An LLL Algorithm for Module Lattices*

Changmin Lee[1], Alice Pellet-Mary[2], Damien Stehlé[1,3], and Alexandre Wallet[4]

[1] Univ. Lyon, EnsL, UCBL, CNRS, Inria, LIP, F-69342 Lyon Cedex 07, France
{changmin.lee, damien.stehle}@ens-lyon.fr
[2] imec-COSIC, ESAT, KU Leuven, Belgium
alice.pelletmary@kuleuven.be
[3] Institut Universitaire de France
[4] NTT Secure Platform Laboratories, Tokyo, Japan
alexandre.wallet.th@hco.ntt.co.jp

Abstract. The LLL algorithm takes as input a basis of a Euclidean lattice, and, within a polynomial number of operations, it outputs another basis of the same lattice but consisting of rather short vectors. We provide a generalization to R-modules contained in K^n for arbitrary number fields K and dimension n, with R denoting the ring of integers of K. Concretely, we introduce an algorithm that efficiently finds short vectors in rank-n modules when given access to an oracle that finds short vectors in rank-2 modules, and an algorithm that efficiently finds short vectors in rank-2 modules given access to a Closest Vector Problem oracle for a lattice that depends only on K. The second algorithm relies on quantum computations and its analysis is heuristic. In the special case of free modules, we propose a dequantized version of this algorithm.

Keywords: LLL algorithm, Module lattices, Ideal lattices, Quantum cryptanalysis

1 Introduction

The NTRU [23], RingSIS [32,40], RingLWE [33,46], ModuleSIS and ModuleLWE [10, 28] problems and their variants serve as security foundations of numerous cryptographic protocols. Their main advantages are their presumed quantum hardness, their flexibility for realizing advanced cryptographic functionalities, and their efficiency compared to their SIS and LWE counterparts [1, 42]. As an illustration of their popularity for cryptographic design, we note that 11 out of the 26 candidates at Round 2 of the NIST standardization process for post-quantum cryptography rely on these problems or variants thereof.[5] From a hardness perspective, these problems are best viewed as standard problems on Euclidean lattices, restricted to random lattices corresponding to modules over the rings of integers of number fields. Further, for some parametrizations, there exist reductions from and to standard worst-case problems for such module lattices [3, 28, 43].

Let K be a number field and R its ring of integers. In this introduction, we will use the power-of-2 cyclotomic fields $K = \mathbb{Q}[x]/(x^d+1)$ and their rings of integers $R = \mathbb{Z}[x]/(x^d + 1)$ as a running example (with d a power of 2). An R-module $M \subset K^n$ is a finitely generated subset of vectors in K^n that is stable under addition and multiplication by elements of R. As an example, if we consider $h \in R/qR$ for some integer q, the set $\{(f, g)^T \in R^2 : fh = g \bmod q\}$ is a module. If h is an NTRU public

* This paper is an extended version of [29], which appeared in the *Proceedings of ASIACRYPT 2019*, Lecture Notes in Computer Science 11922, Springer-Verlag.

[5] See https://csrc.nist.gov/projects/post-quantum-cryptography

key, the corresponding secret key is a vector in that module, and its coefficients are small. Note that for $K = \mathbb{Q}$ and $R = \mathbb{Z}$, we recover Euclidean lattices in $\mathbb{Q}^n$. A first difficulty for handling modules compared to lattices is that R may not be a Euclidean domain, and, as a result, a module M may not be of the form $M = \sum_i R\mathbf{b}_i$ for some linearly independent $\mathbf{b}_i$'s in M. However, as R is a Dedekind domain, for every module M, there exist K-linearly independent $\mathbf{b}_i$'s and fractional ideals I_i such that $M = \sum I_i\mathbf{b}_i$ (see, e.g., [39, Th. 81:3]). The set $((I_i, \mathbf{b}_i))_i$ is called a pseudo-basis of M. A module in K^n can always be viewed as a lattice in $\mathbb{C}^{nd}$ by mapping elements of K to $\mathbb{C}^d$ via the canonical embedding map (for our running example, it is equivalent to mapping a polynomial of degree $< d$ to the vector of its coefficients).

Standard lattice problems, such as finding a full-rank set of linearly independent short vectors in a given lattice, are presumed difficult, even in the context of quantum computations. In order to assess the security of cryptographic schemes based on NTRU/RingSIS/etc, an essential question is whether the restriction to module lattices brings vulnerabilities. Putting aside small polynomial speed-ups relying on the field automorphisms (multiplication by x in our running example), the cryptanalytic state of the art is to view the modules as arbitrary lattices, i.e., forgetting the module structure.

LLL [30] is the central algorithm to manipulate lattice bases. It takes as input a basis of a given lattice, progressively updates it, and eventually outputs another basis of the same lattice that is made of relatively short vectors. Its run-time is polynomial in the input bit-length. For cryptanalysis, one typically relies on BKZ [45] which extends this principle to find shorter vectors at a higher cost. Finding an analogue of LLL for module lattices has been an elusive goal for at least two decades, a difficulty being to even define what that would be. Informally, it should:

- work at the field level (in particular, it should not forget the module structure and view the module just as a lattice);
- it should find relatively short module pseudo-bases by progressively updating the input pseudo-basis;
- it should run in polynomial time with respect to the module rank n and the bit-lengths of the norms of the input vectors and ideals.

The state of the art is far from these goals. Napias [37] proposed such an algorithm for imaginary quadratic fields whose rings of integers are norm-Euclidean, i.e., Euclidean for the algebraic norm. A refined analysis of this algorithm was provided by Camus [12, Se. 1.5]. Fieker and Pohst [18] proposed a general-purpose algorithm. However, it was not proved to provide pseudo-bases consisting of short module vectors, and a cost analysis was provided only for free modules over totally real fields. Fieker [17, p. 47] suggested to use rank-2 module reduction to achieve rank-n module reduction, but there was no follow-up on this approach. Gan, Ling and Mow [21] described and analyzed an LLL algorithm for Gauss integers (i.e., our running example instantiated to $d = 2$). Fieker and Stehlé [20] proposed to apply the LLL algorithm on the lattice corresponding to the module to find short vectors in polynomial time and reconstruct a short pseudo-basis afterwards. More recently, Kim and Lee [26] described such an LLL algorithm for biquadratic fields whose rings of integers are norm-Euclidean, and provided analyses for the shortness of the output and the run-time. They also proposed an extension to arbitrary norm-Euclidean rings, still with a run-time analysis but only conjecturing and experimentally supporting the output quality. In our running example (i.e., power-of-two cyclotomic number fields), asking for a norm-Euclidean number field restricts the applicability to $d \leq 4$ (see [13, 31] for other families of fields).

The rank-2 restriction already captures a fundamental obstacle. The LLL algorithm for 2-dimensional lattices (which is essentially Gauss' algorithm) is a succession of divide-and-swap steps. Given two vectors $\mathbf{b}_1, \mathbf{b}_2 \in \mathbb{Q}^2$, the 'division' consists in shortening $\mathbf{b}_2$ by an integer multiple of $\mathbf{b}_1$. This integer k is the quotient of the Euclidean division of $\langle \mathbf{b}_1, \mathbf{b}_2 \rangle$ by $\|\mathbf{b}_1\|^2$. This leads to a vector $\mathbf{b}_2'$. If the latter is shorter than $\mathbf{b}_1$, then $\mathbf{b}_1$ and $\mathbf{b}_2$ are swapped and a new iteration starts. Crucial to this procedure is the fact that if the projection of $\mathbf{b}_2$ orthogonally to $\mathbf{b}_1$ is very small compared to $\|\mathbf{b}_1\|$, then the division will provide a vector $\mathbf{b}_2'$ that is shorter than $\mathbf{b}_1$. When a swap cannot be made, it means that the projection of $\mathbf{b}_2$ orthogonally to $\mathbf{b}_1$ is not too small, and hence the basis is of good quality, i.e., somewhat orthogonal and hence made of somewhat short vectors. What provides the convergence to a short basis is the Euclideanity of $\mathbb{Z}$. This is why prior works focused on this setup. Put differently, the crucial property is the fact that the covering radius of the $\mathbb{Z}$ lattice is smaller than 1: this makes it possible to shorten a vector $\mathbf{b}_2$ whose projection is sufficiently small by an appropriate integer multiple such that $\mathbf{b}_2'$ becomes smaller than $\mathbf{b}_1$. When we extend to modules, the corresponding lattice becomes R, and its covering radius has no a priori reason to be smaller than 1 (for our running example, it is $\sqrt{d}/2$). Even if we allow an infinite amount of time to find an optimal $k \in R$, the resulting $\mathbf{b}_2 - k\mathbf{b}_1$ may still be longer than $\mathbf{b}_1$, even if $\mathbf{b}_2$ is in the K-span of $\mathbf{b}_1$. This leads us to the following question: does there exist a lattice L_K depending only on K such that being able to solve the Closest Vector Problem (CVP) with respect to L_K allows to find short bases of modules in K^2?

Contributions. The LLL algorithm for Euclidean lattices can be viewed as a way to leverage the ability of Gauss' algorithm to reduce 2-dimensional lattice bases, to reduce n-dimensional lattice bases for any $n \geq 2$. We propose extensions to modules of both Gauss' algorithm and of its LLL leveraging from 2 to n dimensions, hence providing a full-fledged framework for LLL-like reduction of module pseudo-bases.

Our first contribution is an oracle-based algorithm which takes as input a pseudo-basis of a module $M \subset K^n$ over the ring of integers R of an arbitrary number field K, updates it progressively in a fashion similar to the LLL algorithm, and outputs a pseudo-basis of M. The first output vector is short, and the algorithm runs in time polynomial in $\log \Delta_K$ (where Δ_K is the absolute value of the discriminant of the number field K) and the bit-length of the input pseudo-basis. It makes a polynomial number of calls to an oracle that finds short vectors in rank-2 modules. This oracle-based LLL-like algorithm for modules allows us to obtain the following result for our running example (see Theorem 3.9 for a general statement).

Theorem 1.1. *Let $K = \mathbb{Q}[x]/(x^d + 1)$ and $R = \mathbb{Z}[x]/(x^d + 1)$, for d a power of 2. There is a polynomial-time reduction from finding a $(2\gamma d)^{2n-1}$-approximation to a shortest non-zero vector in modules in K^n (with respect to the Euclidean norm inherited from mapping an element of K^n to the concatenation of its n coefficient vectors) to finding a γ-approximation to a shortest non-zero vector in modules in K^2.*

For example, if n is constant, then the reduction allows to obtain polynomial approximation factors in modules in K^n from polynomial approximation factors in modules in K^2.

Our second contribution is a heuristic algorithm to find a very short non-zero vector in an arbitrary module in K^2, given access to a CVP oracle with respect to a lattice depending only on K. We obtain the following result for our running example (see Theorem 4.1 for a general statement).

Theorem 1.2 (Heuristic). *There exists a sequence of lattices L_d and an algorithm $\mathcal{A}$ such that the following holds. Algorithm $\mathcal{A}$ takes as input a pseudo-basis of a rank-2 module $M \subset (\mathbb{Q}/(x^d+1))^2$, and outputs a vector $\mathbf{v} \in M \setminus \{0\}$ that is no more than $2^{(\log d)^{O(1)}}$ longer than a shortest non-zero vector of M. If given access to an oracle solving CVP in L_d, this algorithm runs in quantum polynomial time and makes a polynomial number of calls to the oracle. Finally, for any $\eta > 0$, the lattice L_d can be chosen of dimension $O(d^{2+\eta})$.*

The quantum component of the algorithm is the decomposition of an ideal as the product of a subset of fixed ideals and a principal ideal with a generator [8]. By relying on [5] instead, one can obtain a dequantized variant of Theorem 1.2 relying on more heuristics and in which the algorithm runs in $2^{\widetilde{O}(\sqrt{d})}$ classical time.

In the special case where the input module is free and given by a basis, we show that the algorithm of Theorem 1.2 can be run in *classical* polynomial time, without adding new heuristics. This result also holds when the algorithm of Theorem 1.2 is used in combination with the reduction of Theorem 1.1, provided that the rank-n input module of the reduction is free and given by a basis (note that in this case, it does not necessarily hold that the reduction produces free rank-2 modules). Finally, we observe that this special case is of importance in cryptography, as is captures the NTRU problem.

We insist that the result relies on heuristics. Some are inherited from prior works (such as [41]) and one is new (Heuristic 1 in Section 4). The new heuristic quantifies the distance to L_d of vectors in the real span of L_d that satisfy some properties. This heuristic is difficult to prove as the lattice L_d involves other lattices that are not very well understood (the log-unit lattice and the lattice of class group relations between ideals of small algebraic norms). We justify this heuristic by informal counting arguments and by some experiments in small dimensions.

Finally, we note that the dimension of L_d is near-quadratic in the degree d of the field. This is much more than the lattice dimension d of R, but we do not know how to use a CVP oracle for R to obtain such an algorithm to find short vectors in rank-2 modules. An alternative approach to obtain a similar reduction from finding short non-zero vectors in rank-2 modules to CVP with preprocessing would be as follows: to reach the goal, it suffices to find a short non-zero vector in a $(2d)$-dimensional lattice; by using the LLL algorithm and numerical approximations (see, e.g., [44]), it is possible to further assume that the bit-length of the inputs is polynomial in d; by Kannan's search-to-decision reduction for the shortest vector problem [25], it suffices to obtain an algorithm that decides whether or not a lattice contains a non-zero vector of norm below 1; the latter task can be expressed as an instance of 3SAT, as the corresponding language belongs to NP; finally, 3SAT reduces to CVP with preprocessing [34]. Overall, this gives an alternative to Theorem 1.2 without heuristics, but lattices L_d of much higher dimensions (which still grow polynomially in d).

Technical overview. One of the technical difficulties of extending LLL to modules is the fact that the absolute value $|\cdot|$ over $\mathbb{Q}$ has two canonical generalizations over K: the trace norm and the algebraic norm. Let $(\sigma_i)_{i\le d}$ denote the embedding of K into $\mathbb{C}$. The trace norm and algebraic norm of $x \in K$ are respectively defined as $(\sum_i |\sigma_i(x)|^2)^{1/2}$ and $\prod_i \sigma_i(x)$. When $K = \mathbb{Q}$, the only embedding is the identity map, and both the trace norm and the absolute value of the algebraic norm collapse to the absolute value. When the field degree is greater than 1, they do not collapse, and are convenient for diverse properties. For instance, the trace norm is convenient to measure smallness of a vector over K^n. A nice property is that the bit-length of an element of R is polynomially bounded in the bit-length of the trace norm (for a fixed field K). Oppositely, an element in R may have algebraic norm 1 (in this case, it is called a unit), but can have arbitrarily large bit-length. On the other hand, the algebraic norm is multiplicative, which interacts well with determinants. For example, the determinant of the lattice corresponding to a diagonal matrix over K is simply the product of the algebraic norms of the diagonal entries (up to a scalar depending only on the field K). LLL relies on all these properties, that are conveniently satisfied by the absolute value.

In our first contribution, i.e., the LLL-like algorithm to reduce module pseudo-bases, we crucially rely on the algebraic norm. Indeed, the progress made by the LLL algorithm is measured by the so-called potential function, which is a product of determinants. As observed in prior works [18,26], using the algebraic norm allows for a direct generalization of this potential function to module lattices. What allowed us to go beyond norm-Euclidean number fields is the black-box handling of rank-2 modules. By not considering this difficult component, we can make do with the algebraic norm for the most important parts of the algorithm. The trace norm is still used to control the bit-lengths of the module pseudo-bases occurring during the algorithm, allowing to extend the so-called size-reduction process within LLL, but is not used to "make progress". The black-boxing of the rank-2 modules requires the introduction of a modified condition for deciding which 2-dimensional pseudo-basis to consider to "make progress" on the n-dimensional pseudo-basis being reduced. This condition is expressed as the ratio between 2-determinants, which is compatible with the exclusive use of the algebraic norm to measure progress. It involves the coefficient ideals, which was unnecessary in prior works handling norm-Euclidean fields, as for such fields, all modules can be generated by a basis instead of a pseudo-basis.

Our algorithm for finding short non-zero vectors in rank-2 modules iterates divide-and-swap steps like 2-dimensional LLL (or Gauss' algorithm). The crucial component is the generalization of the Euclidean division, from $\mathbb{Z}$ to R. We are given $a \in K \setminus \{0\}$ and $b \in K$, and we would like to shorten b using R-multiples of a. In the context of $a \in \mathbb{Q} \setminus \{0\}$ and $b \in \mathbb{Q}$, a Euclidean division provides us with $u \in \mathbb{Z}$ such that $|b + ua| \le |a|/2$. We would like to have an analogous division in R. However, the ring R may not be Euclidean. Moreover, the covering radius of the ring R (viewed as a lattice) can be larger than 1, and hence, in most cases, there will not even exist an element $u \in R$ such that $\|b + au\| \leq \|a\|$ (here $\|\cdot\|$ refers to the trace norm). In order to shorten b using a, we also allow b to be multiplied by some element $v \in R$. For this extension to be non-trivial (and useful), we require that v is not too large (otherwise, one can always take $u = b$ and $v = -a$ for instance, if $a, b \in R$, and extend this approach for general $a, b \in K$). Hence, we are interested in finding u, v such that $\|ua + vb\| \leq \varepsilon\|a\|$ and $\|v\| \leq C$ for some $\varepsilon < 1$ and C to be determined later. Intuitively, if we allow for a large number of such multiples v (proportional to $1/\varepsilon$ and to the determinant of the lattice corresponding to R, i.e., the square root of the field discriminant), there should be one such v such that there exists $u \in R$ with $\|vb + au\| \leq \varepsilon\|a\|$. We do not know how to achieve the result with this heuristically optimal number of v's and use potentially larger v's. The astute reader will note that if we use such a v inside a divide-and-swap algorithm, we may end up computing short vectors in sub-modules of the input modules. We prevent this from happening by using the module Hermite Normal Form [7,9,15].

To find u, v such that $\|vb + au\|$ is small, we use the logarithm map Log over K. For this discussion, we do not need to explain how it is defined, but only that it "works" like the logarithm map log over $\mathbb{R}_{>0}$. In particular if $x \approx y$, then $\operatorname{Log} x \approx \operatorname{Log} y$. We would actually prefer to have the converse property, but it does not hold for the standard Log over K. In Subsection 4.1, we propose an extension $\overline{\operatorname{Log}}$ such that $\overline{\operatorname{Log}}\, x \approx \overline{\operatorname{Log}}\, y$ implies that $x \approx y$. In our context, this means that we want to find u, v such that $\overline{\operatorname{Log}}\, v - \overline{\operatorname{Log}}\, u \approx \overline{\operatorname{Log}}(b) - \overline{\operatorname{Log}}(a)$. To achieve this, we will essentially look for such u and v that are product combinations of fixed small elements in R. When applying the $\overline{\operatorname{Log}}$ function, the product combinations become integer combinations of the $\overline{\operatorname{Log}}$'s of the fixed elements. This gives us our CVP instance: the lattice is defined using the $\overline{\operatorname{Log}}$'s of the fixed elements and the target is defined using $\overline{\operatorname{Log}}(b) - \overline{\operatorname{Log}}(a)$. This description is only to provide intuition, as reality is more technical: we use the log-unit lattice and small-norm ideals rather than small-norm elements.

One advantage of using the $\overline{\operatorname{Log}}$ map is that the multiplicative structure of K is mapped to an additive structure, hence leading to a CVP instance. On the downside, one needs extreme closeness in the $\overline{\operatorname{Log}}$ space to obtain useful closeness in K (in this direction, we apply an exponential function). Put differently, we need the lattice to be very dense so that there is a lattice vector that is very close to the target vector. This is the fundamental reason why we end up with a large lattice dimension: we add a large number of $\overline{\operatorname{Log}}$'s of small-norm ideals to densify the lattice. This makes the analysis of the distance to the lattice quite cumbersome, as the Gaussian heuristic gives too crude estimates. For our running example, we have a lattice of dimension $\approx d^2$ and determinant ≈ 1, hence we would expect a 'random' target vector to be at distance $\approx d$ from the lattice. We argue for a distance of at most $\approx \sqrt{d}$ for 'specific' target vectors. Finally, we note that the lattice and its analysis share similarities with the Schnorr-Adleman lattice that Ajtai used to prove NP-hardness of SVP under randomized reductions [2,35] (but we do not know if there is a connection).

Impact. Recent works have showed that lattice problems restricted to ideals of some cyclotomic number fields can be quantumly solved faster than for arbitrary lattices, for some ranges of parameters [16], and for all number fields with not too large discriminant, if allowing preprocessing that depends only on the field [41]. Recall that ideal lattices are rank-1 module lattices. Our work can be viewed as a step towards assessing the existence of such weaknesses for modules of larger rank, which are those that appear when trying to cryptanalyze cryptosystems based on the NTRU, RingSIS, RingLWE, ModuleSIS and ModuleLWE problems and their variants.

Similarly to [16, 41], our results use CVP oracles for lattices defined in terms of the number field only (i.e., defined independently of the input module). In [16, 41], the weaknesses of rank-1 modules stemmed from two properties of these CVP instances: the lattices had dimension quasi-linear in the log-discriminant (quasi-linear in the field degree, for our running example), and either the CVP instances were easy to solve [16], or approximate solutions sufficed [41] and one could rely on Laarhoven's CVP with preprocessing algorithm [27]. In our case, we need (almost) exact solutions to CVP instances for which we could not find any efficient algorithm, and the invariant lattice has a dimension that is more than quadratic in the log-discriminant (in the field degree, for our running example). It is not ruled out that there could be efficient CVP algorithms for such lattices, maybe for some fields, but we do not have any lead to obtain them.

As explained earlier, CVP with preprocessing is known to be NP-complete, so there always exists a fixed lattice allowing to solve the shortest vector problem in lattices of a target dimension. However, the dimension of that fixed lattice grows as a high degree polynomial in the target dimension. The fact that we only need near-quadratic dimensions (when the log-discriminant is quasi-linear in the field degree) may be viewed as a hint that finding short non-zero vectors in rank-2 modules might be easier than finding short non-zero vectors in arbitrary lattices of the same dimension.

Finally, our first result shows the generality of rank-2 modules towards finding short vectors in rank-n modules for any $n \geq 2$. The reduction allows to stay in the realm of polynomial approximation factors (with respect to the field degree) for any constant n. This tends to back the conjecture that there might be a hardness gap between rank-1 and rank-2 modules, and then a smoother transition for higher rank modules.

Notations. We write log the logarithm function in base e. For two real valued functions f and g, we write $f(x) = \widetilde{O}(g(x))$ if and only if there exists some constant $c > 0$ such that $f(x) = O(g(x) \cdot |\log g(x)|^c)$. By abuse of notations, we write $O(x^\alpha \mathrm{poly}(\log x))$ as $\widetilde{O}(x^\alpha)$ even if $\alpha = 0$. We let $\mathbb{Z}, \mathbb{Q}, \mathbb{R}$, and $\mathbb{C}$ denote the sets of integers, rational, real, and complex numbers, respectively. For $x \in \mathbb{C}$, we let $\bar{x}$ denote its complex conjugate. We use lower-case (resp. upper-case) bold letters for vectors (resp. matrices). For vectors $\mathbf{x}_i = (x_{ij})_j$ for $i \leq k$, we write $(\mathbf{x}_1 \| \ldots \| \mathbf{x}_k)$ to denote the vector obtained by concatenation. By default, the matrices are written with column vectors.

For a vector $\mathbf{x} = (x_i)_i \in \mathbb{C}^n$, we write $\|\mathbf{x}\|_i$ for $i \in \{1, 2, \infty\}$ to denote ℓ_i-norm, and we typically omit the subscript when $i = 2$. For a lattice $\Lambda \subset \mathbb{R}^n$, we let $\rho(\Lambda)$ denote the covering radius with respect to Euclidean norm.

2 Preliminaries

In this section, we first recall some necessary algebraic number theory background and discuss some computational aspects. We then extend Gram-Schmidt orthogonalization to matrices over number fields. In this section, we assume that the reader is somehow familiar with the algebraic notions used in this article and in previous works. For more details on these mathematical objects, we refer the reader to [38, Chap. 1] for algebraic number theory questions, to [24] for anything related to modules and to [41] where the same techniques were used in a simpler setting.

2.1 Algebraic background

NUMBER FIELDS. We let K be a number field of degree d and $K_{\mathbb{R}} = K \otimes_{\mathbb{Q}} \mathbb{R}$. A number field comes with r_1 real embeddings and $2r_2$ complex embeddings σ_i's, where $r_1 + 2r_2 = d$. The canonical embedding of K is then defined as $\sigma(x) \in \mathbb{R}^{r_1} \times \mathbb{C}^{2r_2}$, where $\sigma_{r_1+i}(x) = \overline{\sigma_{r_1+r_2+i}(x)}$ for $1 \leq i \leq r_2$. In this work, elements of K are identified to their canonical embeddings. From this perspective, the set $K_{\mathbb{R}}$ is also identified to $\{\mathbf{y} \in \mathbb{R}^{r_1} \times \mathbb{C}^{2r_2} : \forall i \leq r_2 \, , \overline{y_{r_1+r_2+i}} = y_{r_1+i}\}$ (the embedding map σ provides a ring isomorphism between $K_{\mathbb{R}}$ and the latter subspace of $\mathbb{R}^{r_1} \times \mathbb{C}^{2r_2}$). The field norm is defined as $\mathcal{N}(x) = \prod_{i \leq d} \sigma_i(x)$ and the field trace is $\mathrm{Tr}(x) = \sum_{i \leq d} \sigma_i(x)$. The field trace induces a Hermitian inner product over $K_{\mathbb{R}}$ whose associated Euclidean norm is $\|x\| = (\sum_{1 \leq i \leq d} |\sigma_i(x)|^2)^{1/2}$ for $x \in K_{\mathbb{R}}$. We also define $\|x\|_\infty = \max_{i \in [d]} |\sigma_i(x)|$.

We write $K_{\mathbb{R}}^{\times}$ for the subset of vectors in $K_{\mathbb{R}}$ with non-zero entries (it forms a group, for component-wise multiplication). We also write $K_{\mathbb{R}}^{+}$ for the subset of vectors in $K_{\mathbb{R}}$ with non-negative (real) coefficients. For $x \in K_{\mathbb{R}}$, we let $\bar{x}$ refer to the element of $K_{\mathbb{R}}$ obtained by complex conjugation of every coordinate.[6] We can also define a square-root $\sqrt{\cdot} : K_{\mathbb{R}}^{+} \to K_{\mathbb{R}}^{+}$ by taking coordinate-wise square roots.

We let R be the ring of integers of K. It is a free $\mathbb{Z}$-module of rank d, and can be seen as a lattice via the canonical embedding. The absolute value of the discriminant of K, written Δ_K, is then the squared volume of R, i.e., $\Delta_K = |\det((\sigma_i(x_j))_{ij})|^2$ for any $\mathbb{Z}$-basis $(x_i)_{i \leq d}$ of R.

In this article, we will consider families of number fields whose discriminants tend to infinity (i.e., in all complexity statements, the $O(\cdot)$ and other asymptotic notations refer to $\Delta_K \to +\infty$). In order to simplify the cost estimates, we will use the following inequalities

$$\left(\frac{d^d}{d!}\right)^2 \left(\frac{\pi}{4}\right)^d \leq \Delta_K \leq 36 \cdot \rho(R)^{2d},$$

where $\rho(R)$ denotes the covering radius of R when viewed as a lattice. In particular, when Δ_K tends to infinity, it holds that $d = O(\log \Delta_K)$ and that $d \log \rho(R) = \Omega(\log \Delta_K)$. The first inequality is due to Minkowski (see [36, pp. 261–264]). The second one can be obtained from the inclusion $\mathcal{V}(R) \subseteq \mathcal{B}_{\rho(R)}$, where $\mathcal{V}(R) = \{x \in$

[6] Observe that even if complex conjugation might not be well defined over K (i.e., the element $\bar{x}$ might not be in K even if x is), it is however always defined over $K_{\mathbb{R}}$ thanks to its identification to a subspace of $\mathbb{C}^d$. In this article, complex conjugation will only be used on elements of $K_{\mathbb{R}}$, and we make no assumption that K should be stable by conjugation.

$\mathrm{Span}(R) : \|x\| \leq \|x - r\|, \forall r \in R\}$ is the Voronoi cell of the lattice R and $\mathcal{B}_{\rho(R)} = \{x \in \mathrm{Span}(R) : \|x\| \leq \rho(R)\}$ is the ball of radius $\rho(R)$. Observing that $\mathrm{Vol}(\mathcal{V}(R)) = \sqrt{\Delta_K}$ and that $\mathrm{Vol}(\mathcal{B}_{\rho(R)}) = \rho(R)^d \cdot \mathrm{Vol}(\mathcal{B}_1) \leq 6\rho(R)^d$ yields the inequality.

We let $R^\times = \{u \in R \,|\, \exists\, v \in R : uv = 1\}$ denote the group of units of R. Dirichlet's unit theorem states that $R^\times$ is isomorphic to the Cartesian product of a finite cyclic group (formed by the roots of unity contained in K) with the additive group $\mathbb{Z}^{r_1+r_2-1}$. We define $\mathrm{Log} : K_{\mathbb{R}}^\times \to \mathbb{R}^d$ by $\mathrm{Log}(x) = (\log(|\sigma_1(x)|), \ldots, \log(|\sigma_d(x)|))^T$. Let $E = \{x \in \mathbb{R}^d \mid \forall r_1 \leq i \leq r_2 : x_i = x_{i+r_2}\}$. We have $\mathrm{Log}(K_{\mathbb{R}}^\times) \subseteq E$. We also define $H = \{x \in \mathbb{R}^d : \sum_{i\in[d]} x_i = 0\}$ and $\mathbf{1} = (1, \ldots, 1)^T$, which is orthogonal to H in $\mathbb{R}^d$. The set $\Lambda = \{\mathrm{Log}(u) : u \in R^\times\}$ is a lattice, called "log-unit" lattice. It has rank $r_1 + r_2 - 1$, by Dirichlet's units theorem and is full rank in $E \cap H$. Further, its minimum satisfies $\lambda_1(\Lambda) \geq (\log d)/(6d^2)$ (see [19, Cor. 2]).

Ideals. A fractional ideal I of K is an additive subgroup of K which is also stable by multiplication by any element of R, and such that $xI \subseteq R$ for some $x \in \mathbb{Z} \setminus \{0\}$. Any non-zero fractional ideal is also a free $\mathbb{Z}$-module of rank d, and can therefore be seen as a lattice in $K_{\mathbb{R}}$ using the canonical embedding: such lattices are called ideal lattices. The product IJ of two fractional ideals I and J is the fractional ideal generated by all elements xy with $x \in I$ and $y \in J$. Any non-zero fractional ideal I is invertible, i.e., there exists a unique ideal $I^{-1} = \{x \in K : xI \subseteq R\}$ such that $II^{-1} = R$. When $I \subseteq R$, it is said to be an integral ideal. An integral ideal $\mathfrak{p}$ is said to be prime if whenever $\mathfrak{p} = IJ$ with I and J integral, then either $I = \mathfrak{p}$ or $J = \mathfrak{p}$. For a set $\mathfrak{B}$ of prime ideals, we say that an integral ideal I is $\mathfrak{B}$-smooth if I can be factored as a product of ideals in $\mathfrak{B}$. For any $g \in K$, we write $\langle g \rangle = gR$ the smallest fractional ideal containing g, and we say that it is a principal ideal. The quotient of the group of non-zero fractional ideals (for ideal multiplication) by the subgroup consisting in principal ideals is the class group $\mathcal{C}l_K$. Its cardinality h_K is called the class number. Under the GRH, there is a set of cardinality $\leq 2\log h_K = \widetilde{O}(\log \Delta_K)$ of prime ideals of norms $\leq 12\log^2 \Delta_K$ that generates $\mathcal{C}l_K$ (see, e.g., [41, Se. 2.3]). We also will use the bound $h_K \cdot (\det \Lambda) \leq 2^{\widetilde{O}(\log \Delta_K)}$ (see, e.g., [41, Se. 2.4]).

The algebraic norm $\mathcal{N}(I)$ of an integral ideal I is its index as a subgroup of R, and is equal to $\det(\sigma(I))/\Delta_K^{1/2}$. The algebraic norm of a prime ideal is a power of a prime number. For a principal ideal, we also have $\mathcal{N}(\langle g \rangle) = |\mathcal{N}(g)|$. The norm extends to fractional ideals using $\mathcal{N}(I) = \mathcal{N}(xI)/|\mathcal{N}(x)|$, for any $x \in R \setminus \{0\}$ such that $xI \subseteq R$. We have $\mathcal{N}(IJ) = \mathcal{N}(I)\mathcal{N}(J)$ for all fractional ideals I, J.

Lemma 2.1 ([4, Th. 8.7.4]). *Assume the GRH. Let $\pi_K(x)$ be the number of prime integral ideals of K of norm $\leq x$. Then there exists an absolute constant C (independent of K and x) such that $|\pi_K(x) - \mathrm{li}(x)| \leq C \cdot \sqrt{x}(d \log x + \log \Delta_K)$, where $\mathrm{li}(x) = \int_2^x \frac{\mathrm{d}t}{\log t} \sim \frac{x}{\log x}$.*

Module lattices and their geometry. In this work, we call (R-)module any set of the form $M = I_1\mathbf{b}_1 + \ldots + I_n\mathbf{b}_n$, where the I_j's are non-zero fractional ideals of R and the $\mathbf{b}_j$'s are $K_{\mathbb{R}}$-linearly independent[7] vectors in $K_{\mathbb{R}}^m$, for some $m > 0$. The

[7] The vectors $\mathbf{b}_j$'s are said to be $K_{\mathbb{R}}$-linearly independent if and only if there is no non-trivial way to write the zero vector as a $K_{\mathbb{R}}$-linear combination of the $\mathbf{b}_j$'s. Because $K_{\mathbb{R}}$ is a ring and not a field, this definition is stronger than requiring that none of the $\mathbf{b}_j$'s is in the span of the others.

tuple of pairs $((I_1, \mathbf{b}_1), \ldots, (I_n, \mathbf{b}_n))$ is called a pseudo-basis of M, and n is its rank. In particular, fractional ideals are rank-1 modules contained in K, and sets of the form $\alpha \cdot I$ for $\alpha \in K_{\mathbb{R}}^{\times}$ and a non-zero fractional ideal I are rank-1 modules in $K_{\mathbb{R}}$. We refer to [24] for a thorough study of R-modules, and concentrate here on the background necessary to the present work.

Two pseudo-bases $((I_1, \mathbf{b}_1), \ldots, (I_n, \mathbf{b}_n))$ and $((J_1, \mathbf{c}_1), \ldots, (J_n, \mathbf{c}_n))$ represent the same module if and only if there exists $\mathbf{U} = (u_{ij})_{i,j} \in K^{n \times n}$ invertible such that $\mathbf{C} = \mathbf{B} \cdot \mathbf{U}$; we have $u_{ij} \in I_i J_j^{-1}$ and $u'_{ij} \in J_i I_j^{-1}$ for all i, j and for $\mathbf{U}' = (u'_{ij})_{i,j} := \mathbf{U}^{-1}$. Here, the matrix $\mathbf{B}$ is the concatenation of the column vectors $\mathbf{b}_i$ (and similarly for $\mathbf{C}$). If $n > 0$, we define $\det_{K_{\mathbb{R}}} M = \det(\overline{\mathbf{B}}^{\top}\mathbf{B})^{1/2} \cdot \prod_i I_i$. It is an R-module in $K_{\mathbb{R}}$. Note that it is a module invariant, i.e., it is identical for all pseudo-bases of M.

We extend the canonical embedding to vectors $\mathbf{v} = (v_1, \ldots, v_m)^T \in K_{\mathbb{R}}^m$ by defining $\sigma(\mathbf{v})$ as the vector of $\mathbb{R}^{dm}$ obtained by concatenating the canonical embeddings of the v_i's. This extension of the canonical embedding maps any module M of rank n to a (dn)-dimensional lattice in $\mathbb{R}^{dm}$. We abuse notation and use M to refer to both the module and the lattice obtained by applying the canonical embedding. The volume of a module M seen as a lattice is $\det M = \Delta_K^{n/2} \cdot \mathcal{N}(\det_{K_{\mathbb{R}}} M)$. This matches with the module determinant definition from [20, Se. 2.3].

We consider the following inner products for $\mathbf{a}, \mathbf{b} \in K_{\mathbb{R}}^m$:

$$\langle \mathbf{a}, \mathbf{b} \rangle_{K_{\mathbb{R}}} = \sum_{i \in [m]} a_i \bar{b}_i \in K_{\mathbb{R}} \quad \text{and} \quad \langle \mathbf{a}, \mathbf{b} \rangle = \mathrm{Tr}(\sum_{i \in [m]} a_i \bar{b}_i) \in \mathbb{C}.$$

Note that we have $\langle \mathbf{v}, \mathbf{v} \rangle_{K_{\mathbb{R}}} \in K_{\mathbb{R}}^{+}$, as all $\sigma_i(\langle \mathbf{v}, \mathbf{v} \rangle_{K_{\mathbb{R}}})$'s are non-negative. For $\mathbf{v} \in K_{\mathbb{R}}^m$, we define $\|\mathbf{v}\|_{K_{\mathbb{R}}} = \sqrt{\langle \mathbf{v}, \mathbf{v} \rangle_{K_{\mathbb{R}}}}$ and $\|\mathbf{v}\| = \sqrt{\mathrm{Tr}(\langle \mathbf{v}, \mathbf{v} \rangle_{K_{\mathbb{R}}})} = \sqrt{\langle \mathbf{v}, \mathbf{v} \rangle}$. Observe that $\|\mathbf{v}\|$ correspond to the Euclidean norm of $\mathbf{v}$ when seen as a vector of dimension dm via the canonical embedding. We extend the infinity norm to vectors $\mathbf{v} \in K_{\mathbb{R}}^m$ by $\|\mathbf{v}\|_\infty = \max_{i \in [m]} \|v_i\|_\infty$, where $\mathbf{v} = (v_1, \ldots, v_m)$. We also extend the algebraic norm to vectors $\mathbf{v} \in K_{\mathbb{R}}^m$ by setting $\mathcal{N}(\mathbf{v}) := \mathcal{N}(\|\mathbf{v}\|_{K_{\mathbb{R}}})$. For $m = 1$, we see that $\mathcal{N}(\mathbf{v}) = |\mathcal{N}(\mathbf{v})|$. By the arithmetic-geometric inequality, we have $\sqrt{d} \cdot \mathcal{N}(\mathbf{a})^{1/d} \leq \|\mathbf{a}\|$ for $\mathbf{a} \in K_{\mathbb{R}}^m$. Observe also that for any vector $\mathbf{v} = (v_1, \ldots, v_m)^T \in K_{\mathbb{R}}$, we have $\mathcal{N}(\mathbf{v}) \geq \max_i(\mathcal{N}(v_i))$, because for any embedding σ_j, it holds that $|\sigma_j(v_1\overline{v_1} + \cdots + v_m\overline{v_m})| = |\sigma_j(v_1)|^2 + \cdots + |\sigma_j(v_m)|^2 \geq \max_i |\sigma_j(v_i)|^2$.

We define the module minimum $\lambda_1(M)$ as the norm of a shortest non-zero element of M with respect to $\|\cdot\|$. Our module-LLL algorithm will rely on the algebraic norm rather than the Euclidean norm. For this reason, we will also be interested to the minimum $\lambda_1^{\mathcal{N}}(M) = \inf(\mathcal{N}(\mathbf{v}) : \mathbf{v} \in M \setminus \{\mathbf{0}\})$. We do not know if this minimum is always reached for some vector $\mathbf{v} \in M$, but we can find an element of M whose algebraic norm is arbitrarily close to $\lambda_1^{\mathcal{N}}(M)$. The following lemma provides relationships between $\lambda_1(M)$ and $\lambda_1^{\mathcal{N}}(M)$.

Lemma 2.2. *For any rank-n module M, we have:*

$$d^{-d/2}\lambda_1(M)^d \Delta_K^{-1/2} \leq \lambda_1^{\mathcal{N}}(M) \leq d^{-d/2}\lambda_1(M)^d \leq n^{d/2}\Delta_K^{1/2}\mathcal{N}(\det_{K_{\mathbb{R}}} M)^{1/n}.$$

Proof. Let $\mathbf{s} \in M \setminus \{\mathbf{0}\}$ of minimal Euclidean norm. By the arithmetic-geometric inequality, we have $\mathcal{N}(\mathbf{s}) \leq d^{-d/2}\|\mathbf{s}\|^d$. By Minkowski's theorem applied to the canonical embedding of M, we have $\|\mathbf{s}\| \leq \sqrt{nd} \cdot (\det M)^{1/(nd)}$. Using the equality $\det M = \Delta_K^{n/2} \mathcal{N}(\det_{K_\mathbb{R}} M)$ allows to obtain the last two inequalities. Next, for any $\mathbf{s} \in M \setminus \{\mathbf{0}\}$, Minkowski's theorem applied to the lattice $\sigma(R \cdot \mathbf{s})$ gives us $\lambda_1(M) \leq \lambda_1(R \cdot \mathbf{s}) \leq \sqrt{d} \cdot \Delta_K^{1/2} \cdot \mathcal{N}(\mathbf{s})^{1/d}$. The last inequality follows by definition of the infimum. □

2.2 Computing over rings

In this work, we assume that we know an LLL-reduced $\mathbb{Z}$-basis $(r_1, \ldots, r_d)$ of R, with respect to $\|\cdot\|$. Note that computing a $\mathbb{Z}$-basis of R is, in the worst-case, an expensive task (see, e.g., [14, Se. 6.1]). Once such a basis is known, applying the LLL algorithm to it has a bit-complexity that is polynomial in $\log \Delta_K$ and $\max \log \|r_i\|$. Note that the spanned lattice and the positive definite quadratic form may not be integral, but LLL-reduction can be performed by taking approximations to a polynomially bounded precision, because a lower bound for $\lambda_1(R)$ is known (we have $\lambda_1(R) \geq 1$). We refer to [11, 44] for LLL-reduction of non-integral lattices.

Representing elements and ideals. For computations, elements of R can be represented as integer linear combinations of such a LLL-reduced $\mathbb{Z}$-basis of R. The following lemma provides a bound for the involved integer coefficients.

Lemma 2.3 ([**20**, **Le. 2**]). *Let $(r_i)_{i \leq d}$ be a $\mathbb{Z}$-basis of R that is* LLL*-reduced with respect to $\|\cdot\|$. For all $x = \sum x_i r_i \in K$, we have $\max_i |x_i| \leq 2^{3d/2} \|x\|$.*

This means in particular that any $x \in R$ can be represented with at most $\operatorname{poly}(d, \log \|x\|)$ many bits. We will also use the fact that $\max_i \|r_i\| \leq (4d)^{d/2} \Delta_K^{1/2}$, which is implied by the facts that $\max_i \|r_i\|$ is no more than 2^d times longer than the last minimum of R (by LLL-reducedness), by Minkowski's second theorem, and the lower bound $\lambda_1(R) \geq 1$. This implies in particular that for any $x \in K$, the quantity $\log \|x\|$ is polynomially bounded by $\log \Delta_K$ and the bit-length of x (when represented by a vector $(x_1, \cdots, x_d)^T \in \mathbb{Q}^d$ such that $x = \sum_i x_i r_i$).

An ideal can be represented by a $\mathbb{Z}$-basis $(b_i)_{i \leq d}$ with the b_i's belonging to K. The following lemma can be used in combination with Lemma 2.3 to bound the bit-length of a representation of an ideal. The proof follows from the fact that $\det \sigma(I) = \sqrt{\Delta_K} \mathcal{N}(I)$ and from standard LLL-reduction inequalities [30, p. 518].

Lemma 2.4. *Let $(b_i)_{i \leq d}$ be a $\mathbb{Z}$-basis of a fractional ideal $I \subset K$ that is* LLL*-reduced with respect to $\|\cdot\|$. Then $\prod_i \|b_i\| \leq 2^{d^2} \cdot \sqrt{\Delta_K} \cdot \mathcal{N}(I)$.*

Proof. We have $\det \sigma(I) = \sqrt{\Delta_K} \mathcal{N}(I)$. On the other hand, we also have $\det \sigma(I) = \prod_i \|b_i^*\|$, where the set $\{b_i^*\}$ is the Gram-Schmidt basis of $\{b_i\}$. Each LLL-reduced basis vector b_i has norm at most $2^{(d-1)/2}$ times the norm of its Gram-Schmidt vector. Therefore, we have $\prod_i \|b_i\| \leq \prod_i 2^{(d-1)/2} \|b_i^*\| \leq 2^{(d-1)d/2} \cdot \det \sigma(I)$. This completes the proof. □

This lemma implies that an ideal can be represented in bit-length polynomial in $\log \Delta_K$, $\log \mathcal{N}(xI)$ and $\log x$, where x is the smallest positive integer such that $xI \subseteq R$.

Computations with an oracle. In Section 4, we will assume that we have access to an oracle for the Closest Vector Problem, for lattices related to K. For example, we will assume that we can solve CVP for the lattice corresponding to R, with respect to $\|\cdot\|$. This lattice has dimension d.

In a similar vein, we will use the following adaptation from [41, Th. 3.4], to find short elements in rank-1 modules.

Lemma 2.5 (Heuristic). *There exists a lattice $L_{K,1}$ of dimension $\widetilde{O}(\log \Delta_K)$ and depending only on K such that, given access to an oracle solving* CVP *in $L_{K,1}$, the following holds. There exists a heuristic quantum algorithm that takes as input a fractional ideal I of R and any $\alpha \in K_{\mathbb{R}}^{\times}$, and outputs $x \in \alpha I \setminus \{0\}$ such that*

$$\|x\|_\infty \leq \mathrm{c} \cdot |\mathcal{N}(\alpha)|^{1/d} \cdot \mathcal{N}(I)^{1/d},$$

where $\mathrm{c} = 2^{\widetilde{O}(\log \Delta_K)/d}$. In particular, we have $\|x\|_\infty \leq \mathrm{c} \cdot |\mathcal{N}(x)|^{1/d}$.

If $\alpha \in K$, then the algorithm performs a number of quantum operations that is polynomial in $\log \Delta_K$ and the input bit-length, and makes a single call to the oracle solving CVP in $L_{K,1}$.

The result assumes GRH and Heuristic 4 from [41]. The quantum computation performed by the algorithm derives from [8] and consists in computing the log-unit lattice, finding a small generating set $([\mathfrak{p}_i])_i$ of the class group $\mathcal{C}l_K$ of K, and decomposing the class $[I]$ of I in $\mathcal{C}l_K$ in terms of that generating set. These quantum computations can be replaced by classical ones (e.g., [5, 6, 22]), at the expense of increased run-times and additional heuristic assumptions.

The lemma can be derived from [41, Th. 3.4] by replacing Laarhoven's CVPP algorithm [27] by an exact CVPP oracle. In [41], the CVPP algorithm is used with a target vector $\mathbf{t}$ derived from the decomposition of $[I]$ on the $[\mathfrak{p}_i]$'s and the logarithm $\mathrm{Log}(g)$ of an element $g \in K$. To obtain the statement above, we replace $\mathrm{Log}(g)$ by $\mathrm{Log}(g \cdot \alpha) = \mathrm{Log}(g) + \mathrm{Log}(\alpha)$. The last lemma statement $\|x\|_\infty \leq \mathrm{c}|\mathcal{N}(x)|^{1/d}$ comes from the observation that $|\mathcal{N}(x)| \geq \mathcal{N}(\alpha) \cdot \mathcal{N}(I)$ (which holds because x belongs to $\alpha I \setminus \{0\}$).

2.3 Gram-Schmidt orthogonalization

We extend Gram-Schmidt Orthogonalization from matrices over the real numbers to matrices over $K_{\mathbb{R}}^m$. For $(\mathbf{b}_1, \ldots, \mathbf{b}_n) \in K_{\mathbb{R}}^{m \times n}$ such that $\mathbf{b}_1, \ldots, \mathbf{b}_n$ are $K_{\mathbb{R}}$-linearly independent, we define $\mathbf{b}_1^* = \mathbf{b}_1$ and, for $1 < i \leq n$:

$$\mathbf{b}_i^* = \mathbf{b}_i - \sum_{j<i} \mu_{ij} \mathbf{b}_j^* \quad \text{with} \quad \forall j < i: \; \mu_{ij} = \frac{\langle \mathbf{b}_i, \mathbf{b}_j^* \rangle_{K_{\mathbb{R}}}}{\langle \mathbf{b}_j^*, \mathbf{b}_j^* \rangle_{K_{\mathbb{R}}}}.$$

It may be checked that $\langle \mathbf{b}_i^*, \mathbf{b}_j^* \rangle = 0$ for $i \neq j$, and that $\mathbf{b}_i^* = \mathrm{argmin}(\|\mathbf{b}_i - \mathbf{y}\| \mid \mathbf{y} \in \mathrm{Span}_{K_{\mathbb{R}}}(\mathbf{b}_j)_{j<i})$. Note that the linear independence assumption implies that $\|\mathbf{b}_j^*\|_{K_{\mathbb{R}}}$ is invertible and hence that μ_{ij} is well-defined for every $j < i$.

We also extend the QR-factorization to matrices over $K_{\mathbb{R}}$. We define $r_{ii} = \|\mathbf{b}_i^*\|_{K_{\mathbb{R}}}$ for $i \leq n$, $r_{ij} = \mu_{ji} r_{ii}$ when $i < j$, and $r_{ij} = 0$ when $i > j$. We then have $\mathbf{B} = \mathbf{Q} \cdot \mathbf{R}$, where $\mathbf{Q} \in K_{\mathbb{R}}^{m \times n}$ is the matrix whose columns are the $\mathbf{b}_i^* / \|\mathbf{b}_i^*\|_{K_{\mathbb{R}}}$'s and $\mathbf{R} = (r_{ij})_{ij}$. Note that $\overline{\mathbf{Q}}^T \mathbf{Q} = \mathbf{Id}$ and that $\mathbf{R}$ is upper-triangular with diagonal coefficients in $K_{\mathbb{R}}^+$.

The following lemma provides relationships between some module invariants and the QR-factorization.

Lemma 2.6. *Let $M \subset K_{\mathbb{R}}^m$ be a module with pseudo-basis $((I_i, \mathbf{b}_i))_{i \leq n}$. Let $\mathbf{R}$ be the R-factor of $\mathbf{B}$. Then, we have $\det_{K_{\mathbb{R}}} M = \prod_i r_{ii} I_i$ and $\det M = \Delta_K^{n/2} \prod_i \mathcal{N}(r_{ii} I_i)$. Further, for any vector $\mathbf{v} \in K_{\mathbb{R}}^m$ and fractional ideal $I \subset K$ such that $0 \subsetneq \mathbf{v} I \subseteq M$, it holds that $\mathcal{N}(\mathbf{v}) \cdot \mathcal{N}(I) \geq \min_i \mathcal{N}(r_{ii} I_i)$. This implies in particular that $\lambda_1^{\mathcal{N}}(M) = \inf_{\mathbf{s} \in M \setminus \{\mathbf{0}\}} \mathcal{N}(\mathbf{s}) \geq \min_i \mathcal{N}(r_{ii} I_i)$.*

Proof. Recall that we have $\det_{K_{\mathbb{R}}} M = (\det \overline{\mathbf{B}}^{\top} \mathbf{B})^{1/2} \cdot \prod_i I_i$. Using the QR-decomposition (and in particular the facts that $\overline{\mathbf{Q}}^T \mathbf{Q} = \mathbf{Id}$ and that $\mathbf{R}$ is upper-triangular with diagonal coefficients in $K_{\mathbb{R}}^+$), this rewrites as $\det_{K_{\mathbb{R}}} M = (\det \overline{\mathbf{R}}^{\top} \mathbf{R})^{1/2} \cdot \prod_i I_i = \prod_i r_{ii} I_i$. The equality $\det M = \Delta_K^{n/2} \mathcal{N}(\det_{K_{\mathbb{R}}} M)$ leads to the first statement.

For the second statement, note that for any $\mathbf{v} \in K_{\mathbb{R}}^m$ in the $K_{\mathbb{R}}$-span of M (or, equivalently, in the $K_{\mathbb{R}}$-span of the columns of $\mathbf{B}$), we have $\|\overline{\mathbf{Q}}^T \mathbf{v}\|_{K_{\mathbb{R}}} = \|\mathbf{v}\|_{K_{\mathbb{R}}}$ and so $\mathcal{N}(\overline{\mathbf{Q}}^T \mathbf{v}) = \mathcal{N}(\mathbf{v})$. Also, if $0 \subsetneq \mathbf{v} I \subseteq M$ for a non-zero fractional ideal I, then $\overline{\mathbf{Q}}^T \mathbf{v} I \neq 0$ is contained in the module M' spanned by the pseudo-basis $((I_i, \mathbf{r}_i))_{i \leq n}$, where the $\mathbf{r}_i$'s are the columns of $\mathbf{R}$. Let us define $\mathbf{v}' = \overline{\mathbf{Q}}^T \mathbf{v}$. We write $\mathbf{v}' = \sum x_i \mathbf{r}_i$. Note that the x_i's do not necessarily belong to the ideals I_i, because $\mathbf{v}'$ does not necessarily belongs to M'. Consider $i_0 = \max(i | \forall j > i : x_j = 0)$ (such a i_0 exists since $\mathbf{v}' \neq 0$). Because the matrix $\mathbf{R}$ is upper triangular and $\mathbf{v}' I \subseteq M'$, we know that $x_{i_0} r_{i_0, i_0} I \subseteq r_{i_0, i_0} I_{i_0}$, and, in particular, that $\mathcal{N}(x_{i_0} r_{i_0, i_0} I) \geq \mathcal{N}(r_{i_0, i_0} I_{i_0}) \geq \min_i \mathcal{N}(r_{ii} I_i)$ (because $x_{i_0} r_{i_0, i_0} I \neq \{0\}$). We conclude using the fact that $\mathcal{N}(\mathbf{v}) = \mathcal{N}(\mathbf{v}') \geq \mathcal{N}(x_{i_0} r_{i_0, i_0})$ (because $x_{i_0} r_{i_0, i_0}$ is the i_0-th coefficient of $\mathbf{v}'$).

The lower bound on $\lambda_1^{\mathcal{N}}(M)$ follows by taking $\mathbf{v} \in M$, $I = R$ and letting $\mathcal{N}(\mathbf{v})$ tend to $\lambda_1^{\mathcal{N}}(M)$. □

In this work, we will mostly rely on QR-factorization. It carries the same information as Gram-Schmidt orthogonalization, but allows for simpler explanations. However, from a computational perspective, the R-factor may be difficult to represent exactly even for modules contained in K^m, because of the square roots appearing in its definition. This difficulty is circumvented by computing the Gram-Schmidt orthogonalization instead, and using it as a means to represent the R-factor.

Computing Gram-Schmidt orthogonalizations. We first note that the Gram-Schmidt coefficients may not belong to K even if the pseudo-basis does. To explain how to exactly represent the Gram-Schmidt orthogonalization, we need to backtrack a little to operations in K. As seen before (in Lemma 2.3), an element x in R is represented by a vector in $\mathbb{Z}^d$ storing the coefficients of x with respect to a LLL-reduced basis $(r_i)_{i \leq d}$ of R (for $\| \cdot \|$). Multiplication between $x_1, x_2 \in R$ is performed thanks to a table (of $O(d^3)$ integers) storing the representations of each term $r_i r_j$ for all $i, j \leq d$. An element x in K is represented by a pair $(x_{num}, x_{den}) \in R^2$ such that $x = x_{num}/x_{den}$ (and both x_{num} and x_{den} are themselves represented by vectors on $\mathbb{Z}^d$, as explained above). All the above enables additions, multiplications and divisions in K. Now, when computing the Gram-Schmidt orthogonalization, we will make use of complex conjugation in $K_{\mathbb{R}}$ (as we use a Hermitian inner product). Recall that for $x \in K_{\mathbb{R}}$, the element $\bar{x} \in K_{\mathbb{R}}$ is obtained by coordinate-wise complex conjugation of its embedding vector. We define $\overline{R}$ and $\overline{K}$ as the subsets of $K_{\mathbb{R}}$ obtained by applying this operator to the elements of R and K, respectively. These elements can be represented using the $\overline{r}_i$'s rather than the r_i's. We also define $\overline{R}R = \{\overline{y}x : \overline{y} \in \overline{R}, x \in R\}$. Every element $x \in \overline{R}R$ can be expressed as an integer combination of the d^2 elements $\overline{r}_i r_j$ (for $i, j \leq d$), and this vector in $\mathbb{Z}^{d^2}$ is used to represent x. The bit-length of an element in $\overline{R}R$ is the bit-length of this vector in $\mathbb{Z}^{d^2}$. The multiplication table for R allows to perform multiplication in $\overline{R}R$.

Lemma 2.7. *Let $\mathbf{b}_1, \ldots, \mathbf{b}_n \in K^m$ be K-linearly independent. Then the coefficients of the $\mathbf{b}_i^*$'s and μ_{ij}'s can be written as fractions of elements in $\overline{R}R$ whose bit-lengths are polynomially bounded with respect to* $\max_i(\log x + \log \|x\mathbf{b}_i\|)$*, where x is the smallest positive integer such that $x\mathbf{b}_i \in R^m$ for all i.*

Proof. Without loss of generality, we assume that the $\mathbf{b}_i$'s belong to R^m. Let $d_i = \det(\overline{\mathbf{B}}_i^T \mathbf{B}_i)$ with $\mathbf{B}_i = (\mathbf{b}_1, \ldots, \mathbf{b}_i)$ for $i \leq n$. Then, by a direct adaptation of [30, p. 523], we have $d_i \in \overline{R}R$, $d_{i-1}\mathbf{b}_i^* \in (\overline{R}R)^m$ and $d_j\mu_{ij} \in \overline{R}R$ for all $j < i$. This implies, using Lemma 2.3, that the coefficients of the $\mathbf{b}_i^*$'s and the μ_{ij}'s can be written as fractions of elements in $\overline{R}R$ with d_{i-1} and d_j as denominator, respectively. Going down to the expressions in terms of integer combinations in the r_i's, $\overline{r}_j$'s and $\overline{r}_j r_i$'s, it may be checked that these numerators and denominators are sums and products of a polynomial number of terms $x_i r_i$ and $x_j \overline{r}_j$, where each x_i and x_j is an integer. Further, by Lemma 2.3, each such integer is polynomially bounded with respect to $\max_i \log \|\mathbf{b}_i\|$. This allows to complete the proof. $\square$

For lattices, if we have a basis and a full-rank family of short vectors, then we can efficiently obtain a basis of the lattice whose Gram-Schmidt vectors are no longer than those of the full-rank family of short vectors. This was generalized to modules in [20], relying on the extension to modules of the Hermite Normal Form [7, 9, 15].

Lemma 2.8 (**[20, Th. 4]**). *There exists an algorithm that takes as inputs a pseudo-basis $((I_i, \mathbf{b}_i))_{i \leq n}$ of a module $M \subset K_{\mathbb{R}}^m$ and a full-rank set of vectors $(\mathbf{s}_i)_{i \leq n}$ of M and outputs a pseudo-basis $((J_i, \mathbf{c}_i))_{i \leq n}$ such that $\mathbf{c}_i \in M$ and $\mathbf{c}_i^* = \mathbf{s}_i^*$ for all i. If $M \subset K^m$, then it terminates in time polynomial in $\log \Delta_K$ and in the input bit-length.*

Note that the condition that $\mathbf{c}_i \in M$ implies that $R \subseteq J_i$, for all i.

3 LLL-reduction of module pseudo-bases

LLL-reduction of lattice bases is defined in terms of Gram-Schmidt orthogonalization (or, equivalently, QR-factorization). A basis is said LLL-reduced if two conditions are satisfied. The first one, often referred to as size-reduction condition, states that any off-diagonal coefficients r_{ij} of the R-factor should have a small magnitude compared to the diagonal coefficient r_{ii} on the same row. The second one, often referred to as Lovász' condition, states that the 2-dimensional vector $(r_{i,i}, 0)^T$ is no more than $1/\delta$ times longer than $(r_{i,i+1}, r_{i+1,i+1})^T$, for some parameter $\delta < 1$. The size-reduction condition allows to ensure that the norms of the vectors during the LLL execution and at its completion stay bounded. More importantly, in combination with Lovász' condition, it makes it impossible for $r_{i+1,i+1}/r_{i,i}$ to be arbitrarily small (for an LLL-reduced basis). The latter is the crux of both the LLL output quality and its fast termination.

3.1 An LLL algorithm for module lattices

When extending to rings, the purpose of the size-reduction condition is better expressed in terms of the Euclidean norm $\|\cdot\|$, whereas the bounded decrease of the r_{ii}'s is better quantified in terms of the algebraic norm $\mathcal{N}(\cdot)$. This discrepancy makes the definition of a LLL-reduction algorithm for modules difficult. In this section, we circumvent this difficulty by directly focusing on the decrease of the r_{ii}'s, deferring to later sections the handling of the rank-2 modules of pseudo-bases $((I_i, (r_{i,i}, 0)^T), (I_{i+1}, (r_{i,i+1}, r_{i+1,i+1})^T))$. We also defer to later the bounding of bit-lengths.

Definition 3.1 (LLL-reducedness of a pseudo-basis). *A module pseudo-basis $((I_i, \mathbf{b}_i))_{i\leq n}$ is called LLL-reduced with respect to a parameter $\alpha_K \geq 1$ if, for all $i < n$, we have:*

$$\mathcal{N}(r_{i+1,i+1}I_{i+1}) \geq \frac{1}{\alpha_K}\cdot\mathcal{N}(r_{i,i}I_i), \tag{3.1}$$

where $\mathbf{R} = (r_{i,j})_{i,j}$ refers to the R-factor of the matrix basis $\mathbf{B}$.

We first explain that LLL-reduced pseudo-bases are of interest, and we will later discuss their computation (for some value of α_K).

Lemma 3.2. *Assume that $((I_i, \mathbf{b}_i))_{i\leq n}$ is an LLL-reduced pseudo-basis of a module M. Then:*

$$\mathcal{N}(I_1)\mathcal{N}(\mathbf{b}_1) \leq \alpha_K^{(n-1)/2}\cdot(\mathcal{N}(\det_{K_\mathbb{R}} M))^{1/n},$$
$$\mathcal{N}(I_1)\mathcal{N}(\mathbf{b}_1) \leq \alpha_K^{n-1}\cdot\lambda_1^{\mathcal{N}}(M).$$

Proof. From (3.1), we obtain $\mathcal{N}(I_1)\mathcal{N}(\mathbf{b}_1) \leq \alpha_K^i\mathcal{N}(r_{i,i}I_i)$ for all $i \leq n$. Taking the product over all i's gives $(\mathcal{N}(I_1)\mathcal{N}(\mathbf{b}_1))^n \leq \alpha_K^{n(n-1)/2}\mathcal{N}(\det_{K_\mathbb{R}} M)$, by Lemma 2.6. The proof of the first inequality can be completed by taking the n-th root. The second inequality can be obtained by combining the last claim of Lemma 2.6 with (3.1). □

Our LLL algorithm for modules is very similar to the one over the integers. The algorithm proceeds by finding an approximation to a shortest non-zero element in a rank-2 module, with respect to the algebraic norm. Using Lemma 2.2, we obtain a sufficient condition on α_K such that Algorithm 3.1 terminates. In particular, if α_K is sufficiently large, then $\mathcal{N}(r_{i+1,i+1}I_{i+1}) < \frac{1}{\alpha_K}\mathcal{N}(r_{i,i}I_i)$ implies that there is a vector $\mathbf{s}$ in the local projected rank-2 module of norm significantly less than $\mathcal{N}(r_{i,i}I_i)$.

Lemma 3.3. *Take the notations of Algorithm 3.1, and consider an index $i < n$ such that $\alpha_K\cdot\mathcal{N}(r_{i+1,i+1}I_{i+1}) < \mathcal{N}(r_{i,i}I_i)$. We have $\mathcal{N}(\mathbf{s}_i) \leq {\gamma_\mathcal{N}}^d\sqrt{\frac{2^d\Delta_K}{\alpha_K}}\mathcal{N}(r_{i,i}I_i)$.*

Algorithm 3.1 LLL-reduction over K

```
Input: A pseudo-basis $((I_i, \mathbf{b}_i))_{i\le n}$ of a module $M \subset K^m$.
Output: An LLL-reduced pseudo-basis of $M$.
1: while there exists $i < n$ such that $\alpha_K \cdot \mathcal{N}(r_{i+1,i+1}I_{i+1}) < \mathcal{N}(r_{i,i}I_i)$ do
2:     Define $M_i$ as the rank-2 module spanned by $((I_i, \mathbf{a}_i), (I_{i+1}, \mathbf{a}_{i+1}))$, with $\mathbf{a}_i = (r_{ii}, 0)^T$
       and $\mathbf{a}_{i+1} = (r_{i,i+1}, r_{i+1,i+1})^T$;
3:     Find $\mathbf{s}_i \in M_i \setminus \{\mathbf{0}\}$ such that $\mathcal{N}(\mathbf{s}_i) \le {\gamma_{\mathcal{N}}}^d \cdot \lambda_1^{\mathcal{N}}(M_i)$;
4:     Set $\mathbf{s}_{i+1} = \mathbf{a}_i$ if it is linearly independent with $\mathbf{s}_i$, and $\mathbf{s}_{i+1} = \mathbf{a}_{i+1}$ otherwise;
5:     Call the algorithm of Lemma 2.8 with $((I_i, \mathbf{a}_i), (I_{i+1}, \mathbf{a}_{i+1}))$ and $(\mathbf{s}_i, \mathbf{s}_{i+1})$ as inputs, and let
       $((I'_i, \mathbf{a}'_i), (I'_{i+1}, \mathbf{a}'_{i+1}))$ denote the output;
6:     Update $I_i := I'_i$, $I_{i+1} := I'_{i+1}$ and $[\mathbf{b}_i|\mathbf{b}_{i+1}] := [\mathbf{b}_i|\mathbf{b}_{i+1}] \cdot \mathbf{A}^{-1} \cdot \mathbf{A}'$
       (where $\mathbf{A} = [\mathbf{a}_i|\mathbf{a}_{i+1}]$ and $\mathbf{A}' = [\mathbf{a}'_i|\mathbf{a}'_{i+1}]$).
7: end while
8: return $((I_i, \mathbf{b}_i))_{i\le n}$.
```

Proof. Using Lemma 2.2 applied to the rank-2 module M_i, we have

$$\mathcal{N}(\mathbf{s}_i) \le {\gamma_{\mathcal{N}}}^d\, 2^{d/2} \Delta_K^{1/2} \big(\mathcal{N}(r_{i,i}I_i)\mathcal{N}(r_{i+1,i+1}I_{i+1})\big)^{1/2}.$$

Using the assumption allows to complete the proof. □

We are now ready to prove the main result of this section.

Theorem 3.4. *Assume that Step 3 of Algorithm 3.1 is implemented with some algorithm $\mathcal{O}$ for some parameter $\gamma_{\mathcal{N}}$. Assume that $\alpha_K > {\gamma_{\mathcal{N}}}^{2d} 2^d \Delta_K$. Then Algorithm 3.1 terminates and outputs an LLL-reduced pseudo-basis of M. Further, the number of loop iterations is bounded by*

$$\frac{n(n+1)}{\log(\alpha_K/({\gamma_{\mathcal{N}}}^{2d} 2^d \Delta_K))} \cdot \log \frac{\max \mathcal{N}(r_{ii}I_i)}{\min \mathcal{N}(r_{ii}I_i)},$$

where the I_i's and r_{ii}'s are those of the input pseudo-basis.

Proof. We first show that at every stage of the algorithm, the current pseudo-basis $((I_i, \mathbf{b}_i))_{i\le n}$ is a pseudo-basis of M. For this, it suffices to show that the operations performed on it at Step 6 preserve this property. This is provided by the fact that $\mathbf{A}^{-1} \cdot \mathbf{A}'$ maps the pseudo-basis $((I_i, \mathbf{a}_i), (I_{i+1}, \mathbf{a}_{i+1}))$ into the pseudo-basis $((I'_i, \mathbf{a}'_i), (I'_{i+1}, \mathbf{a}'_{i+1}))$ of the same rank-2 module (by Lemma 2.8). Applying the same transformation to $((I_i, \mathbf{b}_i), (I_{i+1}, \mathbf{b}_{i+1}))$ preserves the spanned rank-2 module. The correctness of Algorithm 3.1 is implied by termination and the above.

We now prove a bound on the number of loop iterations, which will in particular imply termination. Consider the quantity

$$\Pi := \prod_{i\le n} \mathcal{N}(r_{ii}I_i)^{n-i+1}.$$

This quantity if bounded from above by $\max \mathcal{N}(r_{ii}I_i)^{n(n+1)/2}$ and from below by $\min \mathcal{N}(r_{ii}I_i)^{n(n+1)/2}$. Below, we show that Π never increases during the execution of the algorithm, and that at every iteration of the while loop, it decreases by a factor $\ge \sqrt{\alpha_K/({\gamma_{\mathcal{N}}}^{2d} 2^d \Delta_K)}$. We also show that the quantity $\min \mathcal{N}(r_{ii}I_i)^{n(n+1)/2}$

can only increase during the execution of the algorithm, hence the lower bound above holds with respect to the input r_{ii} and I_i at every step of the algorithm. Combining the decrease rate with the above upper and lower bounds, this implies that the number of loop iterations is bounded by

$$\frac{n(n+1)}{\log(\alpha_K/({\gamma_{\mathcal{N}}}^{2d} 2^d \Delta_K))} \cdot \log \frac{\max \mathcal{N}(r_{ii}I_i)}{\min \mathcal{N}(r_{ii}I_i)},$$

where the I_i's and r_{ii}'s are those of the input pseudo-basis.

Consider an iteration of the while loop, working at index i. We have $\alpha_K \cdot \mathcal{N}(r_{i+1,i+1}I_{i+1}) < \mathcal{N}(r_{i,i}I_i)$. Step 6 is the only one that may change Π. Observe that we have

$$\Pi = \prod_{j \le n} \mathcal{N}\left(\det_{K_{\mathbb{R}}}\left(((I_i, \mathbf{b}_i))_{i \le j}\right)\right).$$

During the loop iteration, none of the n modules in the expression above changes, except possibly the i-th one. Now, note that

$$\mathcal{N}\left(\det_{K_{\mathbb{R}}}\left(((I_k, \mathbf{b}_k))_{k \le i}\right)\right) = \prod_{k \le i} \mathcal{N}(r_{kk}I_k).$$

During the loop iteration under scope, only the i-th term in this product may change. At Step 6, it is updated from $\mathcal{N}(r_{ii}I_i)$ to $\mathcal{N}(I'_i)\mathcal{N}(\mathbf{a}'_i)$. By Lemma 2.8, we have $\mathcal{N}(I'_i) \le 1$ and $\mathbf{a}'_i = \mathbf{s}_i$. Now, by Lemma 3.3, we have that $\mathcal{N}(\mathbf{s}_i) \le {\gamma_{\mathcal{N}}}^d \sqrt{\frac{2^d \Delta_K}{\alpha_K}} \mathcal{N}(r_{ii}I_i)$. Overall, this gives that $\mathcal{N}(r_{ii}I_i)$ and hence Π decrease by a factor $\ge \sqrt{\alpha_K/({\gamma_{\mathcal{N}}}^{2d} 2^d \Delta_K)}$.

To show that $\min \mathcal{N}(r_{ii}I_i)$ can only increase during the execution of the algorithm, observe that, during a loop iteration, only $\mathcal{N}(r_{ii}I_i)$ and $\mathcal{N}(r_{i+1,i+1}I_{i+1})$ may be modified. Let us call $\mathcal{N}(r'_{ii}I'_i)$ and $\mathcal{N}(r'_{i+1,i+1}I'_{i+1})$ the corresponding values at the end of the iteration. We have seen above that $\mathcal{N}(r'_{ii}I'_i) \le \mathcal{N}(r_{ii}I_i)$, which implies that $\mathcal{N}(r'_{ii}I'_i) \le \max(\mathcal{N}(r_{ii}I_i), \mathcal{N}(r_{i+1,i+1}I_{i+1}))$. We also know from Lemma 2.6 that $\mathcal{N}(r'_{ii}I'_i) \ge \min(\mathcal{N}(r_{ii}I_i), \mathcal{N}(r_{i+1,i+1}I_{i+1}))$. As the determinant of M_i is constant, we have

$$\mathcal{N}(r'_{ii}I'_i) \cdot \mathcal{N}(r'_{i+1,i+1}I'_{i+1}) = \mathcal{N}(r_{ii}I_i) \cdot \mathcal{N}(r_{i+1,i+1}I_{i+1}).$$

This implies that $\mathcal{N}(r'_{i+1,i+1}I'_{i+1}) \ge \min(\mathcal{N}(r_{ii}I_i), \mathcal{N}(r_{i+1,i+1}I_{i+1}))$. Overall, we have that $\mathcal{N}(r'_{ii}I'_i), \mathcal{N}(r'_{i+1,i+1}I'_{i+1}) \ge \min(\mathcal{N}(r_{ii}I_i), \mathcal{N}(r_{i+1,i+1}I_{i+1}))$. $\square$

3.2 Handling bit-lengths

In terms of bit-lengths of the diverse quantities manipulated during the execution of the algorithm, there can be several sources of bit-length growth. Like in the classical LLL-algorithm, the Euclidean norms of off-diagonal coefficients r_{ij} for $i < j$ could grow during the execution. We handle this using a generalized size-reduction algorithm. Other annoyances are specific to the number field setup. There is too much freedom in representing a rank-1 module $I\mathbf{v}$: scaling the ideal I by some $x \in K$ and dividing $\mathbf{v}$ by the same x preserves the module. In the extreme case, it could cost an arbitrarily large amount of space, even to store a trivial rank-1 module such as $R \cdot (1, 0, \dots, 0)^T$, if such a bad scaling is used (e.g., using such an x with large algebraic norm). Finally, even if the ideal I is "scaled", we can still multiply $\mathbf{v}$ by a unit: this preserves the rank-1 module, but makes its representation longer.[8]

Definition 3.5. *A pseudo-basis $((I_i, \mathbf{b}_i))_{i\leq n}$, with $I_i \subset K$ and $\mathbf{b}_i \in K_{\mathbb{R}}^m$ for all $i \leq n$, is said scaled if, for all $i \leq n$,*

$$R \subseteq I_i, \quad \mathcal{N}(I_i) \geq 2^{-d^2} \Delta_K^{-1/2} \text{ and } \|r_{ii}\| \leq 2^d \Delta_K^{1/(2d)} \mathcal{N}(r_{ii} I_i)^{1/d}.$$

It is said size-reduced if $\|r_{ij}/r_{ii}\| \leq (4d)^d \Delta_K^{1/2}$ for all $i < j \leq n$.

Note that if $((I_i, \mathbf{b}_i))_{i\leq n}$ is scaled, then $\mathcal{N}(I_i) \leq 1$ for all $i \leq n$. Further, if the spanned module is contained in R^m, then $\mathbf{b}_i \in R^m$ for all $i \leq n$. Algorithm 3.2 transforms any pseudo-basis into a scaled pseudo-basis of the same module.

Algorithm 3.2 Scaling the ideals.

```
Input: A pseudo-basis $((I_i, \mathbf{b}_i))_{i\leq n}$ of a module $M$.
Output: A scaled pseudo-basis $((I'_i, \mathbf{b}'_i))_{i\leq n}$ of $M$.
1: for $i = 1$ to $n$ do
2:     Use LLL to find $s_i \in r_{ii} \cdot I_i \setminus \{0\}$ such that $\|s_i\| \leq 2^d \Delta_K^{1/(2d)} \mathcal{N}(r_{ii} I_i)^{1/d}$;
3:     Write $s_i = r_{ii} \cdot x_i$, with $x_i \in I_i$;
4:     Define $I'_i = I_i \cdot \langle x_i \rangle^{-1}$ and $\mathbf{b}'_i = x_i \mathbf{b}_i$.
5: end for
6: return $((I'_i, \mathbf{b}'_i))_{i\leq n}$.
```

Lemma 3.6. *Algorithm 3.2 outputs a scaled pseudo-basis of the module M generated by the input pseudo-basis and preserves the $\mathcal{N}(r_{ii} I_i)$'s. If $M \subseteq K^m$, then it runs in time polynomial in $\log \Delta_K$ and the input bit-length.*

Proof. The algorithm scales each column of the pseudo-matrix by some factor $x_i \in K$ and scales the corresponding ideal accordingly. This operation preserves the spanned module. Further, the fact that x_i belongs to K implies that the ideal I'_i remains a fractional ideal of K.

Fix $i \leq n$. The determinant of the canonical embedding of $r_{ii} I_i$ is $\Delta_K^{1/2} \mathcal{N}(r_{ii} I_i)$ and its dimension is d, so LLL can indeed be used to find $s_i \in r_{ii} \cdot I_i \setminus \{0\}$ such that $\|s_i\| \leq 2^d \Delta_K^{1/(2d)} \mathcal{N}(r_{ii} I_i)^{1/d}$. By the arithmetic-geometric inequality, this implies that $\mathcal{N}(s_i) \leq 2^{d^2} \Delta_K^{1/2} \mathcal{N}(r_{ii} I_i)$. We hence have $\mathcal{N}(x_i) \leq 2^{d^2} \Delta_K^{1/2} \mathcal{N}(I_i)$ and $\mathcal{N}(I'_i) = \mathcal{N}(I_i)/\mathcal{N}(x_i) \geq 2^{-d^2} \Delta_K^{-1/2}$. Further, as $x_i \in I_i$, we have $R \subseteq I'_i$, which implies that $\mathcal{N}(I'_i) \leq 1$. Finally, we have $r'_{ii} = x_i r_{ii} = s_i$, hence $\|r'_{ii}\| = \|s_i\| \leq 2^d \Delta_K^{1/(2d)} \mathcal{N}(r_{ii} I_i)^{1/d} = 2^d \Delta_K^{1/(2d)} \mathcal{N}(r'_{ii} I'_i)^{1/d}$. This proves the bound on $\|r'_{ii}\|$ and concludes the proof of correctness of the algorithm.

Now, observe that $\mathbf{b}'_i$ is $K_{\mathbb{R}}$-colinear to $\mathbf{b}_i$. Hence, replacing a vector $\mathbf{b}_i$ by $\mathbf{b}'_i$ does not impact r_{jj} for $j \neq i$. The quantities $\mathcal{N}(r_{jj} I_j)$ are therefore preserved through the execution of the algorithm.

If $M \subseteq K^m$, by Lemmas 2.3, 2.4 and 2.7, all the operations performed in the algorithm can be done in polynomial time. Hence the whole algorithm runs in polynomial time. ☐

[8] Note that ideal scaling and size-reduction have been suggested in [20, Se. 4.1], but without a complexity analysis (polynomial complexity was claimed but not proved).

Algorithm 3.3 aims at size-reducing a scaled pseudo-basis. It relies on a $\lfloor\cdot\rceil_R$ operator which takes as input a $y \in K_{\mathbb{R}}$ and rounds it to some $k \in R$ by writing $y = \sum y_i r_i$ for some y_i's in $\mathbb{R}$, and rounding each y_i to the nearest integer: $k = \sum k_i r_i = \sum \lfloor y_i \rceil r_i$ (remember that the r_i's form an LLL-reduced basis of R). For computations, we will apply this operator numerically, so that we may not have $\max_i |k_i - y_i| \leq 1/2$ but, with a bounded precision computation, we can ensure that $\max_i |k_i - y_i| \leq 1$.

Algorithm 3.3 Size-reduction.

```
Input: A scaled pseudo-basis $((I_i, \mathbf{b}_i))_{i\leq n}$ of a module $M$.
Output: A size-reduced pseudo-basis of $M$.
1: for $j = 1$ to $n$ do
2:    for $i = j-1$ to $1$ do
3:       Compute $x_i = \lfloor r_{ij}/r_{ii} \rceil_R$;
4:       $\mathbf{b}_j := \mathbf{b}_j - x_i \mathbf{b}_i$.
5:    end for
6: end for
7: return $((I_i, \mathbf{b}_i))_{i\leq n}$.
```

Lemma 3.7. *Algorithm 3.3 outputs a scaled size-reduced pseudo-basis of the module M generated by the input pseudo-basis and preserves the $\mathcal{N}(r_{ii}I_i)$'s. If $M \subseteq K^m$, then it runs in time polynomial in $\log \Delta_K$ and the input bit-length.*

Proof. As the input basis is scaled, the ideals I_i contain R and thus the operations performed on the pseudo-basis preserve the spanned module. Further, note that the update of $\mathbf{b}_j$ at Step 4 has no effect on $r_{i'j'}$ for $j' \neq j$ or $j' = j$ and $i' > i$. It transforms r_{ij} into $r_{ij} - x_i r_{ii}$ and $r_{i'j}$ into $r_{i'j} - x_i r_{i'i}$ for $i' < i$. In particular, the new r_{ij} satisfies $\|r_{ij}/r_{ii}\| \leq d \max_{k\leq d} \|r_k\|$. As the r_k's are a LLL-reduced basis of R, we have that $\max_{k\leq d} \|r_k\| \leq (4d)^{d/2} \Delta_K^{1/2}$. This proves the correctness of Algorithm 3.3. The preservation of the $\mathcal{N}(r_{ii}I_i)$'s is direct, as neither the r_{ii}'s nor the I_i's are modified.

Now, assume that $M \subseteq K^m$. Without loss of generality, we can assume that the matrix $\mathbf{B}$, whose columns are the $\mathbf{b}_j$, is in $R^{m\times n}$ (we can always multiply $\mathbf{B}$ by the smallest $x \in \mathbb{Z}_{>0}$ such that $x\mathbf{B} \subseteq R^m$ and divide everything by x at the end of the execution of the algorithm; the bit-length of $x\mathbf{B}$ is polynomially bounded by the one of $\mathbf{B}$). Assume that the outer loop is currently at index j. Consider the inner loop iteration indexed by i. Let m_j^{old} and m_j^{new} respectively denote the value of $\max_{i'<j} \|r_{i'j}/r_{i'i'}\|$ at the start and end of this inner loop iteration. We have:

$$m_j^{new} \leq m_j^{old} + \|x_i\| \cdot \max_{i'<j} \|r_{i'i}/r_{i'i'}\|.$$

We have $\max_{i'<j} \|r_{i'i}/r_{i'i'}\| \leq (4d)^d \Delta_K^{1/2}$, because the first $j-1$ columns are size-reduced. Also, we have $\|x_i\| \leq m_j^{old} + (4d)^d \Delta_K^{1/2}$, as $x_i = \lfloor r_{ij}/r_{ii} \rceil_R$. This gives

$$m_j^{new} \leq (1 + (4d)^d \Delta_K^{1/2}) m_j^{old} + (4d)^{2d} \Delta_K.$$

Iterating over the $j - 1 \leq n$ values of i, we obtain that m_j always stays bounded from above by $(1 + (4d)^d \Delta_K^{1/2})^n (m_j^{init} + (4d)^d \Delta_K^{1/2})$, where m_j^{init} is the value of

$\max_{i'<j} \|r_{i'j}/r_{i'i'}\|$ at the start of the first inner loop iteration. This implies that $\log \|\mathbf{b}_j\|$ always remains below a polynomial in the input size, n and $\log \Delta_K$. Since the bit-length of $\mathbf{b}_j$ is polynomially bounded by $\log \|\mathbf{b}_j\|$ (because $\mathbf{b}_j \in R^m$), we conclude that the bit-length of the pseudo-basis remains polynomially bounded in $\log \Delta_K$ and the input bit-length during the execution of the algorithm. If need be, we can recompute the R-factor at every step of the algorithm (rather than updating it), and, by Lemma 2.7, the cost will still be polynomially bounded in $\log \Delta_K$ and the input bit-length. □

We now consider Algorithm 3.4, which is a variant of Algorithm 3.1 that allows us to prove a bound on the bit cost. The only difference (Step 7) is that we call Algorithms 3.2 and 3.3 at every loop iteration of Algorithm 3.1, so that we are able to master the bit-lengths during the execution. Without loss of generality, we can assume that the pseudo-basis given as input is scaled and size-reduced: if it is not the case, we can call Algorithms 3.2 and 3.3, which will produce a pseudo-basis of the same module, whose bit-length is polynomial in $\log \Delta_K$ and the input bit-length.

Algorithm 3.4 LLL-reduction over K with controlled bit-lengths

```
Input: A scaled size-reduced pseudo-basis $((I_i, \mathbf{b}_i))_{i \le n}$ of a module $M \subseteq K^m$.
Output: An LLL-reduced pseudo-basis of $M$.
1: while there exists $i < n$ such that $\alpha_K \cdot \mathcal{N}(r_{i+1,i+1} I_{i+1}) < \mathcal{N}(r_{i,i} I_i)$ do
2:     Let $M_i$ be the rank-2 module spanned by the pseudo-basis $((I_i, \mathbf{a}_i), (I_{i+1}, \mathbf{a}_{i+1}))$, with $\mathbf{a}_i = (r_{ii}, 0)^T$ and $\mathbf{a}_{i+1} = (r_{i,i+1}, r_{i+1,i+1})^T$;
3:     Find $\mathbf{s}_i \in M_i \setminus \{\mathbf{0}\}$ such that $\mathcal{N}(\mathbf{s}_i) \le {\gamma_{\mathcal{N}}}^d \cdot \lambda_1^{\mathcal{N}}(M_i)$;
4:     Set $\mathbf{s}_{i+1} = \mathbf{a}_i$ if it is linearly independent with $\mathbf{s}_i$, and $\mathbf{s}_{i+1} = \mathbf{a}_{i+1}$ otherwise;
5:     Call the algorithm of Lemma 2.8 with $((I_i, \mathbf{a}_i), (I_{i+1}, \mathbf{a}_{i+1}))$ and $(\mathbf{s}_i, \mathbf{s}_{i+1})$ as inputs, and let $((I'_i, \mathbf{a}'_i), (I'_{i+1}, \mathbf{a}'_{i+1}))$ denote the output;
6:     Update $I_i := I'_i$, $I_{i+1} := I'_{i+1}$ and $[\mathbf{b}_i|\mathbf{b}_{i+1}] := [\mathbf{b}_i|\mathbf{b}_{i+1}] \cdot \mathbf{A}^{-1} \cdot \mathbf{A}'$
       (where $\mathbf{A} = [\mathbf{a}_i|\mathbf{a}_{i+1}]$ and $\mathbf{A}' = [\mathbf{a}'_i|\mathbf{a}'_{i+1}]$);
7:     Update the current pseudo-basis by applying Algorithm 3.2 and then Algorithm 3.3 to it.
8: end while
9: return $((I_i, \mathbf{b}_i))_{i \le n}$.
```

Theorem 3.8. *Assume that Step 3 of Algorithm 3.4 is implemented with some algorithm $\mathcal{O}$ for some parameter $\gamma_{\mathcal{N}}$. Assume that $\alpha_K > {\gamma_{\mathcal{N}}}^{2d} 2^d \Delta_K$. Given as input a scaled and size-reduced pseudo-basis of a module $M \subseteq K^m$, Algorithm 3.4 outputs an* LLL*-reduced pseudo-basis of M in time polynomial in* $\log \Delta_K$, $1/\log(\alpha_K/({\gamma_{\mathcal{N}}}^{2d} 2^d \Delta_K))$ *and the input bit-length.*

Proof. We assume without loss of generality that $M \subseteq R^m$ (we can multiply M by the smallest $x \in \mathbb{Z}_{>0}$ such that $xM \subseteq R^m$ and the bit-length of xM is polynomially bounded by the one of M). The correctness proof of Theorem 3.4 still holds. The only adaptation needed is to observe that during the execution of Step 7, none of the $\mathcal{N}(r_{ii} I_i)$'s changes. This is provided by Lemmas 3.6 and 3.7. Further, note that the bound on the number of loop iterations is polynomial in the bit-length of the input pseudo-basis and $1/\log(\alpha_K/({\gamma_{\mathcal{N}}}^{2d} 2^d \Delta_K))$. It remains to prove that the bit-lengths of all the quantities occurring during the execution of the algorithm remain sufficiently small.

For this, it suffices to show that the pseudo-bases keep bounded bit-lengths. As Algorithms 3.2, 3.3 and the algorithm from Lemma 2.8 run in polynomial time, the bit-lengths of all quantities manipulated during a loop iteration are polynomially bounded in terms of the run-time of $\mathcal{O}$, $\log \Delta_K$ and the bit-length of the pseudo-basis at the start of the same loop iteration. It therefore suffices to bound the bit-lengths of the pseudo-bases occurring at the start of each loop iteration. At that moment, the pseudo-bases are scaled, so the bit-lengths of the coefficient ideals are polynomial in $\log \Delta_K$. We now focus on the vectors $\mathbf{b}_j$. Observe that since we have $M \subseteq R^m$ and the pseudo-bases are scaled, the vectors $\mathbf{b}_j$ belong to R^m (so that their bit-lengths can be bounded by a polynomial in $\log \|\mathbf{b}_j\|$).

Note that $\|\mathbf{b}_j\|^2 = \sum_{i \leq j} \|r_{ij}\|^2$. By size-reducedness, and using the fact that $\|r_{ij}\| \leq \|r_{ij}/r_{ii}\| \cdot \|r_{ii}\|$, we have that $\|\mathbf{b}_j\| \leq \sqrt{d}(4d)^d \Delta_K^{1/2} \max_i \|r_{ii}\|$. As the pseudo-basis is scaled, we have that $\|\mathbf{b}_j\| \leq \sqrt{d} \cdot (8d)^d \cdot \Delta_K \cdot \max_i \mathcal{N}(r_{ii} I_i)^{1/d}$. Now, note that $\max_i \mathcal{N}(r_{ii} I_i)^{1/d}$ never increases during the execution of the algorithm: this is implied by the part of the proof of Theorem 3.4 involving Π. Initially, it is bounded by a polynomial in the input bit-length. Overall, we obtain that at every start of an iteration of the while loop, the quantity $\log \|\mathbf{b}_j\|$ is bounded by a polynomial in $\log \Delta_K$ and the input bit-length. Using Lemma 2.3 allows to complete the proof. □

3.3 Finding short vectors for the Euclidean norm

By Lemma 3.2 and Theorem 3.8 with $\alpha_K = (1 + c/n) \cdot {\gamma_{\mathcal{N}}}^{2d} 2^d \Delta_K$ for a well-chosen constant c, Algorithm 3.4 may be interpreted as a reduction from finding a $2 \cdot ({\gamma_{\mathcal{N}}}^{2d} 2^d \Delta_K)^n$ approximation to a vector reaching $\lambda_1^{\mathcal{N}}$ in rank-n modules, to finding a ${\gamma_{\mathcal{N}}}^d$ approximation to a vector reaching $\lambda_1^{\mathcal{N}}$ in rank-2 modules.

By using Lemma 2.2, we can extend the above to the Euclidean norm instead of the algebraic norm.

Theorem 3.9. *Let $\gamma \geq 1$ and assume that an LLL-reduced $\mathbb{Z}$-basis of R is known. Then there exists a reduction from solving SVP$_{\gamma'}$ in rank-n modules (with respect to $\|\cdot\|$) in K^n to solving SVP$_\gamma$ in rank-2 modules in K^2, where $\gamma' = (2\gamma \Delta_K^{1/d})^{2n-1}$. This reduction runs in time polynomial in $\log \Delta_K$ and the bit-length of the input pseudo-basis.*

Proof. The reduction consists in first using Algorithm 3.4 with Step 3 implemented using the oracle solving SVP$_\gamma$ in rank-2 modules. Using the arithmetic-geometric inequality and Lemma 2.2, one can see that a vector $\mathbf{s}$ satisfying $\|\mathbf{s}\| \leq \gamma \cdot \lambda_1(M)$ also satisfies $\mathcal{N}(\mathbf{s}) \leq \gamma^d \cdot \Delta_K^{1/2} \cdot \lambda_1^{\mathcal{N}}(M)$. Hence, we have an oracle computing a ${\gamma_{\mathcal{N}}}^d = (\gamma \cdot \Delta_K^{1/(2d)})^d$ approximation of $\lambda_1^{\mathcal{N}}(M)$. We then run Algorithm 3.4 with this oracle by setting the parameter α_K to $(1 + \frac{\log 2}{n}) \cdot {\gamma_{\mathcal{N}}}^{2d} 2^d \Delta_K$. By Theorem 3.8, the reduction runs in time polynomial in the input bit-length, $\log \Delta_K$ and $1/\log(\alpha_K / ({\gamma_{\mathcal{N}}}^{2d} 2^d \Delta_K))$. By choice of α_K, the latter is polynomial in n, which is itself bounded from above by the bit-length of the input pseudo-basis. Further, by Lemma 3.2, the output pseudo-basis satisfies $\mathcal{N}(I_1)\mathcal{N}(\mathbf{b}_1) \leq \alpha_K^{n-1} \cdot \lambda_1^{\mathcal{N}}(M)$. By Lemma 2.2 and by definition of α_K and $\gamma_{\mathcal{N}}$, this gives:

$$\mathcal{N}(I_1)\mathcal{N}(\mathbf{b}_1) \leq 2(\gamma^{2d} 2^d \Delta_K^2)^{n-1} \cdot d^{-d/2} \lambda_1(M)^d.$$

Now, an SVP_γ solver for rank-2 modules directly provides an SVP_γ solver for rank-1 module. We hence use our oracle again, on $I_1\mathbf{b}_1$. This provides a non-zero vector $\mathbf{s} \in I_1\mathbf{b}_1 \subseteq M$ such that $\|\mathbf{s}\| \leq \gamma\sqrt{d}\Delta_K^{1/(2d)} \cdot (\mathcal{N}(I_1)\mathcal{N}(\mathbf{b}_1))^{1/d}$, by Minkowski's theorem. Combining the latter with the above upper bound on $\mathcal{N}(I_1)\mathcal{N}(\mathbf{b}_1)$ provides the result. $\square$

4 The divide-and-swap algorithm

We now focus on how to implement Step 3 of Algorithm 3.1, using a CVP oracle for a lattice depending on K only. To handle projected 2-dimensional lattices, the LLL algorithm for integer lattices proceeds like the Gauss-Lagrange reduction algorithm for 2-dimensional lattices. It relies on a divide-and-swap elementary procedure: first shorten the second vector using a $\mathbb{Z}$-multiple of the first one (using a Euclidean division, or, more pedantically, a CVP solver for the trivial lattice $\mathbb{Z}$); then swap these two vectors if the second has become (significantly) shorter than the first one. It has the effect that if this 2-dimensional basis is not reduced, then a swap occurs, and some progress is made towards reducedness of the 2-dimensional basis. This elementary step is repeated as many times as needed to achieve reduction of the lattice under scope. In this section, we generalize this process to rank-2 modules and obtain a Gauss-Lagrange algorithm for modules.

Theorem 4.1 (Heuristic). *For any sequence of number fields K and any $\eta > 0$, there exist a sequence of lattices L_K of dimension $O((d\log\rho(R))^{2+\eta})$, an approximation factor $\gamma_{\mathcal{N}} = 2^{\widetilde{O}(d\log\rho(R))/d}$ and a quantum algorithm $\mathcal{A}$ such that (under Heuristic 1 and the heuristics of Lemma 2.5):*

- *Algorithm $\mathcal{A}$ takes as input a pseudo-basis of a rank-2 module $M \subset K_{\mathbb{R}}^2$, and outputs a vector $\mathbf{v} \in M$ such that $\mathcal{N}(\mathbf{v}) \leq {\gamma_{\mathcal{N}}}^d \lambda_1^{\mathcal{N}}(M)$;*
- *when restricted to modules contained in K^2, Algorithm $\mathcal{A}$ makes a number of queries to an oracle solving the closest vector problem in L_K and requires a total number of quantum operations that are both polynomial in $\log\Delta_K$ and the input bit-length.*

Moreover, the lattice L_K and the algorithm $\mathcal{A}$ can both be quantumly computed from K and η, in time polynomial in $\log\Delta_K$.

The main difference with the case of lattices over $\mathbb{Z}$ is that, for our algorithm to run in quantum polynomial time, we need an oracle solving CVP in a fixed lattice L_K. This is in a sense a generalization of the Euclidean division, which over $\mathbb{Z}$ can be performed in classical polynomial time by solving a CVP instance in the lattice $\mathbb{Z}$. In an arbitrary (non-Euclidean) number field, this "Euclidean division" requires solving CVP in a more sophisticated lattice, still depending only on the number field K.

We observe that the quantity $d\log\rho(R)$ appearing in the bounds of the dimension of L_K and of the approximation factor $\gamma_{\mathcal{N}}$ is $\widetilde{\Theta}(\log\Delta)$ for many number fields of interest (e.g., cyclotomic fields and fields of constant degree d). We are however not aware of any result which would prove that this is the case for all number fields (nor are we aware of any counter-example).

Combining the algorithm for rank-2 modules with the reduction from rank-n to rank-2 modules given in Section 3, we obtain a full LLL algorithm for module lattices.

Corollary 4.2 (Heuristic). *For any sequence of number fields K and any $\eta > 0$, there exist a sequence of lattices L_K of dimension $O((d \log \rho(R))^{2+\eta})$, an approximation factor $\gamma = 2^{\widetilde{O}(d \log \rho(R))/d}$ and a quantum algorithm $\mathcal{A}$ such that (under Heuristic 1 and the heuristics of Lemma 2.5):*

- *Algorithm $\mathcal{A}$ takes as input a pseudo-basis of a rank-n module $M \subset K_{\mathbb{R}}^m$, and outputs a vector $\mathbf{v} \in M$ such that $\|\mathbf{v}\|_\infty \leq \gamma^n \cdot \lambda_1(M)$;*
- *when restricted to modules contained in K^m, Algorithm $\mathcal{A}$ makes a number of queries to an oracle solving the closest vector problem in L_K and requires a total number of quantum operations that are both polynomial in $\log \Delta_K$ and the input bit-length.*

Moreover, the lattice L_K and the algorithm $\mathcal{A}$ can both be quantumly computed from K and η, in time polynomial in $\log \Delta_K$.

Below, we first describe a lattice $L_{K,2}$ that depends only on K and for which we will assume that we possess a CVP oracle. Then, we give an algorithm whose objective is to act as a Euclidean algorithm, i.e., enabling us to shorten an element of $K_{\mathbb{R}}$ using R-multiples of another. Once we have this generalization of the Euclidean algorithm, we finally describe a divide-and-swap algorithm for rank-2 modules.

4.1 Extending the logarithm

The lattice $L_{K,2}$ is defined using (among others) the log-unit lattice Λ. However, the Log function does not suffice for our needs. In particular, for $a, b \in K_{\mathbb{R}}^\times$, the closeness between a and b is not necessarily implied by the closeness of $\operatorname{Log} a$ and $\operatorname{Log} b$, because Log does not take into account the complex arguments of the entries of the canonical embeddings of a and b. However, we will need such a property to hold. For this purpose, we hence extend the Log function. For $x \in K_{\mathbb{R}}^\times$, we define $\overline{\operatorname{Log}}\, x := (\theta_1, \dots, \theta_{r_1+r_2}, \log|\sigma_1(x)|, \dots, \log|\sigma_d(x)|)^T$, where $\sigma_i(x) = |\sigma_i(x)| \cdot \mathrm{e}^{\mathrm{I}\theta_i}$ for all $i \leq r_1 + r_2$ and I is a complex root of x^2+1. The $\overline{\operatorname{Log}}$ function takes values in $(\pi\mathbb{Z}/2\pi\mathbb{Z})^{r_1} \times (\mathbb{R}/(2\pi\mathbb{Z}))^{r_2} \times \mathbb{R}^d$.

Lemma 4.3. *For $x, y \in K_{\mathbb{R}}^\times$, we have*

$$\|x - y\|_\infty \leq \left(\mathrm{e}^{\sqrt{2}\|\overline{\operatorname{Log}}\, x - \overline{\operatorname{Log}}\, y\|_\infty} - 1\right) \cdot \min(\|x\|_\infty, \|y\|_\infty).$$

Observe that for $t \leq (\log 2)/\sqrt{2}$, we have $\mathrm{e}^{\sqrt{2}t} - 1 \leq 2\sqrt{2}t$.

Proof. Let us write

$$\begin{aligned}\overline{\operatorname{Log}}\, x &= (\theta_1, \dots, \theta_{r_1+r_2}, \log|\sigma_1(x)|, \dots, \log|\sigma_{r_1+r_2}(x)|)^T \\ \overline{\operatorname{Log}}\, y &= (\theta_1', \dots, \theta_{r_1+r_2}', \log|\sigma_1(y)|, \dots, \log|\sigma_{r_1+r_2}(y)|)^T.\end{aligned}$$

Let $i \leq r_1+r_2$ and define $z_x = \log|\sigma_i(x)|+\mathrm{I}\theta_i$ and $z_y = \log|\sigma_i(y)|+\mathrm{I}\theta'_i$. By definition of $\overline{\mathrm{Log}}$, we have $\sigma_i(x) = \mathrm{e}^{z_x}$ and $\sigma_i(y) = \mathrm{e}^{z_y}$. Therefore, we can write

$$\frac{|\sigma_i(x)-\sigma_i(y)|}{|\sigma_i(x)|} = |1-\mathrm{e}^{z_y-z_x}| = \Big|\sum_{k\geq 1}\frac{(z_y-z_x)^k}{k!}\Big| \leq \sum_{k\geq 1}\frac{|z_y-z_x|^k}{k!} = \mathrm{e}^{|z_y-z_x|}-1.$$

As $|z_x - z_y| \leq \sqrt{2}\|\overline{\mathrm{Log}}\,x - \overline{\mathrm{Log}}\,y\|_\infty$, we derive that

$$|\sigma_i(x)-\sigma_i(y)| \leq \|x\|_\infty \cdot \left(\mathrm{e}^{\sqrt{2}\|\overline{\mathrm{Log}}\,x-\overline{\mathrm{Log}}\,y\|_\infty}-1\right).$$

Note that this holds for all $i \leq r_1+r_2$ and that we could as well have used $\|y\|_\infty$ rather than $\|x\|_\infty$. $\square$

4.2 The lattice $L_{K,2}$

Let $r = \mathrm{poly}(d)$ and $\beta > 0$ be some parameter to be chosen later. Let Λ denote the log-unit lattice. Let $\mathfrak{B}_0 = \{\mathfrak{p}_1,\dots,\mathfrak{p}_{r_0}\}$ be a set of cardinality $r_0 \leq 2\log h_K$ of prime ideals generating Cl_K, with algebraic norms $\leq \exp(\delta_0)$, with $\delta_0 = O(\log\log\Delta_K)$. We will also consider another set $\mathfrak{B} = \{\mathfrak{q}_1,\dots,\mathfrak{q}_r\}$ of cardinality r, containing prime ideals (not in $\mathfrak{B}_0$) of norms $\leq \exp(\delta)$, for some parameters r and $\delta \leq \delta_0$ to be chosen later. We also ask that among these ideals $\mathfrak{q}_j$, at least half of them have an algebraic norm $\geq \sqrt{\mathrm{e}^\delta}$. Because we want r such ideals, we should make sure that the number of prime ideals of norm bounded by e^δ in R is larger than r. This will asymptotically be satisfied if $r \leq O(\mathrm{e}^\delta/\delta)$ (by Lemma 2.1). The constraint that at least $r/2$ ideals should have norm larger than $\sqrt{\mathrm{e}^\delta}$ is not very limiting, as we expect that roughly $\mathrm{e}^\delta - \sqrt{\mathrm{e}^\delta} \geq r - \sqrt{r}$ ideals should have algebraic norm between $\sqrt{\mathrm{e}^\delta}$ and e^δ (forgetting about the $\mathrm{poly}(\delta)$ terms).

We now define $L_{K,2}$ as the lattice of dimension $\nu = 2(r_1+r_2)+r_0+r-1$ (included in $\mathbb{R}^{\nu+1}$) spanned by the columns of the following basis matrix:

$$\mathbf{B}_{L_{K,2}} := \begin{array}{|c|c|c|c|l}
\multicolumn{2}{c}{\overleftrightarrow{2(r_1+r_2)-1}} & \multicolumn{2}{c}{\overleftrightarrow{r_0+r}} & \\
\hline
\multicolumn{2}{|c|}{0} & \multicolumn{2}{c|}{\beta\cdot a_{g_{2(r_1+r_2)}} \cdots\ \beta\cdot a_{g_\nu}} & \updownarrow 1 \\
\hline
\begin{matrix}\beta\cdot 2\pi & & \\ & \ddots & \\ & & \beta\cdot 2\pi\end{matrix} & \multicolumn{3}{c|}{\beta\cdot\theta_{g_{r_1+r_2+1}} \quad \cdots \quad \beta\cdot\theta_{g_\nu}} & \updownarrow r_1+r_2 \\
\hline
0 & \beta\cdot\mathbf{B}_\Lambda & \multicolumn{2}{c|}{\beta\cdot\mathbf{h}_{g_{2(r_1+r_2)}} \cdots\ \beta\cdot\mathbf{h}_{g_\nu}} & \updownarrow r_1+r_2-1 \\
\hline
\multicolumn{2}{|c|}{0} & \mathbf{B}_{\Lambda_0} & \mathbf{w}_{\nu-r+1} \cdots\ \mathbf{w}_\nu & \updownarrow r_0 \\
\cline{3-4}
\multicolumn{2}{|c|}{} & 0 & \begin{matrix}1 & & \\ & \ddots & \\ & & 1\end{matrix} & \updownarrow r \\
\hline
\end{array}$$

where

- $\mathbf{B}_\Lambda$ is a basis of Λ, and we let $(\mathbf{h}_i)_{r_1+r_2<i<2(r_1+r_2)}$ denote its columns;
- $\mathbf{B}_{\Lambda_0}$ is a basis of the lattice $\Lambda_0 := \{(x_i)_i \in \mathbb{Z}^{r_0} : \prod_i \mathfrak{p}_i^{x_i} \text{ is principal}\}$, and we let $(\mathbf{w}_i)_{2(r_1+r_2)\leq i\leq \nu-r}$ denote its columns;
- for any $g \in K$, we have $a_g = (\log|\mathcal{N}(g)|)/\sqrt{d}$;
- for any $g \in K$, the vector θ_g consists of the first $r_1 + r_2$ entries of $\overline{\operatorname{Log}}(g)$;
- for any $g \in K$, we have $\mathbf{h}_g = i_{H\cap E}(\Pi_H(\operatorname{Log}(g)))$, where Π_H is the orthogonal projection on H and $i_{H\cap E}$ is an isometry mapping $H \cap E$ to $\mathbb{R}^{r_1+r_2-1}$;
- for any $i > r_1 + r_2$, if we parse the bottom $r_0 + r$ coordinates of the i-th column vector as $(w_{i,1}, \dots, w_{i,r_0}, w'_{i,1}, \dots, w'_{i,r})$, then we have that $\langle g_i\rangle = \prod_j \mathfrak{p}_j^{w_{ij}} \cdot \prod_j \mathfrak{q}_j^{w'_{ij}}$;
- the g_i's for $i > r_1 + r_2$ are in K and, among them, $g_{r_1+r_2+1}, \dots g_{2(r_1+r_2)-1}$ are the units of R corresponding to the columns of $\mathbf{B}_\Lambda$.

We now list a few properties satisfied by vectors in this lattice.

Lemma 4.4. *For every vector $(\beta a\|\beta\theta\|\beta\mathbf{h}\|\mathbf{w}\|\mathbf{w}') \in L_{K,2}$ (with blocks of dimensions $1, r_1+r_2, r_1+r_2-1, r_0$ and r), there exists $g \in K \setminus \{0\}$ with*

- $a = (\log|\mathcal{N}(g)|)/\sqrt{d}$
- $\overline{\operatorname{Log}}(g) = (\theta'\|\operatorname{Log}(g))$ *with* $\theta' = \theta \bmod 2\pi$.
- $\mathbf{h} = i_{H\cap E}(\Pi_H(\operatorname{Log}(g)))$
- $\langle g\rangle = \prod_j \mathfrak{p}_j^{w_j} \prod_j \mathfrak{q}_j^{w'_j}$, *where* $\mathbf{w} = (w_1, \cdots, w_{r_0})$ *and* $\mathbf{w}' = (w'_1, \cdots, w'_r)$.

Further, we have that $\|\operatorname{Log}(g)\|_2 = \|(a, \mathbf{h})\|_2$.

Proof. Note that the first claim holds for the vectors of the input basis (with $g = 1$ for the first $r_1 + r_2$ vectors). The property is preserved by vector addition (resp. subtraction), as can be seen by multiplying (resp. dividing) the corresponding g's. Hence it holds for all non-zero vectors of the lattice.

For the last statement, observe that $\operatorname{Log}(g) = \frac{\log|\mathcal{N}(g)|}{d}\mathbf{1} + \Pi_H(\operatorname{Log}(g))$. Hence, by orthogonality, we have:

$$\|\operatorname{Log}(g)\|_2^2 = \left(\frac{\log|\mathcal{N}(g)|}{d}\right)^2 \cdot d + \|\Pi_H(\operatorname{Log}(g))\|_2^2 = \|(a, \mathbf{h})\|_2^2.$$

This completes the proof. □

4.3 On the distance of relevant vectors to the lattice

In this section, we make a heuristic assumption on the distance between target vectors of a specific form and the lattice $L_{K,2}$ defined in the previous section. This heuristic is backed with a counting argument and numerical experiments (for the sake of readability, the description of the numerical experiments and their results have been postponed to Section 6). As $L_{K,2}$ is not full rank, we only consider target vectors $\mathbf{t}$ lying in the span of $L_{K,2}$. Also, as $\mathbf{B}_{L_{K,2}}$ contains the identity matrix in its bottom right corner, we cannot hope to have a covering radius that is much smaller than $\sqrt{r}$. In our case, the lattice dimension ν will be of the order of r, but in our application we will need a vector of $L_{K,2}$ much closer to $\mathbf{t}$ than $\sqrt{r} \approx \sqrt{\nu}$. In order to go below this value, we only consider target vectors $\mathbf{t}$ whose last r coordinates are almost 0.

Heuristic 1. Assume that there exist some integer $B \leq r$ such that $B \geq 200 \cdot (\log h_K) \cdot \delta_0/\delta$ and that

$$\alpha_0 \; := \sqrt{2\pi} \, \Big(\big(\frac{2B}{r^{0.96}}\big)^B \cdot \delta B(\det \Lambda) h_K \Big)^{1/d} \leq \frac{\log d}{12 d^{2.5}}.$$

Assume that the scaling parameter β in $\mathbf{B}_{L_{K,2}}$ is set to $\frac{1}{\alpha_0}\sqrt{\frac{0.01 \cdot B}{2d}}$. Then for any $\mathbf{t} \in \mathrm{Span}(L_{K,2})$ whose last r coordinates $\mathbf{w}'_t$ satisfy $\|\mathbf{w}'_t\|_2 \leq 0.01 \cdot B/\sqrt{r}$, we have $\mathrm{dist}(\mathbf{t}, L_{K,2}) \leq \sqrt{1.05 \cdot B}$.

Discussion about Heuristic 1. We provide below a counting argument to justify Heuristic 1. We consider the following set of vectors of $L_{K,2}$, parametrized by $B \leq r$, which we view as candidates for very close vectors to such target vectors:

$$S_B := \{\mathbf{s} = (\beta a_s \| \beta\theta_s \| \beta \mathbf{h}_s \| \mathbf{w}_s \| \mathbf{w}'_s) \in L_{K,2} : \mathbf{w}'_s \in \{-1, 0, 1\}^r \wedge \|\mathbf{w}'_s\|_1 = B\}.$$

We argue that there is a vector in S_B that is very close to $\mathbf{t}$. Our analysis is heuristic, but we justify it with both mathematical arguments and experiments. We are going to examine the vectors $\mathbf{s} \in S_B$ such that $\mathbf{s} - \mathbf{t}$ is reduced modulo $L_{K,2}$. Let us write $\mathbf{t} = (\beta a_t \| \beta\theta_t \| \beta\mathbf{h}_t \| \mathbf{w}_t \| \mathbf{w}'_t)^T$. We define:

$$\begin{aligned} S^{(1)}_{B,\mathbf{t}} := \{(\beta a_s \| \beta\theta_s \| \beta \mathbf{h}_s \| \mathbf{w}_s \| \mathbf{w}'_s) \in L_{K,2} : \; & \mathbf{w}'_s \in \{-1,0,1\}^r \wedge \|\mathbf{w}'_s\|_1 = B, \\ & \mathbf{w}_s - \lfloor \mathbf{w}_t \rceil \in \mathcal{V}(\Lambda_0), \\ & \mathbf{h}_t - \mathbf{h}_s \in \mathcal{V}(\Lambda), \\ & \theta_t - \theta_s \in (-\pi, \pi]^{r_1 + r_2}\}, \end{aligned}$$

where the notation $\mathcal{V}$ refers to the Voronoi cell (i.e., the set of points which are closer to 0 than to any other point of the lattice). The choice of $\mathbf{w}'_s$ fully determines $\mathbf{s} \in S^{(1)}_{B,\mathbf{t}}$, which gives the bound $|S^{(1)}_{B,\mathbf{t}}| = 2^B \cdot \binom{r}{B} \geq (2r/B)^B$.

We consider the following subset of $S^{(1)}_{B,\mathbf{t}}$:

$$S^{(2)}_{B,\mathbf{t}} = S^{(1)}_{B,\mathbf{t}} \cap \{(\beta a_s \| \beta\theta_s \| \beta \mathbf{h}_s \| \mathbf{w}_s \| \mathbf{w}'_s) \in L_{K,2} : \mathbf{w}_s = \lfloor \mathbf{w}_t \rceil\}.$$

We heuristically assume that when we sample a uniform vector in $S^{(1)}_{B,\mathbf{t}}$, the components $\mathbf{w}_s$ of the vectors $\mathbf{s} \in S^{(1)}_{B,\mathbf{t}}$ are uniformly distributed modulo Λ_0. Then the proportion of those for which $\mathbf{w}_s = \lfloor \mathbf{w}_t \rceil \bmod \Lambda_0$ is $1/\det(\Lambda_0) = 1/h_K$. Hence, we expect that $|S^{(2)}_{B,\mathbf{t}}| \approx |S^{(1)}_{B,\mathbf{t}}|/h_K$.

We consider the following subset of $S^{(2)}_{B,\mathbf{t}}$, parametrized by $\alpha \leq (\log d)/(12 d^{2.5})$:

$$S^{(3)}_{B,\alpha,\mathbf{t}} = S^{(2)}_{B,\mathbf{t}} \cap \{(\beta a_s \| \beta\theta_s \| \beta \mathbf{h}_s \| \mathbf{w}_s \| \mathbf{w}'_s) \in L_{K,2} : \|(\theta_s \| \mathbf{h}_s) - (\theta_t \| \mathbf{h}_t)\|_\infty \leq \alpha\}.$$

We heuristically assume that when we sample a uniform vector in $S^{(2)}_{B,\mathbf{t}}$, the components $(\theta_s, \mathbf{h}_s)$ are uniformly distributed modulo $2\pi\mathbb{Z}^{r_1+r_2} \times \Lambda$. Observe that the first r_1 coordinates of θ_s (corresponding to real embeddings) are either 0 or π. Hence, the probability that $\theta_s = \theta_t$ on these coordinates is 2^{-r_1}. Once these first r_1 coordinates are fixed, the remaining coordinates of $(\theta_s, \mathbf{h}_s)$ have no a priori reason to be bound to a sublattice of $2\pi\mathbb{Z}^{r_2} \times \Lambda$ and we heuristically assume them to be uniformly distributed in $\mathbb{R}^{r_1+2r_2-1}/(2\pi\mathbb{Z}^{r_2} \times \Lambda)$. Overall, the probability that a vector $\mathbf{s} \in S^{(2)}_{B,\mathbf{t}}$ satisfies $\|(\theta_s, \mathbf{h}_s) - (\theta_t, \mathbf{h}_t)\|_\infty \leq \alpha$ is $\approx \frac{\alpha^{r_1+2r_2-1}}{2^{r_1}\cdot(2\pi)^{r_2}\cdot\det(\Lambda)}$. Here, we used the fact that $\sqrt{r_1+2r_2-1}\cdot\alpha$ is smaller than $\lambda_1(2\pi\mathbb{Z}^{r_2} \times \Lambda)/2$ (recall from preliminaries that $\lambda_1^{(\infty)}(\Lambda) \geq (\log d)/(6d^2)$). We conclude that

$$|S^{(3)}_{B,\alpha,\mathbf{t}}| \approx |S^{(2)}_{B,\mathbf{t}}| \frac{\alpha^{r_1+2r_2-1}}{2^{r_1}\cdot(2\pi)^{r_2}\cdot\det(\Lambda)} \geq |S^{(2)}_{B,\mathbf{t}}| \frac{\alpha^{d-1}}{(2\pi)^{d/2}\cdot\det(\Lambda)}.$$

Finally, we consider the following subset of $S^{(3)}_{B,\alpha,\mathbf{t}}$:

$$S^{(4)}_{B,\alpha,\mathbf{t}} = S^{(3)}_{B,\alpha,\mathbf{t}} \cap \{(\beta a_s \| \beta\theta_s \| \beta\mathbf{h}_s \| \mathbf{w}_s \| \mathbf{w}'_s) \in L_{K,2} : |a_s - a_t| \leq \alpha\}.$$

We want a lower bound for $|S^{(4)}_{B,\alpha,\mathbf{t}}|/|S^{(3)}_{B,\alpha,\mathbf{t}}|$. Fix an $\mathbf{s} \in S^{(3)}_{B,\alpha,\mathbf{t}}$. As $\mathbf{t} \in \mathrm{Span}(L_{K,2})$, we have:

$$\sqrt{d}(a_t - a_s) = \sum_j (w_{t,j} - \lfloor w_{t,j} \rceil) \log \mathcal{N}(\mathfrak{p}_j) + \sum_j w'_{t,j} \log \mathcal{N}(\mathfrak{q}_j) - \sum_j w'_{s,j} \log \mathcal{N}(\mathfrak{q}_j),$$

where the $w_{t,j}$ (respectively $w'_{t,j}$ and $w'_{s,j}$) are the coordinates of $\mathbf{w}_t$ (respectively $\mathbf{w}'_t$ and $\mathbf{w}'_s$). We define $b_t = \sum_j (w_{t,j} - \lfloor w_{t,j} \rceil) \log \mathcal{N}(\mathfrak{p}_j) + \sum_j w'_{t,j} \log \mathcal{N}(\mathfrak{q}_j)$, which depends only on $\mathbf{t}$. We would like to have a lower bound on the proportion of vectors $\mathbf{s} \in S^{(3)}_{B,\alpha,\mathbf{t}}$ such that $|\sum_j w'_{s,j} \log \mathcal{N}(\mathfrak{q}_j) - b_t| \leq \sqrt{d}\alpha$. In other words, we would like a lower bound on the probability that $|\sum_j w'_{s,j} \log \mathcal{N}(\mathfrak{q}_j) - b_t| \leq \sqrt{d}\alpha$, when $\mathbf{s}$ is chosen uniformly at random in $S^{(3)}_{B,\alpha,\mathbf{t}}$. In order to simplify the estimation, we will assume in the following that when $\mathbf{s}$ is chosen uniformly at random in $S^{(3)}_{B,\alpha,\mathbf{t}}$, then the vector $\mathbf{w}'_s$ is sampled uniformly in $\{-1, 0, 1\}^r$, with B non-zero coefficients (this is an over-simplification, as we are already restricted to $S^{(3)}_{B,\alpha,\mathbf{t}}$). Let us write Y_j the random variables $w'_{s,j} \log \mathcal{N}(\mathfrak{q}_j)$ for all $1 \leq j \leq r$.

First, let us compute an upper bound on $|b_t|$ (the random variable $|\sum_j Y_j|$ being bounded by δB, if $|b_t| > \delta B + \alpha\sqrt{d}$, then the event would be empty). We have

$$|b_t| \leq \sum_j |w_{t,j} - \lfloor w_{t,j} \rceil| \cdot \delta_0 + \sum_j |w'_{t,j}| \cdot \delta \leq \delta_0 \cdot r_0 + \delta \cdot \|\mathbf{w}'_t\|_1 \leq 0.02 \cdot \delta B,$$

using the assumptions on B and $\|\mathbf{w}'_t\|_1 \leq \sqrt{r} \cdot \|\mathbf{w}'_t\|$.

The element b_t is hence small enough to be reached by the variable $\sum_j Y_j$. However, it will be in the tail of the distribution: the sum of B independent variables bounded by δ has standard deviation at most $\sqrt{B}\cdot\delta$, which is asymptotically smaller than $|b_t|$. This makes the probability very small and difficult to estimate. In order to circumvent this difficulty, we "recenter" our target b_t. A way to do so is to condition the probability at stake on the event that the first few Y_j's have their sum very close to b_t.

More formally, recall that the ideals $\mathfrak{q}_j$'s are chosen so that at least $B/2$ of them have norms at least $2^{\delta/2}$, and sort them by decreasing algebraic norm. Hence, we have $\delta/2 \leq \log \mathcal{N}(\mathfrak{q}_j) \leq \delta$ for $j \leq B/2$. Now, because $|b_t| \leq 0.02 \cdot \delta B$, we know that there exists a choice of $w'_{s,1}, \ldots, w'_{s,\lceil 0.04 \cdot B \rceil} \in \{-1, 1\}$ such that $|\sum_{j \leq \lceil 0.04 \cdot B \rceil} Y_j - b_t| \leq \delta$. Indeed, we can for instance choose the signs of the first $w'_{s,j}$ to be the same as the sign of b_t until we "reach it", that is, until we are at distance at most δ. Then we choose the next sign to be negative, resp. positive if the current sum is larger, resp. smaller than b_t. This ensures that we will never be at distance more than δ from b_t when all the $\lceil 0.04 \cdot B \rceil$ signs have been chosen. We are left with $\lfloor 0.96 \cdot B \rfloor$ non-zero values $w'_{s,j}$ to choose among the $r - \lceil 0.04 \cdot B \rceil$ remaining ideals, but our target vector is now $b'_t := b_t - \sum_{j \leq \lceil 0.04 \cdot B \rceil} Y_j$, which is much smaller than b_t. In particular, $|b'_t|$ is asymptotically smaller than the standard deviation of $\sum_{j > \lceil 0.04 \cdot B \rceil} Y_j$.

Overall, we obtain by conditional probabilities that

$$\begin{aligned}
&\Pr(|\sum_j Y_j - b_t| \leq \sqrt{d}\alpha) \\
&\geq \Pr\big(|\sum_j Y_j - b_t| \leq \sqrt{d}\alpha \text{ and } |b_t - \sum_{j \leq \lceil 0.04 \cdot B \rceil} Y_j| \leq \delta \text{ and } w'_{s,j} \neq 0 \ \forall j \leq \lceil 0.04 \cdot B \rceil\big) \\
&= \Pr\big(|\sum_j Y_j - b_t| \leq \sqrt{d}\alpha \ \big| \ |b_t - \sum_{j \leq \lceil 0.04 \cdot B \rceil} Y_j| \leq \delta \text{ and } w'_{s,j} \neq 0 \ \forall j \leq \lceil 0.04 \cdot B \rceil\big) \\
&\quad \cdot \Pr\big(|b_t - \sum_{j \leq \lceil 0.04 \cdot B \rceil} Y_j| \leq \delta \text{ and } w'_{s,j} \neq 0 \ \forall j \leq \lceil 0.04 \cdot B \rceil\big), \qquad (4.1)
\end{aligned}$$

where the probabilities are taken over the uniform choice of the $w'_{s,j}$'s in $\{-1, 0, 1\}^r$ with B non-zero coefficients. We first deal with the second probability in Equation (4.1). We have seen that there exists at least one choice of the first $\lceil 0.04 \cdot B \rceil$ elements $w'_{s,j}$ such that $|b_t - \sum_{j \leq \lceil 0.04 \cdot B \rceil} Y_j| \leq \delta$. Hence, by counting the number of possible choices for the remaining $\lfloor 0.96 \cdot B \rfloor$ non-zero elements $w'_{s,j}$, we obtain that

$$\begin{aligned}
&\Pr\big(|b_t - \sum_{j \leq \lceil 0.04 \cdot B \rceil} Y_j| \leq \delta \text{ and } w'_{s,j} \neq 0 \ \forall j \leq \lceil 0.04 \cdot B \rceil\big) \\
&\geq \frac{2^{\lfloor 0.96 \cdot B \rfloor} \cdot \binom{r - \lceil 0.04 \cdot B \rceil}{\lfloor 0.96 \cdot B \rfloor}}{2^B \cdot \binom{r}{B}} \\
&\geq 2^{-\lceil 0.04 \cdot B \rceil} \cdot \left(\frac{r - \lceil 0.04 \cdot B \rceil}{\lfloor 0.96 \cdot B \rfloor}\right)^{\lfloor 0.96 \cdot B \rfloor} \cdot \left(\frac{B}{\mathrm{e} \cdot r}\right)^B \\
&\geq \frac{(r - \lceil 0.04 \cdot B \rceil)^{\lfloor 0.96 \cdot B \rfloor}}{\mathrm{e}^B \cdot 2^{\lceil 0.04 \cdot B \rceil} \cdot r^B} \\
&\geq \frac{(0.96 \cdot r)^{\lfloor 0.96 \cdot B \rfloor}}{\mathrm{e}^B \cdot 2^{\lceil 0.04 \cdot B \rceil} \cdot r^B} \\
&\geq \frac{0.344^B}{r^{\lceil 0.04 \cdot B \rceil}},
\end{aligned}$$

where we used the fact that for any $B \leq r$, it holds that $(r/B)^B \leq \binom{r}{B} \leq (\mathrm{e} \cdot r/B)^B$.

It remains to deal with the first probability involved in Equation (4.1). We showed above that this probability is equal to $\Pr(|\sum_{j > 0.04 \cdot B} Y_j - b'_t| \leq \alpha\sqrt{d})$, where $|b'_t| \leq \delta$ and the $w'_{s,j}$ are chosen uniformly at random in $\{-1, 0, 1\}^{r - \lceil 0.04 \cdot B \rceil}$, with $\lfloor 0.96 \cdot B \rfloor$ non-zero coefficients. Because our target is now closer to the center of the distribution of $\sum_j Y_j$, we will assume that

$$\Pr(|\sum_{j>\lceil 0.04\cdot B\rceil} Y_j - b'_t| \le \alpha\sqrt{d}) \ge \frac{\alpha\sqrt{d}}{B\delta},$$

where the lower bound is the probability we would have obtained if the random variable $\sum_{j>\lceil 0.04\cdot B\rceil} Y_j$ was uniformly distributed in $[-B\delta, B\delta]$. This assumption is justified by the fact that the random variable $\sum_{j>\lceil 0.04\cdot B\rceil} Y_j$ has a bell shape and that b'_t is close to the center of the bell (the standard deviation of $\sum_j Y_j$ is roughly $\sqrt{0.96\cdot B}\cdot\delta \gg |b'_t|$). Hence, the probability to be close to b'_t should be larger in the bell shape case than in the uniform case (over $[-0.96\cdot B\delta, 0.96\cdot B\delta] \subset [-B\delta, B\delta]$). This lower bound is backed with numerical experiments described in Section 6.1. This leads us to the following lower bound.

$$\Pr(|\sum_j Y_j - b_t| \le \sqrt{d}\alpha) \ge \frac{0.344^B}{r^{0.04\cdot B}}\cdot\frac{\alpha\sqrt{d}}{B\delta}.$$

We finally obtain that

$$\begin{aligned}|S^{(4)}_{B,\alpha,\mathbf{t}}| &\ge \frac{0.344^B\cdot\alpha\sqrt{d}}{\delta B\cdot r^{0.04\cdot B}}\cdot\frac{\alpha^{d-1}}{(2\pi)^{d/2}\cdot\det(\Lambda)}\cdot\frac{1}{h_K}\cdot\left(\frac{2r}{B}\right)^B\\ &\ge \left(\frac{\alpha}{\sqrt{2\pi}}\right)^d\frac{1}{\delta B\cdot\det(\Lambda)\cdot h_K}\left(\frac{0.344\cdot 2r}{B\cdot r^{0.04}}\right)^B\\ &\ge \left(\frac{\alpha}{\sqrt{2\pi}}\right)^d\frac{1}{\delta B\cdot\det(\Lambda)\cdot h_K}\left(\frac{r^{0.96}}{2B}\right)^B.\end{aligned}$$

When the above is ≥ 1, we expect that there exists $\mathbf{s}\in S^{(4)}_{B,\alpha,\mathbf{t}}$. If that is the case, then we have

$$\|\mathbf{s}-\mathbf{t}\|^2 \le (\beta\cdot\sqrt{2d}\cdot\alpha)^2 + r_0 + \|\mathbf{w}'_t - \mathbf{w}'_s\|^2.$$

By condition on B, we know that $r_0 \le 0.01\cdot B$. Also, by choice of $\mathbf{w}'_t$ (and using the fact that $r\ge B$), we have that $\|\mathbf{w}'_t - \mathbf{w}'_s\|^2 \le (\sqrt{B}+0.01\cdot\sqrt{B})^2 \le 1.03\cdot B$. Finally, choosing α minimal provides the result.

4.4 A "Euclidean division" over R

We will need the following technical observation that, given $a, b\in K_\mathbb{R}$, it is possible to add a small multiple ka of a to b to ensure that $\mathcal{N}(b+ka)\ge\mathcal{N}(a)$.

Lemma 4.5. *For any $a\in K_\mathbb{R}^\times$ and $b\in K_\mathbb{R}$, there exists $k\in[-d,d]\cap\mathbb{Z}$ such that $|\mathcal{N}(b+ka)|\ge|\mathcal{N}(a)|$.*

Note that an integer k such as in Lemma 4.5 can be found efficiently by exhaustive search.

Algorithm 4.1 A Euclidean division over R

Input: A fractional ideal $\mathfrak{a}$, and two elements $a \in K_{\mathbb{R}}^{\times}$ and $b \in K_{\mathbb{R}}$.
Output: A pair $(u, v) \in R \times \mathfrak{a}$.

```
    Computing a better pair $(a_1, b_1)$
 1: Find $s \in \mathfrak{a}^{-1} \setminus \{0\}$ such that $\|s\|_\infty \leq \mathrm{c} \cdot \mathcal{N}(\mathfrak{a}^{-1})^{1/d}$ as in Lemma 2.5.
 2: Find $y \in R \setminus \{0\}$ such that $\|ya\|_\infty \leq \mathrm{c} \cdot |\mathcal{N}(a)|^{1/d}$ as in Lemma 2.5 (with ideal $\langle a \rangle$). Define $a_1 = ya$.
 3: Solve CVP in $R$ to find $x \in R$ such that $\|b/(s \cdot a_1) - x\| \leq \rho(R)$.
 4: Find $k \in \mathbb{Z} \cap [-d, d]$ such that $|\mathcal{N}(b - xsa_1 + ksa_1)| \geq |\mathcal{N}(sa_1)|$ (see Lemma 4.5).
 5: Define $b_1 = b + (k - x)s \cdot a_1$.
    Defining the target vector and solving CVP
 6: Compute $(w_{t,j})_{j \leq r_0}$ and $g_t$ such that $\mathfrak{a}^{-1} = \prod_j \mathfrak{p}_j^{w_{t,j}} \langle g_t \rangle$. Let $\mathbf{w}_t = (w_{t,j})_{j \leq r_0}$.
 7: Let $a_t = (\log \mathcal{N}|b_1/(a_1 g_t)|)/\sqrt{d}$, $\theta_t$ be the first $r_1 + r_2$ coordinates of $\overline{\mathrm{Log}}(b_1/(a_1 g_t))$ and $\mathbf{h}_t = i_{E \cap H}(\mathit{\Pi}_H(\mathrm{Log}(b_1/(a_1 g_t))))$.
 8: Define $\mathbf{t} = (\beta a_t \| \beta \theta_t \| \beta \mathbf{h}_t \| \mathbf{w}_t \| \mathbf{0})$.
 9: Solve CVP in $L_{K,2}$ with target vector $\mathbf{t}$, to obtain a vector $\mathbf{s}$.
    Using $\mathbf{s}$ to create a good ring element
10: Write $\mathbf{s} = (\beta a_s \| \beta \theta_s \| \beta \mathbf{h}_s \| \mathbf{w}_s \| \mathbf{w}'_s)$ and let $g_s \in K^*$ be the associated element as in Lemma 4.4.
11: Define the ideal $I = \mathfrak{a} \prod_{j : w_{s,j} - w_{t,j} < 0} \mathfrak{p}_j^{w_{t,j} - w_{s,j}} \prod_{j : w'_{s,j} < 0} \mathfrak{q}_j^{-w'_{s,j}}$.
12: Find $v \in I \setminus 0$ such that $\|v\|_\infty \leq c \cdot \mathcal{N}(I)^{1/d}$ as in Lemma 2.5.
13: Define $u' = g_s \cdot g_t \cdot v$.
14: return $(u'y + (k - x)sy \cdot v, v)$.
```

Proof. We know that $|\mathcal{N}(a)| = \prod_i |\sigma_i(a)|$ so it suffices to find k such that $|\sigma_i(b + ak)| \geq |\sigma_i(a)|$ holds for all $i \leq d$. Now, the condition $|\sigma_i(b+ak)| < |\sigma(a)|$ is equivalent to $|\sigma_i(b/a) + k| < 1$. As a complex plane open circle of radius 1 contains at most two integers, we deduce that for each $i \leq d$, there are at most two integers k such that $|\sigma_i(b+ak)| < |\sigma(a)|$. Because the set $[-d, d] \cap \mathbb{Z}$ contains $2d+1$ different values of k, then at least one of these should satisfy $|\sigma_i(b + ak)| \geq |\sigma_i(a)|$ for all i. $\square$

We can now describe our "Euclidean division" algorithm over R. Our algorithm takes as input a fractional ideal $\mathfrak{a}$ and two elements $a, b \in K_{\mathbb{R}}$, and outputs a pair $(u, v) \in R \times \mathfrak{a}$. The first five steps of this algorithm aim at obtaining, for any input (a, b), a replacement (a_1, b_1) that satisfies some conditions. Namely, we would like a_1 to be balanced, i.e., $\|a_1\|$ should not be significantly more than $\mathcal{N}(a_1)^{1/d}$. We also would like b_1 to be not much larger that a_1 and $\mathcal{N}(a_1/b_1)$ to be close to 1. These conditions are obtained by multiplying the element a by an appropriate element of R, and removing a multiple of a from b. Note that we require that the output element v should not be too large. As b is not multiplied by anything, these normalization steps will not impact this output property. After these first five steps, the core of the algorithm begins. It mainly consists in the creation of a good target vector $\mathbf{t}$ in $\mathbb{R}^{\nu+1}$, followed by a CVP computation in the lattice $L_{K,2}$. In addition to this CVP instance in $L_{K,2}$, the algorithm also requires solving some CVP instances in the lattice R and in the lattice $L_{K,1}$ of Lemma 2.5.

Theorem 4.6 (Heuristic). *Assume that $\mathfrak{a}$ satisfies $\mathrm{c}^{-d} \leq \mathcal{N}(\mathfrak{a}) \leq \mathrm{c}^d$, with c as in Lemma 2.5. Assume also that B and r are chosen so that*

$$B \geq \max\left(100 \cdot d \cdot \log[(\rho(R)+d)c^4], 2\log h_K \cdot (103 \cdot \frac{\delta_0}{\delta})^2\right),$$

$$\alpha_0 := \sqrt{2\pi}\ \left((\frac{2B}{r^{0.96}})^B \cdot \delta B(\det \Lambda) h_K\right)^{1/d} \leq \frac{\varepsilon}{43 \cdot \sqrt{d} \cdot (\rho(R)+d)\mathrm{c}^4 \cdot \mathrm{e}^{0.55 \cdot \delta \cdot B/d}},$$

for some $\varepsilon > 0$*, and that* $\alpha_0 \leq (\log d)/(12d^{2.5})$*. Set the scaling parameter* β *of* $\mathbf{B}_{L_{K,2}}$ *as in Heuristic 1. Then, under Heuristic 1 and the heuristics of Lemma 2.5, Algorithm 4.1 outputs a pair* $(u,v) \in R \times \mathfrak{a}$ *with*

$$\|ua + vb\|_\infty \leq \varepsilon \cdot \|a\|_\infty,$$
$$\|v\|_\infty \leq \mathrm{c} \cdot 2^{0.55 \cdot \delta \cdot B/d}.$$

If $a, b \in K$*, then Algorithm 4.1 performs a number of quantum operations that is polynomial in* $\log \Delta_K$ *and the input bit-length, and makes five CVP calls in fixed lattices (three in* $L_{K,1}$*, one in* $L_{K,2}$ *and one in* R*).*

Proof. Throughout the proof, we keep the notations of Algorithm 4.1.

We first prove that $(u,v) \in R \times \mathfrak{a}$. As $s \in \mathfrak{a}^{-1}$ and $x, k, y \in R$, it suffices to prove that $(u', v) \in R \times \mathfrak{a}$. By definition of g_t and g_s, we have

$$\langle g_s g_t \rangle = \mathfrak{a}^{-1} \prod_j \mathfrak{p}_j^{w_{s,j} - w_{t,j}} \prod_j \mathfrak{q}_j^{w'_{s,j}} = J \cdot I^{-1},$$

with $J = \prod_{j: w_{s,j} - w_{t,j} > 0} \mathfrak{p}_j^{w_{s,j}-w_{t,j}} \prod_{j: w'_{s,j} > 0} \mathfrak{q}_j^{w'_{s,j}}$. As the $\mathfrak{p}_j$'s and $\mathfrak{q}_j$'s are integral ideals, we see that $J \subseteq R$ and $I \subseteq \mathfrak{a}$. As $v \in I$, we obtain that $v \in \mathfrak{a}$. Since $g_s \cdot g_t \in JI^{-1}$ and $v \in I$, we also have $u' = g_s g_t v \in JI^{-1}I = J \subseteq R$. This gives our first claim.

As a preliminary step towards bounding $\|ua + bv\|_\infty = \|u'a_1 + vb_1\|_\infty$, we study the sizes of a_1 and b_1. Using the equality $b_1 = b - xsa_1 + ksa_1$, we have

$$\|b_1\|_\infty \leq (\|b/(sa_1) - x\|_\infty + |k|) \cdot \|sa_1\|_\infty \leq (\rho(R)+d) \cdot \|s\|_\infty \cdot \|a_1\|_\infty.$$

By definition of a_1, we have $\|a_1\|_\infty \leq \mathrm{c}\|a\|_\infty$. By assumption on $\mathfrak{a}$, we also have $\|s\|_\infty \leq \mathrm{c} \cdot \mathcal{N}(\mathfrak{a}^{-1})^{1/d} \leq \mathrm{c}^2$. Hence, we obtain

$$\|b_1\|_\infty \leq (\rho(R)+d)\mathrm{c}^3 \|a\|_\infty.$$

Now, by definition of a_1, we know that $\|a_1\|_\infty \leq \mathrm{c} \cdot |\mathcal{N}(a_1)|^{1/d}$. Hence, we obtain

$$\mathrm{c}^{-1} \leq |\mathcal{N}(b_1/a_1)|^{1/d} \leq \mathrm{c} \cdot \frac{\|b_1\|_\infty}{\|a_1\|_\infty} \leq (\rho(R)+d) \cdot \mathrm{c}^3.$$

The left inequality is provided by the choice of k at Step 4 (and the fact that $\mathcal{N}(s) \geq \mathcal{N}(\mathfrak{a}^{-1})$).

To bound $\|u'a_1 + vb_1\|_\infty$, we estimate the closeness of $\mathbf{t}$ and $\mathbf{s}$. If $\mathbf{t}$ was in $\mathrm{Span}(L_{K,2})$, then we could apply Heuristic 1. As this is not necessarily the case, we first need to compute the distance between $\mathbf{t}$ and $\mathrm{Span}(L_{K,2})$. This is done in the proof of the following lemma, which is provided after the current proof.

Lemma 4.7 (Heuristic). *Under the assumptions of Theorem 4.6, we have* $\|\mathbf{s}-\mathbf{t}\|_2 \leq \sqrt{1.06 \cdot B}$.

This lemma implies that

$$\|(a_s \| \theta_s \| \mathbf{h}_s) - (a_t \| \theta_t \| \mathbf{h}_t)\|_2 \leq \sqrt{1.06 \cdot B}/\beta \leq 15 \cdot \sqrt{d} \cdot \alpha_0.$$

By definition of $\mathbf{t}$ and construction of $L_{K,2}$, this means that

$$\| \overline{\operatorname{Log}}(g_t g_s \cdot a_1/b_1)\|_2 = \|(a_s \| \theta_s \| \mathbf{h}_s) - (a_t \| \theta_t \| \mathbf{h}_t)\|_2 \leq 15 \cdot \sqrt{d} \cdot \alpha_0.$$

Recall that $u'/v = g_t g_s$. Hence we have $\| \overline{\operatorname{Log}}(u'a_1) - \overline{\operatorname{Log}}(vb_1)\|_\infty \leq 15 \cdot \sqrt{d} \cdot \alpha_0$. Using Lemma 4.3, we deduce that

$$\begin{aligned}\|u'a_1 - vb_1\|_\infty &\leq (\mathrm{e}^{15\cdot\sqrt{2d}\cdot\alpha_0} - 1) \cdot \|b_1\|_\infty \cdot \|v\|_\infty \\ &\leq 43 \cdot \sqrt{d} \cdot \alpha_0 \cdot \|b_1\|_\infty \cdot \|v\|_\infty,\end{aligned}$$

where we used the fact that $\alpha_0 \leq (\log d)/(12d^{2.5})$ and so the exponent should be smaller than $(\log 2)/\sqrt{2}$ for d large enough. We have already bounded $\|b_1\|_\infty$. We now bound $\|v\|_\infty$. By definition of v, we have $\|v\|_\infty \leq \mathrm{c} \cdot \mathcal{N}(I)^{1/d}$. The task is then to provide an upper bound on $\mathcal{N}(I)$. As $IJ = \mathfrak{a} \cdot \prod_j \mathfrak{p}_j^{|w_{s,j} - w_{t,j}|} \cdot \prod_j \mathfrak{q}_j^{|w'_{s,j}|}$, we have:

$$\begin{aligned}\log \mathcal{N}(IJ) &= \log \mathcal{N}(\mathfrak{a}) + \sum_j |w_{s,j} - w_{t,j}| \log \mathcal{N}(\mathfrak{p}_j) + \sum_j |w'_{s,j}| \cdot \log \mathcal{N}(\mathfrak{q}_j) \\ &\leq \log \mathcal{N}(\mathfrak{a}) + \|\mathbf{w}_s - \mathbf{w}_t\|_1 \cdot \delta_0 + \|\mathbf{w}'_s\|_1 \cdot \delta\end{aligned}$$

Recall from Lemma 4.7 that we have $\|\mathbf{s}-\mathbf{t}\|_2 \leq \sqrt{1.06 \cdot B}$. This implies that $\|\mathbf{w}_s - \mathbf{w}_t\|_2, \|\mathbf{w}'_s\|_2 \leq \sqrt{1.06 \cdot B}$. Note that

$$\|\mathbf{w}_s - \mathbf{w}_t\|_1 \leq \sqrt{r_0} \cdot \|\mathbf{w}_s - \mathbf{w}_t\|_2 \leq 1.03 \cdot \sqrt{B \cdot r_0} \leq 0.01 \cdot \frac{\delta}{\delta_0} \cdot B,$$

by assumption on B and the fact that $r_0 \leq 2 \log h_K$. For $\mathbf{w}'_s$, we use the fact that it has integer coordinates, to obtain $\|\mathbf{w}'_s\|_1 \leq \|\mathbf{w}'_s\|_2^2 \leq 1.06 \cdot B$. We thus obtain

$$\log \mathcal{N}(IJ) \leq \log \mathcal{N}(\mathfrak{a}) + 1.07 \cdot \delta \cdot B.$$

As J is integral, this gives an upper bound on $\mathcal{N}(I)$. However this upper bound is not sufficient for our purposes. We improve it by giving an upper bound on $\log \mathcal{N}(IJ^{-1})$, using the fact that the ideal IJ^{-1} is designed to have an algebraic norm close to the one of a_1/b_1. Recall that a_1 and b_1 are constructed so that $\mathcal{N}(a_1/b_1)$ is close to 1, which means that I and J should have roughly the same norm. More precisely, it is worth recalling that $I^{-1}J = \langle g_s g_t \rangle$, and that $\| \overline{\operatorname{Log}}(g_t g_s \cdot a_1/b_1)\|_2 \leq 15 \cdot \sqrt{d} \cdot \alpha_0$. Looking at the first coordinate of the $\overline{\operatorname{Log}}$ vector and multiplying it by $\sqrt{d}$ shows that $|\log|\mathcal{N}(g_s g_t)| + \log|\mathcal{N}(a_1/b_1)||\leq 15 \cdot d \cdot \alpha_0$. This gives us

$$\log \mathcal{N}(IJ^{-1}) \leq |\log|\mathcal{N}(a_1/b_1)|| + 15 \cdot d \cdot \alpha_0$$

Combining the bounds on $\log \mathcal{N}(IJ)$ and $\log \mathcal{N}(IJ^{-1})$, we finally obtain that

$$\log \mathcal{N}(I) \leq \frac{1}{2} \cdot |\log \mathcal{N}(\mathfrak{a})| + 0.535 \cdot \delta \cdot B + \frac{1}{2} \cdot |\log |\mathcal{N}(a_1/b_1)|| + 7.5 \cdot d \cdot \alpha_0.$$

We have seen that $\mathrm{c}^{-1} \leq |\mathcal{N}(b_1/a_1)|^{1/d} \leq (\rho(R) + d) \cdot \mathrm{c}^3$. Finally, recall that $\mathrm{c}^{-d} \leq \mathcal{N}(\mathfrak{a}) \leq \mathrm{c}^d$. Hence, we conclude that $|\log |\mathcal{N}(a_1/b_1)|| + |\log \mathcal{N}(\mathfrak{a})| \leq d \cdot \log((\rho(R) + d) \cdot \mathrm{c}^4) \leq 0.01 \cdot B$ by assumption on B. Recall that we assumed that $\alpha_0 \leq (\log d)/(12d^{2.5}) \leq 1/d$. Hence, we have $d \cdot \alpha_0 \leq 1$. Using the fact that $B \geq 750$ (which is implied by the second term in the max), we obtain $7.5 \cdot d \cdot \alpha_0 \leq 0.01 \cdot B$. We conclude that

$$\log \mathcal{N}(I) \leq 0.55 \cdot \delta \cdot B.$$

Collecting terms and using the assumptions, this allows us to write

$$\begin{aligned} \|u'a_1 - vb_1\|_\infty &\leq 43 \cdot \sqrt{d} \cdot \alpha_0 \cdot \|b_1\|_\infty \cdot \|v\|_\infty \\ &\leq \alpha_0 \cdot 43 \cdot \sqrt{d} \cdot \mathrm{e}^{0.55 \cdot \delta \cdot B/d} \cdot (\rho(R) + d)\mathrm{c}^4 \|a\|_\infty \\ &\leq \varepsilon \cdot \|a\|_\infty. \end{aligned}$$

Finally, the run-time bound follows by inspection. □

Proof (Proof of Lemma 4.7).

By the Pythagorean theorem, if $\mathbf{t}_{L_{K,2}}$ is the orthogonal projection of $\mathbf{t}$ onto $\mathrm{Span}(L_{K,2})$, then we have

$$\|\mathbf{t} - \mathbf{s}\|_2^2 = \|\mathbf{t} - \mathbf{t}_{L_{K,2}}\|^2 + \|\mathbf{t}_{L_{K,2}} - \mathbf{s}\|^2 = \mathrm{dist}(\mathbf{t}, \mathrm{Span}(L_{K,2}))^2 + \|\mathbf{t}_{L_{K,2}} - \mathbf{s}\|^2.$$

This quantity is minimal when $\|\mathbf{t}_{L_{K,2}} - \mathbf{s}\|$ is minimal, and so the closest point to $\mathbf{t}$ in $L_{K,2}$ is also the closest point to $\mathbf{t}_{L_{K,2}}$.

First, observe that $\mathrm{Span}(L_{K,2})$ (of dimension ν) is exactly

$$\left\{(\beta a \| \beta\theta \| \beta\mathbf{h} \| \mathbf{w} \| \mathbf{w}') \in \mathbb{R}^{\nu+1} : \ a = \sum_j w_j \frac{\log \mathcal{N}(\mathfrak{p}_j)}{\sqrt{d}} + \sum_j w'_j \frac{\log \mathcal{N}(\mathfrak{q}_j)}{\sqrt{d}} \right\}.$$

Define $\mathbf{v} = (-\sqrt{d}/\beta, 0, \ldots, 0, \log \mathcal{N}(\mathfrak{p}_1), \ldots, \log \mathcal{N}(\mathfrak{q}_r))^T$, where the block of zeros has dimension $2(r_1 + r_2) - 1$. It is orthogonal to $\mathrm{Span}(L_{K,2})$ and satisfies $\|\mathbf{v}\| > \sqrt{r}$, as each one of the last r coefficients is > 1. Hence

$$\begin{aligned} \mathrm{dist}(\mathbf{t}, \mathrm{Span}(L_{K,2})) = \frac{|\langle \mathbf{v}, \mathbf{t} \rangle|}{\|\mathbf{v}\|} &< \frac{|-\sqrt{d}a_t + \sum_j w_{t,j} \log \mathcal{N}(\mathfrak{p}_j)|}{\sqrt{r}} \\ &= \frac{|-\log |\mathcal{N}(b_1/(a_1 g_t))| - \log \mathcal{N}(\mathfrak{a}) - \log |\mathcal{N}(g_t)||}{\sqrt{r}} \\ &= \frac{|\log |\mathcal{N}(a_1/b_1)| - \log \mathcal{N}(\mathfrak{a})|}{\sqrt{r}}. \end{aligned}$$

We have seen in the proof of Theorem 4.6 that $|\log |\mathcal{N}(a_1/b_1)|| \leq d \cdot \log((\rho(R)+d) \cdot \mathrm{c}^3)$. By assumption on $\mathfrak{a}$, we have that $|\log |\mathcal{N}(a_1/b_1)|| + |\log \mathcal{N}(\mathfrak{a})| \leq d \cdot \log((\rho(R)+d) \cdot \mathrm{c}^4)$. The latter bound is $\leq 0.01 \cdot B$. So we obtain that $\mathrm{dist}(\mathbf{t}, \mathrm{Span}(L_{K,2})) \leq 0.01 \cdot B/\sqrt{r} \leq \sqrt{0.01 \cdot B}$.

As $\|\mathbf{t} - \mathbf{t}_{L_{K,2}}\|_2 \leq 0.01 \cdot B/\sqrt{r}$ and the last r entries of $\mathbf{t}$ are zero, we have that the last r entries of $\mathbf{t}_{L_{K,2}}$ have euclidean norm $\leq 0.01 \cdot B/\sqrt{r}$. We can hence apply Heuristic 1 to $\mathbf{t}_{L_{K,2}}$, which gives us $\|\mathbf{t} - \mathbf{s}\|_2^2 = \|\mathbf{t} - \mathbf{t}_{L_{K,2}}\|^2 + \|\mathbf{t}_{L_{K,2}} - \mathbf{s}\|^2 \leq 0.01 \cdot B + 1.05 \cdot B$. □

We observe that the parameters r and B of Theorem 4.6 can be instantiated as $B = \widetilde{O}(d \log \rho(R))$ (recall from preliminaries that $d \log \rho(R) = \Omega(\log \Delta_K)$), and $r^{0.96} = \Theta((1/\varepsilon)^{d/B} \cdot B \cdot \mathrm{e}^{0.55\delta})$. Thanks to the 0.55 in the exponent, this choice of r is compatible with the condition $r \leq O(\mathrm{e}^{\delta}/\delta)$ which was required for the construction of the lattice $L_{K,2}$ (recall that we want r prime ideals of norm smaller than e^{δ}). We note also that the constants 0.96 and 0.55 appearing in the exponent can be chosen as close as we want to 1 and 0.5 respectively, by adapting the argument above. Hence, assuming $(1/\varepsilon)^{d/B} = O(1)$, we expect to be able to choose e^{δ} as small as $B^{2+\eta}$ for any $\eta > 0$. Overall, the following corollary gives an instantiation of Theorem 4.6 with parameters that are relevant to our upcoming divide-and-swap algorithm.

Corollary 4.8 (Heuristic). *Let $\varepsilon = 1/2^{\widetilde{O}(\log \Delta_K)/d}$. For any $\eta > 0$, there exists a lattice L_K of dimension $O((d \log \rho(R))^{2+\eta})$, an upper bound $C = 2^{\widetilde{O}(d \log \rho(R))/d}$ and an algorithm $\mathcal{A}$ that achieve the following. Under Heuristic 1 and the heuristics of Lemma 2.5, algorithm $\mathcal{A}$ takes as inputs $a \in K_{\mathbb{R}}^{\times}$, $b \in K_{\mathbb{R}}$ and an ideal $\mathfrak{a}$ satisfying $\mathrm{c}^{-d} \leq \mathcal{N}(\mathfrak{a}) \leq \mathrm{c}^{d}$, and outputs $u, v \in R \times \mathfrak{a}$ such that*

$$\|ua + bv\|_\infty \leq \varepsilon \cdot \|a\|_\infty$$
$$\|v\|_\infty \leq C.$$

When restricted to inputs a, b belonging to K, Algorithm $\mathcal{A}$ performs a number of quantum operations that is polynomial in $\log \Delta_K$ and the input bit-length, and makes five calls to an oracle solving CVP in L_K.

Proof. Consider an instantiation of Theorem 4.6 with $B = \widetilde{\Theta}(d \log \rho(R))$, $\delta = 2.451 \cdot \log B$ and $r = B^{2.45}$. This choice of parameters asymptotically satisfies $r \leq O(\mathrm{e}^{\delta}/\delta)$, which was required for the generation of the lattice $L_{K,2}$. It also satisfies the constraints of Theorem 4.6 (using the asymptotic inequality $d \log \rho(R) = \Omega(\log \Delta_K)$). It can be checked by proof inspection that the constant 2.45 can be adapted to $2+\eta$ for an arbitrary $\eta > 0$, which allows to obtain the asymptotic parametrization of the statement. Finally, observe that the algorithm relies on oracles that solve CVP in R (one call), $L_{K,1}$ (three calls) and $L_{K,2}$ (one call). We can consider the direct sum of these lattices, to obtain a single lattice L_K. □

4.5 The divide-and-swap algorithm

In this subsection, we describe a divide-and-swap algorithm, which takes as input a pseudo-basis of a rank-2 module and outputs a short non-zero vector of this module (for the algebraic norm). In order to do so, we will need to link the Euclidean and algebraic norms of vectors appearing during the execution, and limit the degree of freedom of the ideal coefficients. For this purpose we use a strengthening of the notion of scaled pseudo-bases from Section 3.2.

Definition 4.9. *A pseudo-basis* $((I_i, \mathbf{b}_i))_{i \le n}$*, with* $I_i \subset K$ *and* $\mathbf{b}_i \in K_{\mathbb{R}}^m$ *for all* $i \le n$*, is said strongly scaled if, for all* $i \le n$*,*

$$R \subseteq I_i, \quad \mathcal{N}(I_i) \ge \mathrm{c}^{-d} \text{ and } \|r_{ii}\|_\infty \le \mathrm{c} \cdot \mathcal{N}(r_{ii} I_i)^{1/d},$$

where c *is as in Lemma 2.5.*

Algorithm 4.2 below strongly scales a given module pseudo-basis. It is a direct adaptation of Algorithm 3.2 in which the LLL algorithm is replaced by the algorithm from Lemma 2.5 (relying on a CVP oracle for $L_{K,1}$).

Algorithm 4.2 Strongly scaling the ideals.

```
Input: A pseudo-basis $((I_i, \mathbf{b}_i))_{i \le n}$ of a module $M$.
Output: A strongly scaled pseudo-basis $((I'_i, \mathbf{b}'_i))_{i \le n}$ of $M$.
1: for $i = 1$ to $n$ do
2:     Use Lemma 2.5 to find $s_i \in r_{ii} \cdot I_i \setminus \{0\}$ such that $\|s_i\|_\infty \le \mathrm{c} \cdot \mathcal{N}(r_{ii} I_i)^{1/d}$;
3:     Write $s_i = r_{ii} \cdot x_i$, with $x_i \in I_i$;
4:     Define $I'_i = I_i \cdot \langle x_i \rangle^{-1}$ and $\mathbf{b}'_i = x_i \mathbf{b}_i$.
5: end for
6: return $((I'_i, \mathbf{b}'_i))_{i \le n}$.
```

Lemma 4.10. *Algorithm 4.2 outputs a strongly scaled pseudo-basis of the module* M *generated by the input pseudo-basis and preserves the* $\mathcal{N}(r_{ii} I_i)$*'s. If* $M \subseteq K^m$*, then it performs a number of quantum operations that is polynomial in* $\log \Delta_K$ *and the input bit-length, and makes* n *calls to an oracle solving CVP in the lattice* $L_{K,1}$ *of Lemma 2.5.*

The proof is a direct adaptation of the proof of Lemma 3.6. We will use it in dimension 2, but stated it in dimension n (for the sake of consistency).

We can now describe Algorithm 4.3, our divide-and-swap algorithm. During the execution of the algorithm, the R-factor of the current matrix $(\mathbf{b}_1 | \mathbf{b}_2)$ is always computed. The algorithm is very similar to the LLL algorithm in dimension 2, except for Step 4, which is specific to this algorithm. This step ensures that when we swap the vectors, we still obtain a pseudo-basis of the input module. This seems necessary, as our Euclidean division over R involves a multiplication of the second vector by a ring element, and hence the new vector and the second pseudo-basis vector may not span the whole module anymore. At Step 4, note that the gcd is well-defined, as $\langle u \rangle$ and $\langle v \rangle \mathfrak{a}^{-1}$ are integral ideals. As an alternative to Step 4, we could use Lemma 2.8 as in Algorithm 3.1.

Lemma 4.11. *Let* $\gamma_{\mathcal{N}} \ge 4 \cdot C \cdot \mathrm{c}^2$*, where* C *is as in Corollary 4.8. Then, given as input a pseudo-basis of a rank-2 module* $M \subset K_{\mathbb{R}}^2$*, Algorithm 4.3 outputs a vector* $\mathbf{v} \in M \setminus \{\mathbf{0}\}$ *such that* $\mathcal{N}(\mathbf{v}) \le {\gamma_{\mathcal{N}}}^d \lambda_1^{\mathcal{N}}(M)$*. Further, if* $M \subseteq K^2$ *and assuming that Algorithms 4.1 and 4.2 are correct and run in time polynomial in* $\log \Delta_K$ *and their input bit-length, then Algorithm 4.3 also runs in time polynomial in* $\log \Delta_K$ *and the input bit-length.*

Algorithm 4.3 Divide-and-swap.

Input: A pseudo-basis $((\mathfrak{a}_1, \mathbf{b}_1), (\mathfrak{a}_2, \mathbf{b}_2))$ of a module $M \subset K_{\mathbb{R}}^2$.
Output: A vector $\mathbf{v} \in M$.

```
1: while $(\gamma_{\mathcal{N}}/\mathrm{c})^d \mathcal{N}(r_{22}\mathfrak{a}_2) < \mathcal{N}(r_{11}\mathfrak{a}_1)$ do
2:     Strongly scale the pseudo-basis $((\mathfrak{a}_1, \mathbf{b}_1), (\mathfrak{a}_2, \mathbf{b}_2))$ using Algorithm 4.2.
3:     Apply Algorithm 4.1 to $(a, b, \mathfrak{a}) = (r_{11}, r_{12}, \mathfrak{a}_2 \cdot \mathfrak{a}_1^{-1})$ and $\varepsilon = \frac{1}{4\mathrm{c}}$. Let $(u, v)$ be the output.
4:     Let $\mathfrak{b} = \gcd(\langle u \rangle, \langle v \rangle \mathfrak{a}^{-1})$, find $x \in \mathfrak{a}^{-1}\mathfrak{b}^{-1}$ and $y \in \mathfrak{b}^{-1}$ such that $uy - vx = 1$.
5:     Update $(\mathbf{b}_1, \mathbf{b}_2) \leftarrow (u\mathbf{b}_1 + v\mathbf{b}_2, x\mathbf{b}_1 + y\mathbf{b}_2)$ and $(\mathfrak{a}_1, \mathfrak{a}_2) \leftarrow (\mathfrak{a}_1\mathfrak{b}^{-1}, \mathfrak{a}_2\mathfrak{b})$.
6: end while
7: Strongly scale the pseudo-basis $((\mathfrak{a}_1, \mathbf{b}_1), (\mathfrak{a}_2, \mathbf{b}_2))$ using Algorithm 4.2.
8: return $\mathbf{b}_1$
```

Proof. Let us first prove that the pseudo-basis $((\mathfrak{a}_1, \mathbf{b}_1), (\mathfrak{a}_2, \mathbf{b}_2))$ we have throughout the execution of the algorithm remains a pseudo-basis of M. By Lemma 4.10, this property is preserved through Steps 2 and 7. It remains to prove it for Step 5. At this step, we multiply $\begin{pmatrix}\mathbf{b}_1 & \mathbf{b}_2\end{pmatrix}$ on the right by $\mathbf{U} := \begin{pmatrix} u & x \\ v & y \end{pmatrix}$. Let $\mathfrak{a}_1, \mathfrak{a}_2$ (resp. $\mathfrak{a}_1' = \mathfrak{a}_1\mathfrak{b}^{-1}, \mathfrak{a}_2' = \mathfrak{a}_2\mathfrak{b}$) denote the coefficient ideals at the start (resp. completion) of Step 5. We know from the preliminaries that this transformation outputs a pseudo-basis of the same module if $\mathbf{U}$ is invertible over K and $u_{ij} \in \mathfrak{a}_i(\mathfrak{a}_j')^{-1}$ and $u_{ij}' \in \mathfrak{a}_i'(\mathfrak{a}_j)^{-1}$ for all $i, j \in \{1, 2\}$, with $\mathbf{U}' = \mathbf{U}^{-1}$. In our case, because we asked that $uy - vx = 1$, then $\mathbf{U}$ is indeed invertible and we have $\mathbf{U}^{-1} = \begin{pmatrix} y & -x \\ -v & u \end{pmatrix}$. Observe that by definition of $\mathfrak{b} = \gcd(\langle u \rangle, \langle v \rangle \mathfrak{a}^{-1})$, we have $u \in \mathfrak{b}$ and $v \in \mathfrak{a} \cdot \mathfrak{b} = \mathfrak{a}_1^{-1}\mathfrak{a}_2\mathfrak{b}$. Using these properties and the fact that $x \in \mathfrak{a}_1\mathfrak{a}_2^{-1}\mathfrak{b}^{-1}$ and $y \in \mathfrak{b}^{-1}$ by definition, one can then check that all the conditions $u_{ij} \in \mathfrak{a}_i(\mathfrak{a}_j')^{-1}$ and $u_{ij}' \in \mathfrak{a}_i'(\mathfrak{a}_j)^{-1}$ are indeed satisfied.

We now prove that the output vector $\mathbf{v}$ belongs to M. As $\mathbf{v} = \mathbf{b}_1$, it suffices that $R \subseteq \mathfrak{a}_1$. This is provided by the application of Algorithm 4.2 at Step 7.

Assume now that the algorithm terminates, and let us show that the output vector $\mathbf{v}$ satisfies $\mathcal{N}(\mathbf{v}) \leq {\gamma_{\mathcal{N}}}^d \lambda_1^{\mathcal{N}}(M)$. Because of the application of Algorithm 4.2 at Step 7, we have $\mathcal{N}(\mathfrak{a}_1) \geq \mathrm{c}^{-d}$. This, and the inequality $\gamma_{\mathcal{N}} \geq \mathrm{c}$, imply that $\mathcal{N}(\mathbf{b}_1) = \mathcal{N}(r_{11}) \leq \mathrm{c}^d \cdot \mathcal{N}(r_{11}\mathfrak{a}_1) \leq {\gamma_{\mathcal{N}}}^d \cdot \mathcal{N}(r_{11}\mathfrak{a}_1)$. On the other hand, because we exited the while loop, we have $(\gamma_{\mathcal{N}}/\mathrm{c})^d \mathcal{N}(r_{22}\mathfrak{a}_2) \geq \mathcal{N}(r_{11}\mathfrak{a}_1)$ (by Lemma 4.10, Step 7 does not change the values of $\mathcal{N}(r_{11}\mathfrak{a}_1)$ and $\mathcal{N}(r_{22}\mathfrak{a}_2)$). We conclude that (using Lemma 2.6):

$$\mathcal{N}(\mathbf{b}_1) \leq {\gamma_{\mathcal{N}}}^d \cdot \min(\mathcal{N}(r_{11}\mathfrak{a}_1), \mathcal{N}(r_{22}\mathfrak{a}_2)) \leq {\gamma_{\mathcal{N}}}^d \lambda_1^{\mathcal{N}}(M).$$

It remains to show that the algorithm is polynomial time if $M \subseteq K^2$ (assuming that Algorithms 4.1 and 4.2 are polynomial time). Without loss of generality, we can assume that $M \subseteq R^2$ (we can multiply M by the smallest positive integer x such that $xM \subseteq R^2$, which increases the bit-length of the pseudo-basis by at most a polynomial factor). For this, we first prove that the number of loop iterations is polynomial. We do so by proving that $\mathcal{N}(r_{11}\mathfrak{a}_1)$ decreases by a factor $\geq 2^d$ at each iteration. As the product $\mathcal{N}(r_{11}\mathfrak{a}_1)\mathcal{N}(r_{22}\mathfrak{a}_2) = \det(M)/\Delta_K$ is constant and we stop whenever $\mathcal{N}(r_{11}\mathfrak{a}_1)$ becomes smaller than $(\gamma_{\mathcal{N}}/\mathrm{c})^d \mathcal{N}(r_{22}\mathfrak{a}_2)$, the number of iterations is bounded by $\log \mathcal{N}(r_{11}\mathfrak{a}_1)/(d \log 2)$ (for the r_{11} and $\mathfrak{a}_1$ of the input). Here, we used the fact that $\mathcal{N}(r_{11}\mathfrak{a}_1)\mathcal{N}(r_{22}\mathfrak{a}_2) \geq 1$ (because $M \subseteq R^2$) and $(\gamma_{\mathcal{N}}/\mathrm{c})^d \geq 1$, so we cannot enter the while loop if $\mathcal{N}(r_{11}\mathfrak{a}_1) < 1$.

Recall that at the end of Step 2, we have $\|r_{ii}\|_\infty \leq \mathrm{c} \cdot \mathcal{N}(r_{ii}\mathfrak{a}_i)^{1/d}$ for $i = 1, 2$. Recall also that Algorithm 4.1 outputs u, v such that $\|ur_{11} + vr_{12}\|_\infty \leq \varepsilon \|r_{11}\|_\infty$ and $\|v\|_\infty \leq C$. The new vector $\mathbf{b}_1$ at the end of the loop iteration is $u\mathbf{b}_1 + v\mathbf{b}_2$. We compute an upper bound on its algebraic norm:

$$\begin{aligned}\mathcal{N}(u\mathbf{b}_1 + v\mathbf{b}_2) &\leq (\sqrt{d})^{-d} \cdot \|u\mathbf{b}_1 + v\mathbf{b}_2\|^d = (\sqrt{d})^{-d} \cdot \left\| \begin{pmatrix} ur_{11} + vr_{12} \\ vr_{22} \end{pmatrix} \right\|^d \\ &\leq (\sqrt{d})^{-d} \cdot (\|ur_{11} + vr_{12}\| + \|vr_{22}\|)^d \\ &\leq (\|ur_{11} + vr_{12}\|_\infty + \|vr_{22}\|_\infty)^d \\ &\leq (\varepsilon\|r_{11}\|_\infty + \|v\|_\infty \cdot \|r_{22}\|_\infty)^d .\end{aligned}$$

Using the facts that the basis is strongly scaled and that the condition of Step 1 is satisfied, we have:

$$\begin{aligned}\mathcal{N}(u\mathbf{b}_1 + v\mathbf{b}_2) &\leq \mathrm{c}^d \cdot \left(\varepsilon \cdot \mathcal{N}(r_{11}\mathfrak{a}_1)^{1/d} + C \cdot \mathcal{N}(r_{22}\mathfrak{a}_2)^{1/d}\right)^d \\ &\leq \mathrm{c}^d \cdot (\varepsilon + C \cdot (\mathrm{c}/\gamma_{\mathcal{N}}))^d \cdot \mathcal{N}(r_{11}\mathfrak{a}_1).\end{aligned}$$

Now, by choice of ε and $\gamma_{\mathcal{N}}$:

$$\mathcal{N}(u\mathbf{b}_1 + v\mathbf{b}_2) \leq \mathrm{c}^d \cdot \left(\frac{1}{4\mathrm{c}} + \frac{1}{4\mathrm{c}}\right)^d \cdot \mathcal{N}(r_{11}\mathfrak{a}_1) = 2^{-d} \cdot \mathcal{N}(r_{11}\mathfrak{a}_1).$$

Recall that $\mathfrak{a}_1$ is also updated as $\mathfrak{a}_1\mathfrak{b}^{-1}$ at the end of the loop iteration. Hence, we conclude by arguing that $\mathcal{N}(\mathfrak{a}_1\mathfrak{b}^{-1}) \leq 1$. Note that $\mathcal{N}(\mathfrak{a}_1) \leq 1$ holds due to scaling, and that $\mathcal{N}(\mathfrak{b}) \geq 1$ holds because $\mathfrak{b}$ is integral. Overall, we obtain that $\mathcal{N}(r_{11}\mathfrak{a}_1)$ decreases by a factor $\geq 2^d$ during a loop iteration.

To complete the cost analysis, we observe that all the steps run in time polynomial in $\log \Delta_K$ and the bit-length of the involved elements, except maybe Step 4. In this step, it is a priori not obvious that the elements x and y satisfying the stated conditions even exist. The conditions can be re-stated as $uy \in \langle u \rangle \mathfrak{b}^{-1}$, $vx \in \langle v \rangle \mathfrak{a}^{-1}\mathfrak{b}^{-1}$ and $uy - vx = 1$. Hence such x, y exist if the ideals $\langle u \rangle \mathfrak{b}^{-1}$ and $\langle v \rangle \mathfrak{a}^{-1}\mathfrak{b}^{-1}$ are coprime. This is indeed the case, by construction of $\mathfrak{b}$. Further, we can compute x, y in polynomial time by computing a basis of the lattice spanned by the two ideals (which is R, as they are coprime). Finally, note that by ideal scaling and the fact that the quantity $\mathcal{N}(r_{11}\mathfrak{a}_1)$ decreases throughout the algorithm, all pseudo-bases occurring through the execution have bit-lengths polynomial in the input bit-length. □

Instantiating this lemma with the value of C obtained in Corollary 4.8, we can finally prove Theorem 4.1.

Proof (of Theorem 4.1). The theorem is obtained by combining Lemma 4.11 with Corollary 4.8. To apply Corollary 4.8, we need $1/\varepsilon = 2^{\widetilde{O}(\log \Delta_K)/d}$, which is indeed the case in Algorithm 4.3. Note that the choice of ε in Algorithm 4.3 only depends on K. □

We conclude this section with the proof of the full LLL-like algorithm, obtained by combining the divide-and-swap algorithm above with the reduction algorithm given in Section 3.

Proof (of Corollary 4.2). Algorithm $\mathcal{A}$ is obtained by running Algorithm 3.4 and instantiating Step 3 with Algorithm 4.3. We choose the parameter α_K of Algorithm 3.4 to be equal to $(1 + \frac{\log 2}{n}) \cdot (2^d {\gamma_{\mathcal{N}}}^{2d} \Delta_K)$, where $\gamma_{\mathcal{N}}$ is as in Theorem 4.1. By Theorem 3.8, Theorem 4.1 and Lemma 3.2, we know that at the end of the algorithm, we have a pseudo-basis of M satisfying

$$\mathcal{N}(I_1)\mathcal{N}(\mathbf{b}_1) \leq 2 \cdot (2^d {\gamma_{\mathcal{N}}}^{2d} \Delta_K)^{n-1} \cdot \lambda_1^{\mathcal{N}}(M).$$

We complete the algorithm by strongly scaling the pseudo-basis using Algorithm 4.2, which gives us

$$\|b_1\|_\infty \leq 2\mathrm{c} \cdot (2{\gamma_{\mathcal{N}}}^2 \Delta_K^{1/d})^{n-1} \cdot \lambda_1^{\mathcal{N}}(M),$$

with c as defined in Lemma 2.5. Defining $\gamma = \max(2\mathrm{c},\, 2{\gamma_{\mathcal{N}}}^2 \Delta_K^{1/d})$ and recalling that $\lambda_1^{\mathcal{N}}(M) \leq \lambda_1(M)$ (see Lemma 2.2) yields the desired bound $\|b_1\|_\infty \leq \gamma^n \cdot \lambda_1(M)$.

The run-time of the algorithm follows from Theorem 3.8, Theorem 4.1, Lemma 4.10 and our choice of α_K. □

5 Dequantizing the algorithm

In this section, we give a classical (non-quantum) variant of our LLL algorithm for module lattices, when the input module is free and provided with a basis, and if a lattice L'_K (slightly different from the lattice L_K of Corollary 4.2) has been precomputed beforehand. A basis of a free module is nothing else than a pseudo-basis $((I_i, \mathbf{b}_i))$ where all the ideals I_i are the trivial ideal R.

Free modules are an important special case for cryptography. Indeed, the decisional NTRU problem can be reduced to an approx-SVP instance in a rank-2 free module with a known basis.[9] Hence, Theorem 5.1 below shows that for lattices coming from the NTRU problem, our LLL algorithm can be dequantized (note that it still requires an oracle solving CVP in a fixed lattice).

Theorem 5.1 (Heuristic). *For any sequence of number fields K and any $\eta > 0$, there exist a sequence of lattices L'_K of dimension $O((d \log \rho(R))^{2+\eta})$, an approximation factor $\gamma = 2^{\widetilde{O}(d \log \rho(R))/d}$ and an algorithm $\mathcal{A}_c$ such that (under Heuristic 1 and the heuristics of Lemma 2.5):*

- *Algorithm $\mathcal{A}_c$ takes as input a basis of a free rank-n module $M \subset K_{\mathbb{R}}^m$, and outputs a vector $\mathbf{v} \in M$ such that $\|\mathbf{v}\|_\infty \leq \gamma^n \cdot \lambda_1(M)$;*
- *when restricted to modules contained in K^m, Algorithm $\mathcal{A}_c$ makes a number of queries to an oracle solving the closest vector problem in L'_K and requires a total number of classical operations that are both polynomial in $\log \Delta_K$ and the input bit-length.*

[9] The decisional NTRU problem consists in distinguishing $f/g \bmod q$ from uniform, when f and g are sampled in R from a parameter distribution, typically such that both $\|f\|$ and $\|g\|$ are small with overwhelming probability.

Note that Theorem 5.1 only states the existence of the lattice L'_K and the algorithm $\mathcal{A}_c$. If one wants to compute them, this can be done either in quantum polynomial time [8] or in classical sub-exponential time [6, 22] (the main bottleneck is the computation the lattice $L_{K,2}$ from Section 4.2, which requires to compute the relations between the prime ideals of the sets $\mathfrak{B}_0$ and $\mathfrak{B}$).

Proof. As in the proof of Corollary 4.2, Algorithm $\mathcal{A}_c$ is obtained by running Algorithm 3.4 and instantiating Step 3 with Algorithm 4.3, followed by one call to Algorithm 4.2 (in order to strongly-scale the output pseudo-basis). The correctness proof of Algorithm $\mathcal{A}_c$ is then exactly the same as before, and we refer the reader to the proof of Corollary 4.2 for details.

Let us now show that when the input module is free and given by a basis, then the algorithms involved in $\mathcal{A}_c$ can be run in classical polynomial time, given an oracle solving CVP in a fixed lattice L'_K. Let the lattice $L_{K,2}$ (from Section 4.2) be fixed once and for all, and let $\overline{\mathfrak{B}} = \mathfrak{B}_0 \cup \mathfrak{B}$ be the set of prime ideals used in its construction. The lattice L'_K is the direct sum of the lattice R, the lattice $L_{K,2}$ and the lattice $L'_{K,1}$ from Lemma 5.2 below. Note that, compared to the lattice used in the quantum algorithm, the lattice $L_{K,1}$ has been replaced by a lattice $L'_{K,1}$ of higher dimension. This, however, does not impact the asymptotic upper bound on the dimension of L'_K.

Let us first sketch the main ideas of the proof. In Algorithm $\mathcal{A}_c$, quantum computations are used only to decompose some ideals I as products of the form

$$I = \langle \alpha \rangle \cdot \prod_{\mathfrak{p} \in \overline{\mathfrak{B}}} \mathfrak{p}^{w_\mathfrak{p}}, \tag{5.1}$$

for some $\alpha \in K$ and $w_\mathfrak{p} \in \mathbb{Z}$. This is occurring at every iteration of Step 6 of Algorithm 4.1, and every call to the algorithm of Lemma 2.5. We call such an identity a *decomposition* of I, and we say that we know a decomposition of I if we know the element α and the integers $w_\mathfrak{p}$ of Equation (5.1).[10] We will show that if we start with a pseudo-basis $((I_1, \mathbf{b}_1), \ldots, (I_n, \mathbf{b}_n))$ of the module M for which we know a decomposition of every ideal I_i, then we can compute a decomposition of every ideal appearing during the execution of the algorithm, in classical polynomial time. The case of free modules given by a basis is an instance of the latter situation, as we always know the trivial decomposition $R = \langle 1 \rangle \cdot \prod_{\mathfrak{p} \in \overline{\mathfrak{B}}} \mathfrak{p}^0$ of the ideal R.

More rigorously, Lemmas 5.3, 5.4 and 5.5 below show that Algorithms 4.2, 4.3 and 3.4 can be run in classical polynomial time (given an oracle solving CVP in L'_K), if we are given a decompositions of every input ideal. They also show that if this is the case, then we also know a decomposition of every ideal returned by the algorithms. Since algorithm $\mathcal{A}_c$ is a combination of these three algorithms, we conclude that $\mathcal{A}_c$ can be run in classical polynomial time if given as input a basis of a free module. □

Before going through the different algorithms, we will need a stronger version of Lemma 2.5.

[10] Observe that every ideal has a decomposition, since $\overline{\mathfrak{B}}$ generates the class group, but it is unknown how to compute it in classical polynomial time. Also, note that every ideal has infinitely many decompositions, but we only need to know one of them.

Lemma 5.2 (Heuristic, [41]). *There exists a lattice $L'_{K,1}$ of dimension $O(|\overline{\mathfrak{B}}|)$ and depending only on K such that, given access to an oracle solving* CVP *in $L'_{K,1}$, the following holds. Under GRH and Heuristic 4 from [41], there exists a classical algorithm that takes as input a fractional ideal I of R and any $\alpha \in K_{\mathbb{R}}^{\times}$, and outputs $x \in \alpha I \setminus \{0\}$ such that*

- $\|x\|_\infty \leq \mathrm{c} \cdot |\mathcal{N}(\alpha)|^{1/d} \cdot \mathcal{N}(I)^{1/d}$, *where* $\mathrm{c} = 2^{\widetilde{O}(\log \Delta_K)/d}$*;*
- $\langle x \rangle \cdot (\alpha I)^{-1}$ *is $\overline{\mathfrak{B}}$-smooth.*

If $\alpha \in K$ and if the ideal I is given with a known decomposition $I = \langle \alpha' \rangle \cdot \prod_{\mathfrak{p} \in \overline{\mathfrak{B}}} \mathfrak{p}^{w_\mathfrak{p}}$, then the algorithm makes a single call to the oracle solving CVP in $L'_{K,1}$ and performs a number of classical operations that is polynomial in $\log \Delta_K$ and the input bit-length.

As for Lemma 2.5, this lemma can be obtained from the results in [41]. To do so, observe that the lattice $L'_{K,1}$ of the lemma (called L in [41, Section 3.1]) can be instantiated with any set of prime ideals of cardinality at least $\max(\log h_K, \log \Delta_K + d \log\log \Delta_K)$ generating the class group (this is required for Heuristic 4 of [41]). These conditions are satisfied by our set $\overline{\mathfrak{B}}$, which we therefore use to instantiate the lattice. The fact that the algorithm can be run in polynomial classical time when a decomposition of I is known and that $\langle x \rangle \cdot (\alpha I)^{-1}$ is $\overline{\mathfrak{B}}$-smooth then follows from a careful examination of Algorithm 3.2 in [41]. We note that the only difference between this new lattice $L'_{K,1}$ and the lattice $L_{K,1}$ of Lemma 2.5 lies in the choice of the factor base: in $L_{K,1}$, we kept the choice of [41] which aims at making the dimension of the lattice small, whereas in $L'_{K,1}$, we force the factor base to be the same as the one of $L_{K,2}$.

We now consider the strongly scaling algorithm (Algorithm 4.2).

Lemma 5.3. *Let $((I_i, \mathbf{b}_i))_{i \leq n}$ be a pseudo-basis of a module $M \subset K^m$, and assume that we know a decomposition of every ideal I_i. Then, on input $((I_i, \mathbf{b}_i))_{i \leq n}$, Algorithm 4.2 makes n calls to an oracle solving CVP in the fixed lattice $L'_{K,1}$ of Lemma 5.2 and performs a number of classical operations that is polynomial in $\log \Delta_K$ and the input bit-length. Moreover, one can efficiently and classically compute a decomposition of every ideal appearing during the execution of the algorithm.*

Proof. The only quantum step of Algorithm 4.2 is the call to Lemma 2.5 in Step 2. However, as we have seen in Lemma 5.2, this step can be performed in classical polynomial time (provided we have a CVP oracle in $L'_{K,1}$), if we know a decomposition of every ideal of the input pseudo-basis. Hence Algorithm 4.2 can also be performed in classical polynomial time.

Regarding the newly created ideals $I'_i = \langle x_i \rangle^{-1} \cdot I_i$, we can obtain decompositions of them from the decompositions of the I_i's, by multiplying the principal ideal involved in the decomposition of I_i by $\langle x_i^{-1} \rangle$ (since the x_i's are known). □

We now consider the divide-and-swap algorithm for modules of rank 2.

Lemma 5.4. *Let $((\mathfrak{a}_1, \mathbf{b}_1), (\mathfrak{a}_2, \mathbf{b}_2))$ be a pseudo-basis of a module $M \subset K^2$, and assume we know a decomposition of $\mathfrak{a}_1$ and a decomposition of $\mathfrak{a}_2$. Then, on input $((\mathfrak{a}_1, \mathbf{b}_1), (\mathfrak{a}_2, \mathbf{b}_2))$, Algorithm 4.3 makes a number of calls to an oracle solving CVP in L'_K and performs a number of classical operations that are both polynomial* *in* $\log \Delta_K$ *and the input bit-length. Moreover, one can efficiently and classically compute a decomposition of every ideal appearing during the execution of the algorithm.*

Proof. Let us start by proving that we can efficiently compute a decomposition of every ideal appearing during the execution of the algorithm. First, recall that by Lemma 5.3, the ideals returned by the call to the strongly scaling algorithm have a known decomposition. Second, we note that given a decomposition of two ideals, one can easily compute a decomposition of the product or of the inverse of the ideals (for instance, if $\mathfrak{a}_1 = \langle \alpha_1 \rangle \prod_{\mathfrak{p} \in \overline{\mathfrak{B}}} \mathfrak{p}^{w_{\mathfrak{p},1}}$ and $\mathfrak{a}_2 = \langle \alpha_2 \rangle \prod_{\mathfrak{p} \in \overline{\mathfrak{B}}} \mathfrak{p}^{w_{\mathfrak{p},2}}$, then $\mathfrak{a}_2 \cdot \mathfrak{a}_1^{-1} = \langle \alpha_2 \cdot \alpha_1^{-1} \rangle \prod_{\mathfrak{p} \in \overline{\mathfrak{B}}} \mathfrak{p}^{w_{\mathfrak{p},2} - w_{\mathfrak{p},1}}$).

At Step 4, we argue that the integral ideal $\langle v \rangle \mathfrak{a}^{-1}$ is $\overline{\mathfrak{B}}$-smooth. Since $\mathfrak{b}$ divides $\langle v \rangle \mathfrak{a}^{-1}$, it will also be $\overline{\mathfrak{B}}$-smooth and since the cardinality of $\overline{\mathfrak{B}}$ is polynomial, we can efficiently compute a decomposition by exhaustive search of prime divisors. To see that $\langle v \rangle \mathfrak{a}^{-1}$ is $\overline{\mathfrak{B}}$-smooth, we need to go back to Algorithm 4.1. By Lemma 5.2, we know that $\langle v \rangle \cdot I^{-1}$ is $\overline{\mathfrak{B}}$-smooth (see Step 12). Furthermore, by definition of I, it holds that $I \cdot \mathfrak{a}^{-1}$ is $\overline{\mathfrak{B}}$-smooth (see Step 11). Hence, multiplying both we see that $\langle v \rangle \cdot \mathfrak{a}^{-1}$ is $\overline{\mathfrak{B}}$-smooth, as required.

Finally, we must also examine the ideals created in Algorithm 4.1 (which is called at Step 3 of Algorithm 4.3). The only ideal created in the algorithm is the ideal I defined at Step 11, for which we know a decomposition, since we know a decomposition of $\mathfrak{a}$ and the integers $w_{t,j} - w_{s,j}$ and $w'_{s,j}$. We conclude that all the ideals created by Algorithm 4.3 can be created together with a known decomposition.

We now show that these decompositions allow us to run the algorithm in classical polynomial time. First, we observe that the only steps of Algorithm 4.3 that might require a quantum computer are the calls to Algorithms 4.2 and 4.1; all the other steps can be performed efficiently with a classical computer. Recall that the case of Algorithms 4.2 has already been handled in Lemma 5.3, hence we are left with Algorithm 4.1.

In Algorithm 4.1, Steps 1, 2 and 12 can be performed classically thanks to Lemma 5.2. In Step 6, we want to compute $(w_{t,j})_{j \le r_0}$ and g_t such that $\mathfrak{a}^{-1} = \prod_j \mathfrak{p}_j^{w_{t,j}} \langle g_t \rangle$. By assumption, we already know a decomposition of the form $\mathfrak{a}^{-1} = \langle \alpha \rangle \prod_{\mathfrak{p} \in \overline{\mathfrak{B}}} \mathfrak{p}^{w_\mathfrak{p}}$. This decomposition involves all prime ideals in $\overline{\mathfrak{B}}$ whereas we would like to have a decomposition involving only the ideals in $\mathfrak{B}_0$. Since we know the lattice $L_{K,2}$, we see from its definition in Section 4.2 that a decomposition of every ideal $\mathfrak{q}_j \in \mathfrak{B}$ over $\mathfrak{B}_0$ is $\mathfrak{q}_j = \langle g_{\nu - r + j} \rangle \prod_i \mathfrak{p}_i^{-(\mathbf{w}_{\nu - r + j})_i}$: this leads us to the desired decomposition of $\mathfrak{a}^{-1}$ over $\mathfrak{B}_0$. All other steps of Algorithm 4.1 can be performed efficiently with a classical computer, and an oracle solving CVP in R and in $L_{K,2}$. This completes the proof. ◻

We can now prove that the full LLL algorithm from Section 3.2 can be performed in classical polynomial time if we start with a pseudo-basis where the ideals have known decompositions.

Lemma 5.5. *Let* $((I_i, \mathbf{b}_i))_{i \le n}$ *be a pseudo-basis of a module* $M \subset K^m$*, and assume that we know a decomposition of every* I_i*. Then, on input* $((I_i, \mathbf{b}_i))_{i \le n}$*, Algorithm 3.4 makes a number of calls to an oracle solving CVP in* L'_K *and performs a number of classical operations that are both polynomial in* $\log \Delta_K$ *and the input bit-length.*

Moreover, one can efficiently and classically compute a decomposition of every ideal appearing during the execution of the algorithm.

Proof. We consider a slightly modified version of Algorithm 3.4. We implement Steps 3 to 5 using Algorithm 4.3 (instead of only Step 3). Indeed, Step 5 of Algorithm 3.4 might be problematic, since Lemma 2.8 does not tell us anything about the newly constructed ideals I'_i and I'_{i+1}. In particular, it might be hard to compute decompositions for them. However, observe that the purpose of Steps 3 to 5 of Algorithm 3.4 is to construct a pseudo-basis $((I'_i, \mathbf{a}'_i), (I'_{i+1}, \mathbf{a}'_{i+1}))$ of the module M_i such that the first vector satisfies $\mathcal{N}(\mathbf{a}'_i) \leq {\gamma_{\mathcal{N}}}^d \cdot \lambda_1^{\mathcal{N}}(M_i)$. This is done by calling an algorithm that outputs a short vector of M_i in Step 3 and then Steps 4 and 5 are used to reconstruct a pseudo-basis having this short vector as first vector. Looking at Algorithm 4.3, one can see that it performs the computation of the desired pseudo-basis, but outputs only the first vector to match with the requirement of Step 3. We could instead output in Algorithm 4.3 the full pseudo-basis of the module, and factor Steps 3 to 5 of Algorithm 3.4. Doing so, and using Lemma 5.4, this shows that we can classically compute a decomposition of the new ideals I'_i and I'_{i+1}, provided we knew a decomposition of the input ideals I_i and I_{i+1}.

Apart from these steps, new ideals are constructed in Step 7 of Algorithm 3.4, with the call to Algorithm 3.2 (the scaling algorithm). As for Algorithm 4.2 (the strongly scaling algorithm), we can efficiently compute a decomposition of the ideals obtained after scaling. Hence all the ideals appearing during the execution of this variant of Algorithm 3.4 can be computed together with a decomposition.

Finally, the only potentially quantum part of Algorithm 3.4 is the call to the divide-and-swap algorithm (Algorithm 4.3). Lemma 5.4 tells us that this algorithm can be run in classical polynomial time if the input ideals have known decompositions, which we assumed given. We conclude that Algorithm 3.4 can be run in classical polynomial time. □

6 Numerical experiments

In this section, we provide some numerical experiments confirming our heuristic arguments of Section 4.3. The code used to run these experiments is available at `https://github.com/apelletm/code_module_LLL`. We first tested experimentally the assumption we made on the distribution of the first coordinate of the vectors (which is related to the norm of the ideals). We show below that our assumption on the asymptotic behaviour of $\Pr(|\sum_{j>0.04\cdot B} \mathbf{w}'_{s,j} \log \mathcal{N}(\mathfrak{q}_j) - b'_t| \leq \alpha\sqrt{d})$ seems reasonable. We then performed experiments to test Heuristic 1 as a whole. Because the complexity of these experiments grows as 2^{d^2} with d the degree of the number field, we could not test the heuristic on number fields of large degree. For the number fields of small degree we tested, Heuristic 1 seems to hold.

6.1 Testing the distribution of the norm

We provide some experimental results justifying our assumption that

$$\Pr(|\sum_{j>0.04\cdot B} \mathbf{w}'_{s,j} \log \mathcal{N}(\mathfrak{q}_j) - b'_t| \leq \alpha\sqrt{d}) \geq \frac{\alpha\sqrt{d}}{B\delta},$$

for any target b'_t satisfying $|b'_t| \leq \delta$. Write $p_\ell = \Pr(|\sum_{j>0.04 \cdot B} \mathbf{w}'_{s,j} \log \mathcal{N}(\mathfrak{q}_j) - b'_t| \leq \ell)$. We first checked that for a fixed number field and choice of B and δ, this probability is proportional to ℓ (for small ℓ's). To do so, we used the following procedure:

– select ℓ in some interval;
– choose a random target b'_t uniformly in $[-\delta, \delta]$;
– sample 500 000 points according to the distribution of $\sum_j \mathbf{w}'_{s,j} \log \mathcal{N}(\mathfrak{q}_j)$;
– count those that are at distance at most ℓ of b'_t.

This gives us an empirical probability p_ℓ. One can have in mind that ℓ should be small, as $\alpha\sqrt{d} = O(d^{-3/2} \log d)$. We then computed it for different values of ℓ ranging in $[0.001, 0.01]$, $[0.01, 0.1]$ and $[0.1, 1]$, to check that the our assumption on the proportionality still looked reasonable for different orders of magnitude of ℓ. The results are plotted in Figures 6.1, 6.2 and 6.3 for a power of two cyclotomic field, a cyclotomic field which is not a prime power, and a "random" number field (the coefficients of the defining polynomial being chosen uniformly at random between -4 and 4). One can observe that the probabilities are indeed proportional the length of the interval ℓ, and that the proportionality factor is roughly the same for the three different ranges of ℓ.

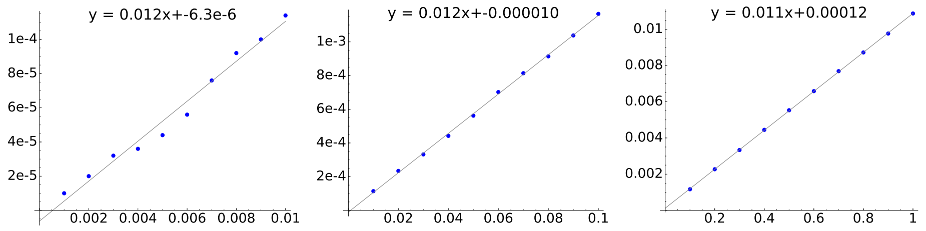

Fig. 6.1. Empirical probability p_ℓ as a function of ℓ in a cyclotomic field of conductor 64 (degree 32)

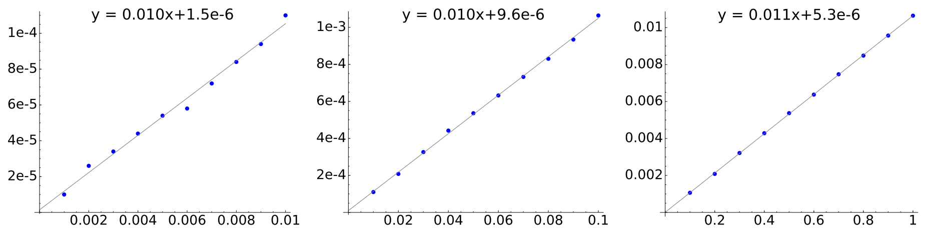

Fig. 6.2. Empirical probability p_ℓ as a function of ℓ in a cyclotomic field of conductor 100 (degree 40)

These results suggest that assuming $p_\ell = c \cdot \ell$ for some c is reasonable. It remains to check the asymptotic behavior of the proportionality factor c compared to $1/(B\delta)$ We hence computed an empiric c for different cyclotomic number fields of increasing degree.[11] We set $B = d$ and $\delta = d^2 \log(d^2)$, as these are roughly the values that we

[11] We restricted to cyclotomic fields to allow faster experiments.

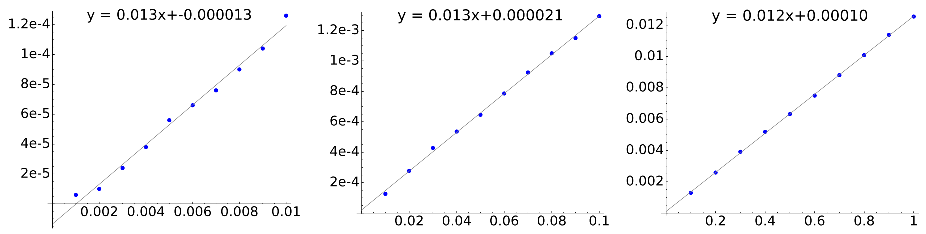

Fig. 6.3. Empirical probability p_ℓ as a function of ℓ in a "random" number field of degree 32

want to choose later on in our algorithm. The number of points used to compute the empirical probabilities p_ℓ was set to 10 000 and we chose ℓ ranging in $[0.1, 1]$ to compute the slope. Then, we computed $1/c$ and $B\delta$ and observed that $1/c$ is indeed smaller than $B\delta$ (see Table 1). We also plotted the values of $1/c$ as a function of $B\delta$ in Figure 6.4. The results suggests that the value $1/c$ increase linearly with $B\delta$, with a slope smaller than 1. Therefore it seems reasonable to assume that the inequality $1/c \leq B\delta$ holds asymptotically.

d	8	16	20	24	32	36	40	44	48	54	56	60	64	66	70
m	16	32	44	35	64	57	100	69	65	81	87	99	128	67	71
$B \cdot \delta$	64	168	225	284	409	475	541	609	678	783	819	891	964	1000	1074
$1/c$	33	87	39	93	67	124	97	116	99	113	142	142	162	159	189

Table 1. Empirical values of $1/c$ for different cyclotomic number fields of conductor m and degree d

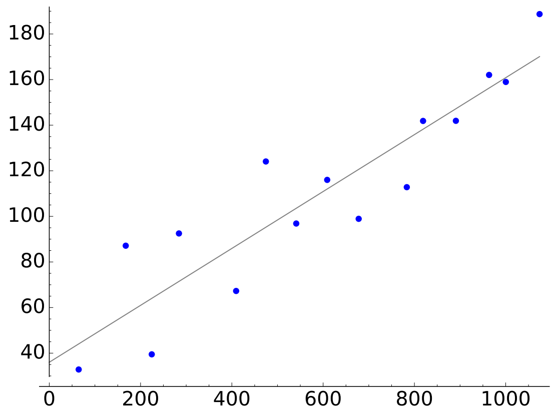

Fig. 6.4. Evolution of $1/c$ (computed empirically) as a function of $B\delta$

6.2 Testing the validity of Heuristic 1 in cyclotomic fields of small dimension

We also tested the validity of Heuriscic 1 as a whole. We tried to test it with parameters relevant to its use in Algorithm 4.3, which means that r should be quadratic in the dimension d of the number fields. As we needed to solve CVP instances in a lattice of dimension larger than r, this means that we could only test number fields of degree up to $d = 8$.

For each degree $d = 4, 6, 8$, we tested 3 different number fields of degree d, and for every number field, we tested different values of r, ranging from roughly d^2 to 200. Once d and r were fixed, we defined $\delta = \log(r)/\log(d)$ and $B = 10 \cdot \max(\log h_K, 1) \cdot \max(\delta_0/\delta, 1)$. Compared to the lower bound on B given in Heuristic 1, the constant 100 has been replaced by 10 (because we want $r \geq B$ and we cannot deal with r larger than 200). We also added the max with 1, because in the special case we consider, the rings are principal, and so $h_K = 1$ and $\delta_0 = 0$. We then defined α_0 and β as in Heuristic 1, except that the constant 0.96 in α_0 was replaced by 1 and the constant 0.01 is β was replaced by 0.1. The constraint $\alpha_0 \leq \frac{\log d}{12 d^{2.5}}$ was never satisfied in our experiments, as it would have required r to be much larger. However, the fact that this constraint is not satisfied should only make the distance between the target points and the lattice larger, as it was used to ensure that the space of solutions was not wrapped up modulo Λ (if the space is folded, it should decrease the number of solutions).

Once all these parameters were computed, we constructed the lattice $L_{K,2}$ as in Heuristic 1. We then sampled 90 target points $\mathbf{t}_i$ of the desired shape (i.e., in the span on $L_{K,2}$ and with their last r coordinates of Euclidean norm at most $0.01B/\sqrt{r}$), and computed the maximum distance of these points to the lattice $L_{K,2}$ (by solving CVP in $L_{K,2}$).

We plot on Figure 6.5 the evolution the maximum distance $\max_i \operatorname{dist}(\mathbf{t}_i, L_{K,2})$ (in blue) and $\sqrt{B}$ (in red) as a function of r. The points on the curves are obtained by taking the average value for the three number fields with the same degree d (these values where very close, and taking the average smooths slightly the curves). One can observe that $\max_i \operatorname{dist}(\mathbf{t}_i, L_{K,2})$ is indeed smaller than $\sqrt{B}$ (so in particular smaller than $\sqrt{(1+\varepsilon)B}$ for any $\varepsilon > 0$), and that this seems to stabilize and hold asymptotically.

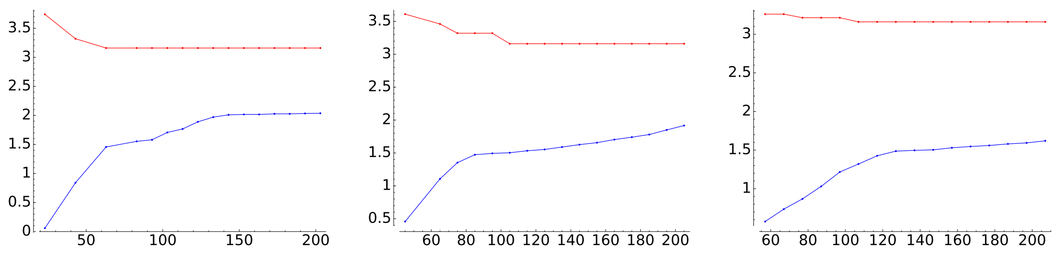

Fig. 6.5. Evolution of $\max_i \operatorname{dist}(\mathbf{t}_i, L_{K,2})$ (in blue) and $\sqrt{B}$ (in red) as a function of r. From left to right: number fields of degree 4, 6 and 8.

Table 2 below gives more details on the parameters of the experiments. For readability, we give the values only for one number field per degree d, but the values for the other number fields of the same degree were very similar. In the table, we provide the values of $\max_i \operatorname{dist}(\mathbf{t}_i, L_{K,2})$ and $\sqrt{B}$ as plotted above, but also the values of the parameters α_0 and β. One can observe, as was mentioned above, that the values of α_0 we were able to reach never satisfy the constraint $\alpha_0 \leq \frac{\log d}{12d^{2.5}}$. In particular, in Algorithm 4.3, we use Heuristic 1 with α_0 very small (of the order of $1/d^{\log d}$, as we will have ε of this order of magnitude) and hence $\beta = \frac{1}{\alpha_0}\sqrt{\frac{0.01B}{2d}}$ should be quite large (slightly more than polynomial in d). In the experiments, one can see that for degree 6 and 8, we were only able to reach $\beta = 2$, which is not very realistic for our concrete choice of parameters. In the case of degree 4, we reached more interesting values of β (up to 16.7, which is roughly $d^{\log d}$).

The table also contains intervals that we called 'B in practice'. For a specific target vector t_i, the value 'B in practice' corresponds to the ℓ_1 norm of the last r coordinates of v_i, the closest point to t_i in $L_{K,2}$. Recall that in our analysis, we heuristically assumed that the closest point to t_i would have at most B non-zero coefficients (equal to 1 or -1) in its last r coordinates. We experimentally observed that, indeed, the non-zero coefficients are 1 or -1, but it seems that the closest point has significantly less than B non-zero coefficients. This does not invalidate the heuristic, but it seems to say that the distance between t_i and $L_{K,2}$ might be less that what we expected (and this is indeed what we observe when we compare $\max_i \operatorname{dist}(\mathbf{t}_i, L_{K,2})$ and $\sqrt{B}$). The interval given in the table is the smallest interval containing all the 'B in practice' for all the 90 target points. One can observe that for degree 6 and 8, the numbers 'B in practice' remain very small (between 0 and 2), meaning that the closest point can be obtained with a linear combination of a small number of the basis vectors (at most 9 vectors among the 200 vectors of the basis). This is probably the reason why the CVP solver is so fast and we can use it in lattices of dimension 200. In the case of degree 4, the number 'B in practice' increases up to 4, and the run-time of the CVP solver degrades consequently.

Overall, the experiments seem consistent with the heuristic for these cyclotomic fields, but this is not very satisfactory as we could test only very small cyclotomic number fields. It could be that the small degree of the number fields (and the fact that they are principal) changes completely the behaviour of the heuristic. Also, the parameter setting of our algorithm is asymptotic (with hidden constants and logarithms in the $\widetilde{O}$), so testing it for only 3 number fields of very small degree does not say a lot. Our code is open access and has been done to handle any number field, so it could be used to create the lattice $L_{K,2}$ for much larger number fields, but then the CVP step would be very costly.

Acknowledgments. We thank Léo Ducas and Guillaume Hanrot for helpful and constructive discussions. We thank Thomas Espitau for helpful discussions about the state of the art. The first author received support from Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Education (2019R1A6A3A03033778) and from the LABEX MILYON (ANR-10-LABX-0070) of Université de Lyon, within the program "Investissements d'Avenir" (ANR-11-IDEX-0007) operated by the French National Research Agency

d	r	m	α_0	β	B in practice	$\max_i \text{dist}(\mathbf{t}_i, L_{K,2})$	$\sqrt{B}$
4	20	5	20.5	0.0204	$[0,0]$	0.0551	3.74
4	40	5	1.21	0.306	$[0,0]$	0.760	3.32
4	60	5	0.403	0.878	$[0,2]$	1.46	3.16
4	80	5	0.200	1.77	$[0,2]$	1.61	3.16
4	100	5	0.116	3.06	$[0,3]$	1.75	3.16
4	120	5	0.0740	4.78	$[2,3]$	1.95	3.16
4	140	5	0.0508	6.96	$[2,4]$	2.01	3.16
4	160	5	0.0366	9.66	$[2,4]$	2.02	3.16
4	180	5	0.0274	12.9	$[2,4]$	2.03	3.16
4	200	5	0.0212	16.7	$[2,4]$	2.04	3.16
6	40	9	2.25	0.146	$[0,0]$	0.471	3.61
6	60	9	0.919	0.344	$[0,0]$	1.09	3.46
6	80	9	0.537	0.564	$[0,2]$	1.46	3.32
6	100	9	0.388	0.743	$[0,2]$	1.50	3.16
6	120	9	0.289	1.00	$[0,2]$	1.54	3.16
6	140	9	0.224	1.29	$[0,2]$	1.63	3.16
6	160	9	0.180	1.60	$[0,2]$	1.76	3.16
6	180	9	0.149	1.94	$[0,2]$	1.75	3.16
6	200	9	0.125	2.30	$[2,3]$	1.94	3.16
8	60	16	1.40	0.178	$[0,0]$	0.745	3.16
8	80	16	0.987	0.253	$[0,0]$	1.06	3.16
8	100	16	0.751	0.333	$[0,1]$	1.35	3.16
8	120	16	0.601	0.416	$[0,2]$	1.49	3.16
8	140	16	0.498	0.502	$[0,2]$	1.49	3.16
8	160	16	0.422	0.592	$[0,2]$	1.56	3.16
8	180	16	0.366	0.684	$[0,2]$	1.57	3.16
8	200	16	0.321	0.778	$[0,2]$	1.62	3.16

Table 2. Experimental values of the different parameters of Heuristic 1, observed in cyclotomic fields of conductor m and degree d, for different values of r.

(ANR). The second author was supported in part by CyberSecurity Research Flanders with reference number VR20192203 and by the Research Council KU Leuven grant C14/18/067 on Cryptanalysis of post-quantum cryptography. The third author was supported in part by BPI-France in the context of the national project RISQ (P141580) and by the European Union PROMETHEUS project (Horizon 2020 Research and Innovation Program, grant 780701). The third and last authors were supported in part by the ERC Starting Grant ERC-2013-StG-335086-LATTAC. Part of this work was done while Changmin Lee, Alice Pellet-Mary and Damien Stehlé were visiting the Simons Institute for the Theory of Computing.

References

[1] M. Ajtai. Generating hard instances of lattice problems. In *STOC*, 1996.

[2] M. Ajtai. The shortest vector problem in l_2 is NP-hard for randomized reductions. In *STOC*, 1998.

[3] M. R. Albrecht and A. Deo. Large Modulus Ring-LWE $\geq$ Module-LWE. In *ASIACRYPT*, 2017.

[4] E. Bach and J. O. Shallit. *Algorithmic Number Theory: Efficient Algorithms.* MIT Press, 1996.

[5] J.-F. Biasse, T. Espitau, P.-A. Fouque, A. Gélin, and P. Kirchner. Computing generator in cyclotomic integer rings. In *EUROCRYPT*, 2017.

[6] J.-F. Biasse and C. Fieker. Subexponential class group and unit group computation in large degree number fields. *LMS J Comput Math*, 2014.

[7] 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. *J Symb Comput*, 2017.

[8] 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 *SODA*, 2016.

[9] W. Bosma and M. Pohst. Computations with finitely generated modules over Dedekind domains. In *ISSAC*, 1991.

[10] Z. Brakerski, C. Gentry, and V. Vaikuntanathan. (Leveled) fully homomorphic encryption without bootstrapping. *ToCT*, 2014.

[11] J. Buchmann. Reducing Lattice Bases by Means of Approximations. In *ANTS*, 1994.

[12] T. Camus. *Méthodes algorithmiques pour les réseaux algébriques.* PhD thesis, Université Grenoble Alpes, 2017.

[13] J.-P. Cerri. *Spectres euclidiens et inhomogènes des corps de nombres.* PhD thesis, Université Henri Poincaré, Nancy, 2005.

[14] H. Cohen. *A Course in Computational Algebraic Number Theory.* Springer, 1995.

[15] H. Cohen. Hermite and Smith normal form algorithms over Dedekind domains. *Math Comp*, 1996.

[16] R. Cramer, L. Ducas, and B. Wesolowski. Short Stickelberger class relations and application to ideal-SVP. In *EUROCRYPT*, 2017.

[17] C. Fieker. *Über relative Normgleichungen in älgebraischen Zahlkörpern.* PhD thesis, TU Berlin, 1997.

[18] C. Fieker and M. E. Pohst. Lattices over number fields. In *ANTS*, 1996.

[19] C. Fieker and M. E. Pohst. Dependency of units in number fields. *Math Comp*, 2006.

[20] C. Fieker and D. Stehlé. Short bases of lattices over number fields. In *ANTS*, 2010.

[21] Y. H. Gan, C. Ling, and W. H. Mow. Complex lattice reduction algorithm for low-complexity full-diversity MIMO detection. *IEEE Trans Signal Processing*, 2009.

[22] A. Gélin. *Calcul de groupes de classes d'un corps de nombres et applications à la cryptologie.* PhD thesis, Paris 6, 2017.

[23] J. Hoffstein, J. Pipher, and J. H. Silverman. NTRU: a ring based public key cryptosystem. In *ANTS*, 1998.

[24] A. Hoppe. *Normal forms over Dedekind domains, efficient implementation in the computer algebra system KANT.* PhD thesis, TU Berlin, 1998.

[25] R. Kannan. Minkowski's convex body theorem and integer programming. *Math Oper Res*, 1987.

[26] T. Kim and C. Lee. Lattice reductions over euclidean rings with applications to cryptanalysis. In *IMACC*, 2017.

[27] T. Laarhoven. Sieving for closest lattice vectors (with preprocessing). In *SAC*, 2016.

[28] A. Langlois and D. Stehlé. Worst-case to average-case reductions for module lattices. *Des Codes Cryptography*, 2015.

[29] C. Lee, A. Pellet-Mary, D. Stehlé, and A. Wallet. An lll algorithm for module lattices. In *International Conference on the Theory and Application of Cryptology and Information Security*, pages 59–90. Springer, 2019.

[30] A. K. Lenstra, H. W. Lenstra, Jr., and L. Lovász. Factoring polynomials with rational coefficients. *Math Ann*, 1982.

[31] P. Lezowski. Computation of the euclidean minimum of algebraic number fields. *Math Comp*, 83(287):1397–1426, 2014.

[32] V. Lyubashevsky and D. Micciancio. Generalized compact knapsacks are collision resistant. In *ICALP*, 2006.

[33] V. Lyubashevsky, C. Peikert, and O. Regev. On ideal lattices and learning with errors over rings. In *EUROCRYPT*, 2010.

[34] D. Micciancio. The hardness of the closest vector problem with preprocessing. *Trans Inf Theory*, 2001.

[35] D. Micciancio and S. Goldwasser. *Complexity of lattice problems: a cryptographic perspective.* Kluwer Academic Press, 2002.

[36] H. Minkowski. *Gesammelte Abhandlungen.* Chelsea, New York, 1967.

[37] H. Napias. A generalization of the LLL-algorithm over Euclidean rings or orders. *J théorie des nombres de Bordeaux*, 1996.

[38] J. Neukirch. Algebraic number theory. In *Grundlehren der Mathematischen Wissenschaften*, volume 322. Springer, 1999.

[39] O. T. O'Meara. *Introduction to Quadratic Forms.* Springer, 1963.

[40] C. Peikert and A. Rosen. Efficient collision-resistant hashing from worst-case assumptions on cyclic lattices. In *TCC*, 2006.

[41] A. Pellet-Mary, G. Hanrot, and D. Stehlé. Approx-SVP in ideal lattices with pre-processing. In *EUROCRYPT*, 2019.

[42] O. Regev. On lattices, learning with errors, random linear codes, and cryptography. *J ACM*, 2009.

[43] M. Rosca, D. Stehlé, and A. Wallet. On the Ring-LWE and Polynomial-LWE problems. In *EUROCRYPT*, 2018.

[44] Saruchi, I. Morel, D. Stehlé, and G. Villard. LLL reducing with the most significant bits. In *ISSAC*, 2014.

[45] C.-P. Schnorr and M. Euchner. Lattice basis reduction: improved practical algorithms and solving subset sum problems. *Math Progr*, 1994.

[46] D. Stehlé, R. Steinfeld, K. Tanaka, and K. Xagawa. Efficient public key encryption based on ideal lattices. In *ASIACRYPT*, 2009.