlin Heidelberg, 2016, pp. 559–585. ISBN: 978-3-662-49896-5.

[35] J.-P. D'Anvers et al. "Saber: Module-LWR Based Key Exchange, CPA-Secure Encryption and CCA-Secure KEM". In: *Progress in Cryptology – AFRICACRYPT 2018.* Ed. by A. Joux, A. Nitaj, and T. Rachidi. Cham: Springer International Publishing, 2018, pp. 282–305. ISBN: 978-3-319-89339-6.

[36] W. Diffie and M. Hellman. "New directions in cryptography". In: *IEEE Transactions on Information Theory* 22.6 (1976), pp. 644–654. DOI: 10.1109/TIT.1976.1055638.

[37] L. Ducas. "Advances on quantum cryptanalysis of ideal lattices". In: 2017.

[38] K. Eisenträger et al. "A quantum algorithm for computing the unit group of an arbitrary degree number field". In: *Proceedings of the Annual ACM Symposium on Theory of Computing* (May 2014), pp. 293–302. DOI: 10.1145/2591796.2591860.

[39] P. van Emde Boas. *Another Np-complete Partition Problem and the Complexity of Computing Short Vectors in a Lattice.* Mathematical preprints series. Universiteit van Amsterdam. Mathematisch Instituut, 1981. URL: https://books.google.com.au/books?id=rRcgrgEACAAJ.

[40] C. Fieker and C. Friedrichs. "On Reconstruction of Algebraic Numbers". In: *Algorithmic Number Theory.* Ed. by W. Bosma. Berlin, Heidelberg: Springer Berlin Heidelberg, 2000, pp. 285–296. ISBN: 978-3-540-44994-2.

[41] U. Fincke and M. Pohst. “Improved Methods for Calculating Vectors of Short Length in a Lattice, Including a Complexity Analysis”. In: *Mathematics of Computation* 44.170 (1985), pp. 463–471. ISSN: 00255718, 10886842. URL: `http://www.jstor.org/stable/2007966`.

[42] F. Fontein, M. Schneider, and U. Wagner. *PotLLL: A Polynomial Time Version of LLL With Deep Insertions.* 2013. arXiv: `1307.7534 [cs.CR]`.

[43] P.-A. Fouque et al. `https://csrc.nist.gov/Projects/post-quantum-cryptography/post-quantum-cryptography-standardization`.

[44] J. N. Franklin. *Matrix theory.* Courier Corporation, 2012.

[45] C. Gentry. “A Fully Homomorphic Encryption Scheme”. AAI3382729. PhD thesis. Stanford, CA, USA: Stanford University, 2009. ISBN: 978-1-109-44450-6.

[46] C. Gentry and S. Halevi. “Implementing Gentry’s Fully-Homomorphic Encryption Scheme”. In: vol. 6632. May 2011, pp. 129–148. DOI: `10.1007/978-3-642-20465-4_9`.

[47] O. Goldreich, S. Goldwasser, and S. Halevi. “Public-key cryptosystems from lattice reduction problems”. In: *Advances in Cryptology—CRYPTO’97.* Springer, 1997, pp. 112–131.

[48] J. Hafner and K. McCurley. “A rigorous subexponential algorithm for computation of class groups”. In: *Journal of the American Mathematical Society* 2 (1989), pp. 837–850.

[49] G. Hanrot and D. Steh. “A COMPLETE WORST-CASE ANALYSIS OF KANNAN’S SHORTEST LATTICE VECTOR ALGORITHM”. In: (July 2012).

[50] B. Helfrich. “Algorithms to construct minkowski reduced and hermite reduced lattice bases”. In: *Theoretical Computer Science* 41 (1985), pp. 125–139. ISSN: 0304-3975. DOI: `https://doi.org/10.1016/0304-3975(85)90067-2`. URL: `https://www.sciencedirect.com/science/article/pii/0304397585900672`.

[51] C. Hermite. “Extraits de lettres de M. Ch. Hermite à M. Jacobi sur différents objects de la théorie des nombres.” In: *Journal für die reine und angewandte Mathematik (Crelles Journal)* 1850 (), pp. 261–278.

[52] R. Hiromasa. “Digital Signatures from the Middle-Product LWE”. In: *Provable Security.* Ed. by J. Baek, W. Susilo, and J. Kim. Cham: Springer International Publishing, 2018, pp. 239–257. ISBN: 978-3-030-01446-9.

[53] J. Hoffstein et al. "Choosing Parameters for NTRUEncrypt". In: *Topics in Cryptology – CT-RSA 2017*. Ed. by H. Handschuh. Cham: Springer International Publishing, 2017, pp. 3–18. ISBN: 978-3-319-52153-4.

[54] J. Hoffstein, J. Pipher, and J. H. Silverman. "NTRU: A ring-based public key cryptosystem". In: *Algorithmic number theory*. Springer, 1998, pp. 267–288.

[55] N. L. Johnson, A. W. Kemp, and S. Kotz. *Univariate discrete distributions*. Vol. 444. John Wiley & Sons, 2005.

[56] R. Kannan. "Improved Algorithms for Integer Programming and Related Lattice Problems". In: *Proceedings of the Fifteenth Annual ACM Symposium on Theory of Computing*. STOC '83. New York, NY, USA: Association for Computing Machinery, 1983, pp. 193–206. ISBN: 0897910990. DOI: `10.1145/800061.808749`. URL: `https://doi.org/10.1145/800061.808749`.

[57] R. Kannan. "Minkowski's Convex Body Theorem and Integer Programming". In: *Mathematics of Operations Research* 12.3 (1987), pp. 415–440. ISSN: 0364765X, 15265471. URL: `http://www.jstor.org/stable/3689974`.

[58] R. Kannan and A. Bachem. "Polynomial Algorithms for Computing the Smith and Hermite Normal Forms of an Integer Matrix". In: *SIAM J. Comput.* 8 (Nov. 1979), pp. 499–507. DOI: `10.1137/0208040`.

[59] P. Kirchner, T. Espitau, and P.-A. Fouque. "Fast Reduction of Algebraic Lattices over Cyclotomic Fields". In: *Advances in Cryptology – CRYPTO 2020*. Ed. by D. Micciancio and T. Ristenpart. Cham: Springer International Publishing, 2020, pp. 155–185. ISBN: 978-3-030-56880-1.

[60] A. Korkine and G. Zolotareff. "Sur les formes quadratiques". In: *Mathematische Annalen* 6 (Jan. 1970), pp. 366–389. DOI: `10.1007/BF01442795`.

[61] A. Langlois and D. Stehlé. "Worst-Case to Average-Case Reductions for Module Lattices". In: *Designs, Codes and Cryptography* 75 (June 2014). DOI: `10.1007/s10623-014-9938-4`.

[62] C. Lee et al. "An LLL Algorithm for Module Lattices". In: Nov. 2019, pp. 59–90. ISBN: 978-3-030-34620-1. DOI: `10.1007/978-3-030-34621-8_3`.

[63] A. Lenstra, H. Lenstra, and L. Lovász. "Factoring Polynomials with Rational Coefficients". In: *Mathematische Annalen* 261 (Dec. 1982). DOI: `10.1007/BF01457454`.

[64] A. Lesavourey, T. Plantard, and W. Susilo. "Short Principal Ideal Problem in multicubic fields". In: *Journal of Mathematical Cryptology* 14.1 (1Jan. 2020), pp. 359–392. DOI: `https://doi.org/10.1515/jmc-2019-0028`. URL: `https://www.degruyter.com/view/journals/jmc/14/1/article-p359.xml`.

[65] R. Liu and Y. Pan. "Computing Hermite Normal Form Faster via Solving System of Linear Equations". In: *Proceedings of the 2019 on International Symposium on Symbolic and Algebraic Computation*. ISSAC '19. Beijing, China: Association for Computing Machinery, 2019, pp. 283–290. ISBN: 9781450360845. DOI: `10.1145/3326229.3326238`. URL: `https://doi.org/10.1145/3326229.3326238`.

[66] V. Lyubashevsky, C. Peikert, and O. Regev. "On Ideal Lattices and Learning with Errors over Rings". In: *Advances in Cryptology – EUROCRYPT 2010*. Ed. by H. Gilbert. Berlin, Heidelberg: Springer Berlin Heidelberg, 2010, pp. 1–23. ISBN: 978-3-642-13190-5.

[67] R. Merkle and M. Hellman. "Hiding information and signatures in trapdoor knapsacks". In: *IEEE Transactions on Information Theory* 24.5 (1978), pp. 525–530. DOI: `10.1109/TIT.1978.1055927`.

[68] D. Micciancio. "Improving Lattice Based Cryptosystems Using the Hermite Normal Form". In: *LNCS* 2146 (Nov. 2001). DOI: `10.1007/3-540-44670-2_11`.

[69] D. Micciancio and S. Goldwasser. *Complexity of Lattice Problems*. USA: Kluwer Academic Publishers, 2002. ISBN: 0792376889.

[70] H. Minkowski. *Geometrie der Zahlen*. fre. Leipzig, 1910.

[71] T. Mukherjee and N. Stephens-Davidowitz. "Lattice Reduction for Modules, or How to Reduce ModuleSVP to ModuleSVP". In: *Advances in Cryptology – CRYPTO 2020*. Ed. by D. Micciancio and T. Ristenpart. Cham: Springer International Publishing, 2020, pp. 213–242. ISBN: 978-3-030-56880-1.

[72] J. Neukirch. "Algebraic Number Theory". In: 1999.

[73] P. Q. Nguên and D. Stehlé. "Floating-Point LLL Revisited". In: *Advances in Cryptology – EUROCRYPT 2005*. Ed. by R. Cramer. Berlin, Heidelberg: Springer Berlin Heidelberg, 2005, pp. 215–233. ISBN: 978-3-540-32055-5.

[74] P. Q. Nguyen and D. Stehlé. "An LLL Algorithm with Quadratic Complexity". In: *SIAM J. Comput.* 39 (2009), pp. 874–903.

[75] Y. Pan et al. *On the Ideal Shortest Vector Problem over Random Rational Primes*. Cryptology ePrint Archive, Report 2021/245. `https://eprint.iacr.org/2021/245`. 2021.

[76] *PARI/GP version* `2.11.2`. available from `http://pari.math.u-bordeaux.fr/`. Univ. Bordeaux: The PARI Group, 2019.

[77] C. Peikert. “A Decade of Lattice Cryptography”. In: *Found. Trends Theor. Comput. Sci.* 10.4 (Mar. 2016), pp. 283–424. ISSN: 1551-305X. DOI: `10.1561/0400000074`. URL: `https://doi.org/10.1561/0400000074`.

[78] C. Peikert. “Public-Key Cryptosystems from the Worst-Case Shortest Vector Problem: Extended Abstract”. In: *Proceedings of the Forty-First Annual ACM Symposium on Theory of Computing.* STOC ’09. Bethesda, MD, USA: Association for Computing Machinery, 2009, pp. 333–342. ISBN: 9781605585062. DOI: `10.1145/1536414.1536461`. URL: `https://doi.org/10.1145/1536414.1536461`.

[79] A. Pellet-Mary, G. Hanrot, and D. Stehlé. “Approx-SVP in Ideal Lattices with Pre-processing”. In: *Advances in Cryptology – EUROCRYPT 2019.* Ed. by Y. Ishai and V. Rijmen. Cham: Springer International Publishing, 2019, pp. 685–716. ISBN: 978-3-030-17656-3.

[80] C. Pernet and W. Stein. “Fast computation of Hermite normal forms of random integer matrices”. In: *Journal of Number Theory* 130.7 (2010), pp. 1675–1683. ISSN: 0022-314X. DOI: `https://doi.org/10.1016/j.jnt.2010.01.017`. URL: `http://www.sciencedirect.com/science/article/pii/S0022314X10000727`.

[81] T. Plantard, W. Susilo, and K. T. Win. “A Digital Signature Scheme Based on CVP_∞”. In: *Public Key Cryptography – PKC 2008.* Ed. by R. Cramer. Berlin, Heidelberg: Springer Berlin Heidelberg, 2008, pp. 288–307. ISBN: 978-3-540-78440-1.

[82] T. Plantard, W. Susilo, and Z. Zhang. “LLL for ideal lattices: re-evaluation of the security of Gentry–Halevi’s FHE scheme”. In: *Designs, Codes and Cryptography* 76 (Mar. 2014). DOI: `10.1007/s10623-014-9957-1`.

[83] T. PLANTARD et al. `https://documents.uow.edu.au/~thomaspl/drs/2017/specification.pdf`.

[84] M. Pohst. “A Modification of the LLL Reduction Algorithm.” In: *Journal of Symbolic Computation* 4 (Aug. 1987), pp. 123–127. DOI: `10.1016/S0747-7171(87)80061-5`.

[85] M. Pohst. “On the Computation of Lattice Vectors of Minimal Length, Successive Minima and Reduced Bases with Applications”. In: *SIGSAM Bull.* 15.1 (Feb. 1981), pp. 37–44. ISSN: 0163-5824. DOI: `10.1145/1089242.1089247`. URL: `https://doi.org/10.1145/1089242.1089247`.

[86] O. Regev. “On Lattices, Learning with Errors, Random Linear Codes, and Cryptography”. In: *J. ACM* 56.6 (Sept. 2009). ISSN: 0004-5411. DOI: 10.1145/1568318.1568324. URL: https://doi.org/10.1145/1568318.1568324.

[87] P. Ribenboim. *Classical theory of algebraic numbers*. Springer Science & Business Media, 2013.

[88] R. L. Rivest, A. Shamir, and L. Adleman. “A Method for Obtaining Digital Signatures and Public-Key Cryptosystems”. In: *Commun. ACM* 21.2 (Feb. 1978), pp. 120–126. ISSN: 0001-0782. DOI: 10.1145/359340.359342. URL: https://doi.org/10.1145/359340.359342.

[89] M. Roşca et al. “Middle-Product Learning with Errors”. In: *Advances in Cryptology – CRYPTO 2017*. Ed. by J. Katz and H. Shacham. Cham: Springer International Publishing, 2017, pp. 283–297. ISBN: 978-3-319-63697-9.

[90] P. Samuel. *Algebraic theory of numbers*. Hermann, 1970.

[91] B. Schmal. “Diskriminanten,$\mathbb{Z}$-Ganzheitsbasen und relative Ganzheitsbasen bei multiquadratischen Zahlkörpern”. In: *Archiv der Mathematik* 52 (1989), pp. 245–257.

[92] C. P. Schnorr and M. Euchner. “Lattice Basis Reduction: Improved Practical Algorithms and Solving Subset Sum Problems”. In: *Math. Program.* 66.2 (Sept. 1994), pp. 181–199. ISSN: 0025-5610. DOI: 10.1007/BF01581144. URL: https://doi.org/10.1007/BF01581144.

[93] C. P. Schnorr and H. H. Hörner. “Attacking the Chor-Rivest Cryptosystem by Improved Lattice Reduction”. In: *Advances in Cryptology — EUROCRYPT ’95*. Ed. by L. C. Guillou and J.-J. Quisquater. Berlin, Heidelberg: Springer Berlin Heidelberg, 1995, pp. 1–12. ISBN: 978-3-540-49264-1.

[94] C. Schnorr. “A hierarchy of polynomial time lattice basis reduction algorithms”. In: *Theoretical Computer Science* 53.2 (1987), pp. 201–224. ISSN: 0304-3975. DOI: https://doi.org/10.1016/0304-3975(87)90064-8. URL: http://www.sciencedirect.com/science/article/pii/0304397587900648.

[95] P. Shor. “Algorithms for Quantum Computation: Discrete Logarithms and Factoring”. In: *Proceedings of the 35th Annual IEEE Symposium on Foundations of Computer Science* (1994).

[96] A. Sipasseuth, T. Plantard, and W. Susilo. “Improving the Security of the DRS Scheme with Uniformly Chosen Random Noise”. In: *Information Security and Privacy*. Ed. by J. Jang-Jaccard and F. Guo. Cham: Springer International Publishing, 2019, pp. 119–137. ISBN: 978-3-030-21548-4.

[97] N. Smart and F. Vercauteren. “Fully homomorphic encryption with relatively small key and ciphertext sizes”. In: *Fully Homomorphic Encryption with Relatively Small Key and Ciphertext Sizes* (Jan. 2010), pp. 420–443.

[98] D. Stehlé et al. “Efficient Public Key Encryption Based on Ideal Lattices”. In: *Advances in Cryptology – ASIACRYPT 2009*. Ed. by M. Matsui. Berlin, Heidelberg: Springer Berlin Heidelberg, 2009, pp. 617–635. ISBN: 978-3-642-10366-7.

[99] B. M. Trager. “Algebraic Factoring and Rational Function Integration”. In: *Proceedings of the Third ACM Symposium on Symbolic and Algebraic Computation*. SYMSAC ’76. Yorktown Heights, New York, USA: Association for Computing Machinery, 1976, pp. 219–226. ISBN: 9781450377904. DOI: `10.1145/800205.806338`. URL: `https://doi.org/10.1145/800205.806338`.

[100] J. Westlund. “On the Fundamental Number of the Algebraic Number-Field $k(\sqrt[p]{m})$”. In: *Transactions of the American Mathematical Society* 11.4 (1910), pp. 388–392.

[101] Y. Yu and L. Ducas. “Learning Strikes Again: The Case of the DRS Signature Scheme”. In: *Advances in Cryptology – ASIACRYPT 2018*. Ed. by T. Peyrin and S. Galbraith. Cham: Springer International Publishing, 2018, pp. 525–543. ISBN: 978-3-030-03329-3.

Appendix A

Alternative structures for DRE

As we mentioned earlier, there are no better general results that the ones we already provided as the bounds are reached in practice. However additional structures could reach better bounds. We will explore some possibilities and their influence on the length of the shortest vector and the covering radius.

A.1 All positive, all negative

This subsection considers the case where every $m_{i,j}$ is positive or negative.

Negative case

The negative case offers properties that are not necessarily useful by themselves, but could help in the creation of novel structures for cryptography or in the general understanding of diagonal dominant matrices.

Lemma A.1 (Shortest vector of the negative case)**.** *Let B be a c.d.d. matrix where $b_{i,j} \leqslant 0$ for all $i \neq j$. Then $v = \sum_{i=1}^{n} B_i$ is a shortest non-zero vector of $\mathcal{L}(B)$.*

Proof. $v_i = D - CN(B, i)$, thus reaching the minimal bound for shortest non-zero vectors in every position. □

The advantage of this lemma is to be able to use our worst-case assumption as the general case, however as far as we are concerned we do not see a practical usage for it.

Positive case

The positive case gives an interesting intuition for reduction algorithms: they give a very attractive graphical intuition as every vector operation moves every coefficient in the same “direction” (go up or down), i.e the vector’s coefficient interval range is guaranteed to be shrinking in each iteration until convergence.

As far as the length of the shortest vector is concerned, there is no guarantee it will be higher than the minimal bound. In fact, the example below shows we can reach the general bound:

Example. The matrix

$$\begin{bmatrix} D & D-1 & 0 & 0 & 0 & 0 \\ 0 & D & D-1 & 0 & 0 & 0 \\ 0 & 0 & D & D-1 & 0 & 0 \\ 0 & 0 & 0 & D & D-1 & 0 \\ 0 & 0 & 0 & 0 & D & D-1 \\ D-1 & 0 & 0 & 0 & 0 & D \end{bmatrix}$$

generates the vector $[1, -1, 1, -1, 1, -1]$

Some constructions with bounded noise coefficients and specific distributions can force limitations on how small the shortest vector can be, however those are very specific cases and it is unclear if we should expand on it in this paper.

A.2 Polarity-circular blocks

This section deals with matrices that have specific distribution on positive and negative noise coefficients.

2×2 blocks

Here we consider the case where the noise matrix M takes the following form:

$$\begin{bmatrix} 0 & A \\ B & 0 \end{bmatrix}$$

where every coefficient of A is strictly positive and B strictly negative. (A and B can be reversed and are square). In that case, $D > CN(M) \geqslant n/2$ and the shortest vector is large. In dimension 2, it is clear that the shortest vector is a vector of the basis. In larger dimension, it is not that simple.

Lemma A.2 (Shortest vector of 2×2 sign-blocks)**.** *Let $B \in \mathbb{Z}^{n \times n}$ be c.d.d and as described above. Then $\lambda_1(\mathcal{L}(B)) \geqslant D$.*

3×3 blocks

Now consider the case where the noise matrix M takes the following form:

$$\begin{bmatrix} 0 & A_{12} & B_{13} \\ B_{21} & 0 & A_{23} \\ A_{31} & B_{32} & 0 \end{bmatrix}$$

where every coefficient of A_{ij} is strictly negative and B_{ij} strictly positive (signs of A_{ij} and B_{ij} can be reversed and are square). We assume further the following:

$$\forall j \in [\![1,n]\!], \sum_{i=1}^{n} m_{i,j} = 0.$$

Let us fix some notation. We will write:

- $I = [\![1,n]\!]$;
- $I_k = [\![\frac{(k-1)n}{3}+1, \frac{kn}{3}]\!]$ for $k \in \{1,2,3\}$.

Lemma A.3. *Let $M = [m_{i,j}]_{\substack{i\in[\![1,n]\!]\\ j\in[\![1,n]\!]}} \in \mathrm{M}_n(\mathbb{Z})$ a c.d.d. matrix with a structure such as defined above and $n \in 3\mathbb{N}$, and three different values $k_1, k_2, k_3 \in \{1,2,3\}$. Consider $l \in \{-1,0,1\}^n \setminus \{0\}$ such that $l_i \geqslant 0$ for all $i \in I_{k_1}$ or $l_i \leqslant 0$ for all $i \in I_{k_1}$. Then the following statements are true.*

(i) $(\forall i \in I_{k_1} \cup I_{k_2}, l_i = 0) \implies \|lM\|_\infty \geqslant D\,\|l\|_\infty$; *(same for $I_{k_1} \cup I_{k_3}$ and $I_{k_2} \cup I_{k_3}$).*

(ii) $\exists k \in \{k_2,k_3\} \mid \forall j \in I_k, l_j = 0 \implies \|lM\|_\infty \geqslant D.$

(iii) $\forall k \in \{k_2,k_3\}, \exists i_k \in I_k \mid l_{i_k} \neq 0 \implies \|lM\|_\infty \geqslant D - \frac{CN(M)}{2} + 1.$

Proof. Without any loss of generality, we can assume that $l_i \geqslant 0$ for all $i \in I_1$ and $m_{i_j} > 0$ for all $(i,j) \in I_2 \times I_1$. The sign matrix of M is as follows:

$$\begin{bmatrix} 0 & - & + \\ + & 0 & - \\ - & + & 0 \end{bmatrix}.$$

The first statement is clear. Now let us prove statement 2.. It corresponds to proving Lemma A.2. Without loss of generality assume $l_j = 0$ for all $j \in I_3$ (i.e $k = 3$). If there is $j \in I_2$ such that $l_j < 0$, then since $l_i \geqslant 0$ and $m_{i,j} \leqslant 0$ for all $i \in I_1$, we have

$$(lM)_j = -|l_j|D - \sum_{i=1}^{n/3} l_i|m_{i,j}| \leqslant -D-1,$$

thus $\|lM\|_\infty > D$. If $l_j \geqslant 0$ for all $j \in I_2$ then $\|(lM)_i\|_\infty \geqslant D$ for all $i \in I_1$.
Let us now prove (iii). Following the same reasoning as before, one can see that if $l_{i_2} < 0$ then

$$(lM)_{i_2} = -|l_j|D - \sum_{i\in I_1} l_i|m_{i,j}| + \sum_{i\in I_3} l_i m_{i,j} \leqslant -D - 1 + \sum_{i\in I_3} l_i m_{i,j} < 0$$

thus $|(lM)_{i_2}| \geqslant D + 1 - \frac{CN(M)}{2}$. Similarly if $l_{i_3} > 0$ then $|(lM)_{i_3}| \geqslant D + 1 - \frac{CN(M)}{2}$.
Finally if $l_i \geqslant 0$ for all $i \in I_2$ and $l_i \leqslant 0$ for all $i \in I_3$ then $\|lM\|_\infty > D$ and (iii) is true.
Since all of the above can be adapted to the cases where $l_i \leqslant 0$ for all $i \in I_1$, or where we replace I_1 by I_2 or I_3 we proved that if there is $k \in \{1,2,3\}$ such that all of the coefficients l_i with $i \in I_k$ have the same sign, then $\|lM\|_\infty > D - \frac{CN(M)}{2}$. □

Lemma A.4. *Let $M \in \mathrm{M}_n(\mathbb{Z})$ with a structure such as defined above and $n \in 3\mathbb{N}$. Then for $l \in \{-1,0,1\}^n$, $v = lM$ has $\|v\|_\infty \geq \min\{D - \frac{CN(M)}{2}, D - CN(M) + \frac{n}{3} + 2\}$.*

Proof. The previous lemma dealt with the case where $\exists k \in \{1,2,3\}$ such that $\forall i \in I_k, l_i \geqslant 0$ or $\forall i \in I_k, l_i \leqslant 0$. Now assume the following:

$$\forall k \in \{1,2,3\}, \exists (i_k, j_k) \in I_k^2, (l_{i_k} > 0) \wedge (l_{j_k} < 0).$$

Remark that it implies $n \geqslant 6$. With no loss of generality, let us fix $k = 1$ and define

$$A = \{i \in I_2 \mid l_i \geqslant 0\} \text{ and } B = \{i \in I_3 \mid l_i \leqslant 0\}$$

First assume that $|A| \geqslant \frac{n}{6}$ and $|B| \geqslant \frac{n}{6}$. Then we have

$$\begin{aligned}(lM)_{i_1} &= D + \sum_{i\in A\cup B} |l_i m_{i,i_1}| - \sum_{i\in I_2\cup I_3\setminus A\cup B} |l_i m_{i,i_1}| \\ &\geqslant D + 2 - 2(\frac{CN(M)}{2} - \frac{n}{6}) \\ &\geqslant D - CN(M) + \frac{n}{3} + 2.\end{aligned}$$

Now assume $|A| \leqslant \frac{n}{6}$ and $|B| \leqslant \frac{n}{6}$. Then similarly as before we obtain

$$(lM)_{j_1} = -D + \sum_{i\in A\cup B} |l_i m_{i,j_1}| - \sum_{i\in I_2\cup I_3\setminus A\cup B} |l_i m_{i,j_1}| \leqslant -D + CN(M) - \frac{n}{3} - 2.$$

Finally, assume $|A| \geqslant \frac{n}{6}$ and $|B| \leqslant \frac{n}{6}$. This means that $|I_3 \setminus B| \geqslant \frac{n}{6}$ so we obtain

$$\begin{aligned}(lM)_{j_1} &= -D + \sum_{i \in A \cup B} |l_i m_{i,j_1}| - \sum_{i \in I_2 \cup I_3 \setminus A \cup B} |l_i m_{i,j_1}| \\ &\leqslant -D - 1 - \frac{n}{6} + (\frac{CN(M)}{2} - 1) + (\frac{CN(M)}{2} - \frac{n}{6}) \\ &\leqslant -D + CN(M) - \frac{n}{3} - 2.\end{aligned}$$

The case $\#A \geqslant \frac{n}{6}$ and $\#B < \frac{n}{6}$ follows a similar reasoning. □

Finally, using Theorem 3.2 one can deduce from the results over $l \in \mathbb{Z}^n$ with $\|l\|_\infty = 1$ a lower bound for λ_1.

Corollary A.1. *Consider M a c.d.d. matrix by blocks as described above. Then it verifies $\lambda_1^{(\infty)}(\mathcal{L}(B)) \geqslant \min\{D - CN(M) + \frac{n}{3} + 2, D - \frac{CN(M)}{2}\}$.*

Note that those bounds are reached in the very worst case, and we present below an example that was built to reach the bound.

Example. Set $D = 19, CN(M) = 18, n = 6$. This gives $\lambda_1^{(\infty)} \geqslant 5$. Consider the matrix

$$M = \begin{bmatrix} D & 0 & -1 & 1-\frac{\beta}{2} & 1 & \frac{\beta}{2}-1 \\ 0 & D & 1-\frac{\beta}{2} & -1 & \frac{\beta}{2}-1 & 1 \\ \frac{\beta}{2}-1 & 1 & D & 0 & -1 & 1-\frac{\beta}{2} \\ 1 & \frac{\beta}{2}-1 & 0 & D & 1-\frac{\beta}{2} & -1 \\ 1-\frac{\beta}{2} & -1 & \frac{\beta}{2}-1 & 1 & D & 0 \\ -1 & 1-\frac{\beta}{2} & 1 & \frac{\beta}{2}-1 & 0 & D \end{bmatrix} = \begin{bmatrix} 19 & 0 & -1 & -8 & 1 & 8 \\ 0 & 19 & -8 & -1 & 8 & 1 \\ 8 & 1 & 19 & 0 & -1 & -8 \\ 1 & 8 & 0 & 19 & -8 & -1 \\ -8 & -1 & 8 & 1 & 19 & 0 \\ -1 & -8 & 1 & 8 & 0 & 19 \end{bmatrix}$$

and $l = \begin{bmatrix} -1 & 1 & 1 & -1 & -1 & 1 \end{bmatrix}$. This gives $v = lM = \begin{bmatrix} -5 & 5 & 5 & -5 & -5 & 5 \end{bmatrix}$ which has a norm of 5.

Note that unlike the example above, for large dimensions (and large diagonal value D) it is very unlikely that the maximum noise with absolute value $(\frac{CN(M)}{2} - \frac{n}{3} + 1)$ is picked for uniform distributions. Bounding the maximum noise coefficient will further increase the minimum possible length of the shortest vector.

Appendix B

Proofs of some result on dihedral groups

Here we consider a prime p, t a generator of the multiplicative group $\mathbb{F}_p^*$ and the semi-direct product $G \cong \langle \tau, \sigma \mid \tau^{p-1} = \sigma^p = 1, \tau\sigma\tau^{-1} = \sigma^t \rangle$. Recall that for any $u \in \langle \sigma \rangle$ and any $a \in [\![0, p-1]\!]$ one has $\tau^a u \tau^{-a} = u^{t^a}$ so any element of G can be written in the form $\tau^a\sigma^b$ or $\sigma^c\tau^d$ for some a, b, c, d. Remark further that if $g = \prod_i \tau^{a_i}\sigma^{b_i} \in G$ then the corresponding a and d are equal to $\sum_i a_i$.

Lemma B.1. *The subgroups of G are of the form $\langle \tau^a, \sigma \rangle$ with $a \in [\![0, p-2]\!]$ or of the form $\langle \tau^a\sigma^b \rangle$ with $a \in [\![1, p-2]\!]$ and $b \in [\![0, p-1]\!]$*

Proof. Consider a subgroup $H = \langle g_1, \ldots, g_r \rangle = \langle \tau^{a_1}\sigma^{b_1}, \ldots, \tau^{a_r}\sigma^{b_r} \rangle$ with $(a_i, b_i) \in [\![0, p-2]\!] \times [\![0, p-1]\!]$. First assume $\tau \in H$. Then one can write $H = \langle \tau, \sigma^{b_1}, \ldots, \sigma^{b_r} \rangle$ i.e. H is either $\langle \tau \rangle$ or $\langle \tau, \sigma \rangle$. Now assume $\sigma \in H$ instead. Then $H = \langle \sigma, \tau^{a_1}, \ldots, \sigma^{a_r} \rangle$ and there is $d \in [\![0, p-2]\!]$ such that H is $\langle \tau^d, \sigma \rangle$. Finally assume that neither τ nor σ belongs to H. One can see that for $i \neq j$ two integers in $[\![1, r]\!]$

$$(a_i = a_j) \wedge (b_i \neq b_j) \implies \exists b \neq 0 \mid \sigma^b \in H \implies \sigma \in H$$

from which we deduce

$$\forall (i, j) \in [\![1, r]\!], i \neq j \implies a_i \neq a_j.$$

Let $d = \gcd(a_1, \ldots, a_r)$. Using Bézout's identity one can see that there is $b \in [\![0, p-1]\!]$ such that $\tau^d\sigma^b$ is an element of H. Let us show that H is in fact equal to $\langle \tau^d\sigma^b \rangle$. Consider $i \in [\![1, r]\!]$ and write $h_i = (\tau^d\sigma^b)^{\frac{a_i}{d}}$. There is $c_i \in [\![0, p-1]\!]$ such that $h_i = \tau^{a_i}\sigma^{c_i}$. Following a previous reasoning we conclude that $h_i = g_i$. This is true for all $i \in [\![1, r]\!]$ so $H = \langle \tau^d\sigma^b \rangle$. $\square$

Lemma B.2. *The subgroups of G of the form $\langle \tau^a\sigma^b \rangle$ with $(a, b) \in [\![1, p-2]\!] \times \{0, 1\}$ have order $o(\tau^a) = o(t^a)$.*

Proof. Given an integer k one has $(\tau^a\sigma^b)^k = \sigma^e\tau^{ak}$ with

$$e = bt^a + bt^{2a} + \cdots + bt^{ka} = bt^a\frac{1-t^{ka}}{1-t^a}.$$

thus

$$\sigma^e\tau^a = 1 \iff (ak \equiv 0 \bmod (p-1)) \wedge (e = bt^a\frac{1-t^{ka}}{1-t^a} \equiv 0 \bmod p).$$

Then remark that one has also

$$ak \equiv 0 \bmod (p-1) \implies t^{ak} = 1 \bmod p \implies e \equiv 0 \bmod p.$$

□

Lemma B.3. *The subgroups of G with order $p-1$ are the p groups of the form $\langle\tau\sigma^b\rangle$ with $b \in [\![0, p-1]\!]$.*

Proof. A subgroup of G of order $p-1$ does not contain σ so it is necessarily of the form $\langle\tau^a\sigma^b\rangle$. Since $o(\tau^a\sigma^b) = o(\tau^a)$ one has

$$o(\tau^a\sigma^b) = p-1 \implies \langle\tau^a\rangle = \langle\tau\rangle$$

therefore there is $c \in [\![0, p-1]\!]$ such that $\tau\sigma^c \in \langle\tau^a\sigma^b\rangle$. □