



**HAL**  
open science

## Reducing number field defining polynomials: an application to class group computations

Alexandre Gélin, Antoine Joux

► **To cite this version:**

Alexandre Gélin, Antoine Joux. Reducing number field defining polynomials: an application to class group computations. Algorithmic Number Theory Symposium XII, Aug 2016, Kaiserslautern, Germany. pp.315–331 10.1112/S1461157016000255 . hal-01362144

**HAL Id: hal-01362144**

**<https://hal.science/hal-01362144v1>**

Submitted on 6 Mar 2017

**HAL** is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

# Reducing number field defining polynomials: An application to class group computations

Alexandre Gélin and Antoine Joux

## ABSTRACT

In this paper, we describe how to compute smallest monic polynomials that define a given number field  $\mathbb{K}$ . We make use of the one-to-one correspondence between monic defining polynomials of  $\mathbb{K}$  and algebraic integers that generate  $\mathbb{K}$ . Thus, a smallest polynomial corresponds to a vector in the lattice of integers of  $\mathbb{K}$  and this vector is short in some sense. The main idea is to consider weighted coordinates for the vectors of the lattice of integers of  $\mathbb{K}$ . This allows us to find the desired polynomial by enumerating short vectors in these weighted lattices. In the context of the subexponential algorithm of Biasse and Fieker for computing class groups, this algorithm can be used as a precomputation step that speeds up the rest of the computation. It also widens the applicability of their faster conditional method – which requires a defining polynomial of small height – to a much larger set of number field descriptions.

## 1. Introduction

Computing the class group and the regulator of a number field are two major tasks in algorithmic algebraic number theory. The first results only addressed quadratic number fields: Shanks [15, 16] described an algorithm based on the baby-step-giant-step method which computes the class group and the regulator in exponential runtime  $O(|\Delta_{\mathbb{K}}|^{\frac{1}{5}})$  under the extended Riemann hypothesis, where  $\Delta_{\mathbb{K}}$  denotes the absolute discriminant of the considered number field.

The first subexponential algorithm, proposed by Hafner and McCurley [8], computes the class group structure of an imaginary quadratic number field in heuristic time  $L_{|\Delta_{\mathbb{K}}|}(\frac{1}{2}, \sqrt{2})$ . This  $L$ -notation is classical when presenting index calculus algorithms with subexponential complexity. Given two constants  $\alpha$  and  $c$  with  $\alpha \in [0, 1]$  and  $c \geq 0$ ,  $L_N(\alpha, c)$  is used as a shorthand for:

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

where  $o(1)$  tends to 0 as  $N$  tends to infinity.

In [3], Buchmann extended this method to all number fields, obtaining a heuristic complexity of  $L_{|\Delta_{\mathbb{K}}|}(\frac{1}{2}, 1.7)$ , for any arbitrary but fixed number field degree. Recently, Biasse and Fieker [2] obtained a subexponential complexity for all number fields, without restriction on the extension degree. They achieved a  $L_{|\Delta_{\mathbb{K}}|}(\frac{2}{3} + \varepsilon, O(1))$  complexity<sup>†</sup> in the general case and  $L_{|\Delta_{\mathbb{K}}|}(\frac{1}{2}, O(1))$  when the extension degree  $n$  satisfies the inequality  $n \leq (\log |\Delta_{\mathbb{K}}|)^{3/4-\varepsilon}$ . Furthermore, for some restricted classes of number fields they achieve an even better  $L_{|\Delta_{\mathbb{K}}|}(a, O(1))$  complexity with  $a$  possibly as low as  $\frac{1}{3}$ . More precisely, they achieve this improved complexity when the polynomial used to define the number field has small coefficients compared to the discriminant of the field.

*Contribution.* Our main result is an algorithm that given a number field, described by some defining polynomial, computes a reduced polynomial, i.e. a monic polynomial with minimal

---

This work has been supported in part by the European Union's H2020 Programme under grant agreement number ICT-644209.

<sup>†</sup>For an arbitrary small  $\varepsilon > 0$ .

coefficients defining this number field. The complexity of the reduction algorithm is of the form  $L_{|\Delta_{\mathbb{K}}|}$  where the first constant depends on the extension degree  $n$  and the second one on the size of the coefficients of any minimal defining polynomial<sup>‡</sup>. We are especially interested in the fields for which it is possible to find a reduced polynomial whose coefficients are small compared to the field discriminant. Indeed, for these fields, using a reduced polynomial allows the algorithm of Biasse and Fieker to run in time  $L_{|\Delta_{\mathbb{K}}|}(a, O(1))$  with  $\frac{1}{3} \leq a < \frac{1}{2}$ . Thus, by running our algorithm as a preliminary step to theirs, we generalize their approach, with its reduced complexity, to a wider class of number field descriptions.

*Outline.* The article is organized as follows. In Section 2, we introduce the necessary notation and tools. Section 3 presents our motivations for finding such an algorithm in the context of class group computations. In Section 4, we present the current state of the art for reducing number field defining polynomials, namely the algorithm of Cohen and Diaz y Diaz. Our new reduction algorithm is described in Section 5 and its complexity is analyzed in Section 6. Finally, some examples of its application to class group computations are shown in Section 7.

## 2. A reminder on number fields and their defining polynomials

As usual, an algebraic number field is given as the quotient of  $\mathbb{Q}[X]$  by a preferably monic irreducible polynomial in  $\mathbb{Z}[X]$ . Let  $\mathbb{K}$  be a number field of degree  $n = [\mathbb{K} : \mathbb{Q}]$  and denote by  $\mathcal{O}_{\mathbb{K}}$  the ring of integers of  $\mathbb{K}$ . It is always possible to rewrite  $\mathbb{K}$  as  $\mathbb{Q}[\theta]$  with  $\theta$  in  $\mathcal{O}_{\mathbb{K}}$ . Since  $\theta$  is an algebraic integer, its minimal polynomial  $T$  is monic, moreover as  $\theta$  generates  $\mathbb{K}$  the degree of  $T$  is  $n$ . As a consequence we have rewritten:

$$\mathbb{K} \simeq \mathbb{Q}[X]/(T), \quad \text{with } T \text{ monic in } \mathbb{Z}[X].$$

In the sequel, we only consider such monic polynomials in  $\mathbb{Z}[X]$ , each of them corresponding to an integer that generates  $\mathbb{K}$ . We denote by  $t_k$  the coefficients of  $T$  and by  $\tau_j$  its (necessarily distinct) roots. Thus, we have:

$$T(X) = \sum_{k=0}^n t_k X^k = \prod_{j=1}^n (X - \tau_j).$$

We also denote by  $H(T) = \max_k \{|t_k|\}$  the *height* of the polynomial.

Our main goal is, given  $\mathbb{Q}[\theta]$ , to find an algebraic integer  $\theta'$  in  $\mathcal{O}_{\mathbb{K}}$  that minimizes the height of its minimal polynomial among the algebraic integers such that  $\mathbb{Q}[\theta'] = \mathbb{Q}[\theta]$ . Indeed, a defining polynomial with smaller coefficients allows for more efficient computations in the number field. A particularly relevant fact is that when bounding the norm of algebraic integers expressed as polynomials in  $\theta'$ , the height of the defining polynomial of  $\theta'$  appears in the bound.

### 2.1. Discriminants

In the context of number fields, two important notions of discriminant appear. The discriminant of the defining polynomial  $T$  denoted by  $\Delta(T)$  and the discriminant of the number field  $\mathbb{K}$  denoted by  $\Delta_{\mathbb{K}}$ . These values are related but different in general.

Given an integral basis  $\omega_1, \dots, \omega_n$  of  $\mathcal{O}_{\mathbb{K}}$ , i.e. a  $\mathbb{Z}$ -basis of the free module  $\mathcal{O}_{\mathbb{K}}$ , the discriminant of  $\mathbb{K}$  is the determinant<sup>†</sup> of the  $n \times n$  matrix  $M$  whose entries are  $M_{i,j} = \text{Tr}_{\mathbb{K}/\mathbb{Q}}(\omega_i \omega_j)$ . Let  $\sigma_1, \dots, \sigma_n$  denote the complex embeddings of  $\mathbb{K}$  in  $\mathbb{C}$ . The discriminant also can be

---

<sup>‡</sup>Note that this size is the same for all minimal defining polynomials.

<sup>†</sup>The value is independent of the choice of integral basis for  $\mathcal{O}_{\mathbb{K}}$ .

interpreted as the square of the determinant of the matrix  $B$  whose entries are  $B_{i,j} = \sigma_i(\omega_j)$ . Indeed, the trace matrix is exactly  $M = {}^tB \cdot B$ .

For a polynomial  $T = \sum t_k X^k$  of degree  $n$ , its discriminant is defined as

$$\Delta(T) = (-1)^{\frac{n(n-1)}{2}} \frac{1}{t_n} \text{Res}(T, T'),$$

where  $\text{Res}(T, T')$  is the resultant of  $T$  and its derivative. Note that, despite the fact that we only consider monic polynomials, we give here the general formula that includes the leading term  $t_n$ .

The link with the discriminant of the field  $\mathbb{K}$  comes from the fact that the discriminant of a monic polynomial  $T$  corresponds to the discriminant of the suborder<sup>†</sup> of  $\mathcal{O}_{\mathbb{K}}$  defined by  $T$ , i.e.  $\mathbb{Z}[\theta]$  where  $\theta$  is a root of  $T$ . This implies:

$$\Delta(T) = C^2 \Delta_{\mathbb{K}},$$

where  $C \in \mathbb{N}$  is the *index* of the suborder  $C = [\mathcal{O}_{\mathbb{K}} : \mathbb{Z}[\theta]]$ .

Since we desire to find polynomials with a small height that define  $\mathbb{K}$ , it is useful to find relations between the discriminant  $\Delta(T)$  – known up to the  $C^2$  factor – and the height  $H(T)$ . A first relation relies on Hadamard’s inequality. Indeed, the resultant of  $T$  and  $T'$  can be computed as the determinant of a  $(2n - 1) \times (2n - 1)$  matrix whose entries are all bounded by  $nH(T)$  in absolute value. As a consequence:

$$|\Delta_{\mathbb{K}}| \leq |\Delta(T)| \leq (n\sqrt{2n - 1} H(T))^{2n-1}. \tag{2.1}$$

### 2.2. Mahler measure

The Mahler measure [10, 11] of a polynomial  $T$ , denoted by  $M(T)$ , is defined by  $\log M(T) = \int_0^1 \log |T(e^{2i\pi t})| dt$ . Thanks to Jensen’s formula, it can also be written in the alternative form:

$$M(T) = |t_n| \prod_{j=1}^n \max(1, |\tau_j|),$$

which is more convenient for our purpose. Again, despite considering monic polynomials, we include  $t_n$  for the sake of generality.

The link between the Mahler measure and the height of a polynomial is quite tight and can be illustrated by the two following inequalities. First, Mahler shows in [11] that for all  $k \in \{0, \dots, n\}$ , we have  $|t_k| \leq \binom{n}{k} M(T)$  thus:

$$H(T) \leq \binom{n}{\lfloor \frac{n}{2} \rfloor} M(T) \leq 2^n M(T) \tag{2.2}$$

In addition, it is proven in [13] that:

$$M(T) \leq \left( \sum_{i=0}^n |t_i|^2 \right)^{\frac{1}{2}} \quad \text{which implies} \quad M(T) \leq \sqrt{n+1} H(T). \tag{2.3}$$

Using the Mahler measure, it is possible to refine the bound on  $|\Delta(T)|$  we derived from Hadamard’s inequality (see [12, Theorem 1]) and to obtain:

$$|\Delta(T)| \leq n^n M(T)^{2n-2}. \tag{2.4}$$

---

<sup>†</sup>Here, the fact that  $T$  is monic is essential. Otherwise,  $\mathbb{Z}[\theta]$  would not be a suborder of  $\mathcal{O}_{\mathbb{K}}$ .

PROPOSITION 2.1. *Let  $T$  be a monic irreducible polynomial of degree  $n \geq 2$ . Then  $T$  defines a number field  $\mathbb{K}$  whose discriminant  $\Delta_{\mathbb{K}}$  satisfies*

$$|\Delta_{\mathbb{K}}| \leq |\Delta(T)| \leq n^{2n} H(T)^{2n-2}.$$

*Proof.* It is an almost direct consequence of combining (2.3) and (2.4) which yields the improved bound for the discriminant. It remains to check the simple fact that:

$$\forall n \in \mathbb{N}^*, \quad (n+1)^{n-1} \leq n^n.$$

□

### 3. Motivations and link with class group computation

Our main motivation for reducing defining polynomials of number fields is to speed up the computations performed to determine class groups. Indeed, in [2] Biasse and Fieker propose a complexity improvement for class group computation when the number field is defined by a polynomial with small enough coefficients. More precisely, given parameters  $n_0, d_0 > 0$  and  $0 < \alpha < \frac{1}{2}$ , they introduced the classes<sup>†</sup>  $\mathcal{C}_{n_0, d_0, \alpha}$  of number fields  $\mathbb{K} = \mathbb{Q}[\theta] = \mathbb{Q}[X]/(T)$  such that the defining polynomial  $T \in \mathbb{Z}[X]$  of  $\mathbb{K}$  satisfies

$$n = \deg(T) = n_0(\log |\Delta_{\mathbb{K}}|)^{\alpha}(1 + o(1)) \quad \text{and} \quad d = \log H(T) = d_0(\log |\Delta_{\mathbb{K}}|)^{1-\alpha}(1 + o(1)), \quad (3.1)$$

where  $\Delta_{\mathbb{K}}$  is the discriminant of  $\mathbb{K}$  – equivalently, the discriminant of the order  $\mathcal{O}_{\mathbb{K}}$ . For these classes, they state the following theorem:

THEOREM 3.1 ([2, Theorem 6.2]). *Under GRH and smoothness heuristics about ideals, if the number field  $\mathbb{K}$  is in  $\mathcal{C}_{n_0, d_0, \alpha}$ , there exists an  $L_{|\Delta_{\mathbb{K}}|}(a, c)$  algorithm for class group and unit group computation for some  $c > 0$  and  $a$  such that  $1 - a \geq \alpha \geq 1 - 2a$ ,  $a \geq \frac{1}{3}$  and  $a \geq \alpha$ .*

When a number field  $\mathbb{K}$  is in  $\mathcal{C}_{n_0, d_0, \alpha}$ , it is defined by a polynomial  $T$  verifying (3.1) so that:

$$\log H(T) = \frac{d_0 n_0 (1 + o(1))}{n} \log |\Delta_{\mathbb{K}}| \quad \text{i.e.} \quad H(T) = |\Delta_{\mathbb{K}}|^{\frac{\kappa}{n}} \quad (3.2)$$

where  $\kappa = n_0 d_0 (1 + o(1))$  tends to a constant.

We know from Proposition 2.1 that the height of  $T$  satisfies  $H(T) \geq n^{\frac{n}{n-1}} |\Delta_{\mathbb{K}}|^{\frac{1}{2(n-1)}}$ . This means that the best – i.e. the smallest – we can hope for is  $\kappa = \frac{1}{2} + o(1)$ . Furthermore, for randomly chosen polynomial we expect to be close to this best case.

However, as far as we know, the best theoretical bound on the minimal height of smallest defining polynomials of an arbitrary number field is far from being that good. Indeed, it is known that:

THEOREM 3.2. *Let  $\mathbb{K}$  be a number field of degree  $n$ . There exists  $\theta \in \mathcal{O}_{\mathbb{K}} \setminus \mathbb{Z}$  whose minimal polynomial  $P_{\theta}$  satisfies*

$$H(P_{\theta}) \leq 3^n \left( \frac{|\Delta_{\mathbb{K}}|}{n} \right)^{\frac{n}{2n-2}}.$$

---

<sup>†</sup>The results of [2] are slightly more general than what we state, since they consider the computation of class groups for all orders in  $\mathbb{K}$ . For simplicity, we only look at the maximal order  $\mathcal{O}_{\mathbb{K}}$ .

*Proof.* The proof is provided in Appendix A. □

In particular, this result allows us to conclude in the case of primitive number fields, i.e. number fields that do not contain non-trivial subfields. In this case, every element of  $\mathcal{O}_{\mathbb{K}} \setminus \mathbb{Z}$  corresponds to a generator of  $\mathbb{K}$  whose minimal polynomial defines  $\mathbb{K}$ .

**COROLLARY 3.3.** *In primitive number fields, there exists a defining polynomial  $T$  whose height satisfies*

$$H(T) \leq 3^n \left( \frac{|\Delta_{\mathbb{K}}|}{n} \right)^{\frac{n}{2n-2}}.$$

Note that this bound is much worse than we would like for our application. Indeed the bound on the height is only about the square root of the discriminant while we would like something of the order of the  $n$ -th root of the discriminant. Alternatively, we see that it corresponds to a value  $\kappa \approx \frac{n}{2}$ , much larger than the best case  $\kappa \approx \frac{1}{2}$ .

For our purpose, it is difficult to directly work with the classes  $\mathcal{C}_{n_0, d_0, \alpha}$ . Instead, we introduce a more convenient and more general variation. More precisely, we first introduce slightly modified classes  $\mathcal{C}'_{n_0, d_0, \alpha}$  satisfying:

$$\begin{aligned} n &= n_0 \left( \frac{\log |\Delta_{\mathbb{K}}|}{\log \log |\Delta_{\mathbb{K}}|} \right)^{\alpha} (1 + o(1)) \quad \text{and} \\ d &= \log H(T) = d_0 (\log |\Delta_{\mathbb{K}}|)^{1-\alpha} (\log \log |\Delta_{\mathbb{K}}|)^{\alpha} (1 + o(1)). \end{aligned}$$

Introducing powers of  $\log \log |\Delta_{\mathbb{K}}|$  in this way is more consistent with what is usually done for discrete logarithms computations using index calculus and simplifies the asymptotic analysis of index calculus algorithms. All the theorems of [2] can be readily adapted to account for this change.

This modification being done, we now generalize the definition to include more number fields. Let  $n_0 > 1$  be a real parameter arbitrarily close to 1,  $d_0 > 0$ ,  $\alpha \in [0, 1]$  and  $\gamma \in [0, 1]$  such that  $\alpha + \gamma \geq 1$ , we define  $\mathcal{D}_{n_0, d_0, \alpha, \gamma}$  as the set of all number fields  $\mathbb{K}$  of discriminant  $\Delta_{\mathbb{K}}$  that admit a monic defining polynomial  $T \in \mathbb{Z}[X]$  of degree  $n$  that satisfies:

$$\begin{aligned} \frac{1}{n_0} \left( \frac{\log |\Delta_{\mathbb{K}}|}{\log \log |\Delta_{\mathbb{K}}|} \right)^{\alpha} \leq n \leq n_0 \left( \frac{\log |\Delta_{\mathbb{K}}|}{\log \log |\Delta_{\mathbb{K}}|} \right)^{\alpha} \quad \text{and} \quad (3.3) \\ d = \log H(T) \leq d_0 (\log |\Delta_{\mathbb{K}}|)^{\gamma} (\log \log |\Delta_{\mathbb{K}}|)^{1-\gamma}. \end{aligned}$$

In addition to the modification between  $\mathcal{C}$  and  $\mathcal{C}'$ , there are two essential differences between our classes  $\mathcal{D}$  and the previous classes. First, we have two different exponents  $\alpha$  and  $\gamma$  for  $\log |\Delta_{\mathbb{K}}|$  in  $n$  and  $d$ , rather than using  $\gamma = 1 - \alpha$ . Second, instead of having a  $(1 + o(1))$  we introduce inequalities in order to be more precise. We also have the useful property that  $\mathcal{D}_{n_0, d_0, \alpha, \gamma} \subset \mathcal{D}_{n'_0, d'_0, \alpha, \gamma'}$  whenever  $n_0 \leq n'_0$ ,  $d_0 \leq d'_0$  and  $\gamma \leq \gamma'$ .

Thanks to Corollary 3.3, we need only consider classes  $\mathcal{D}_{n_0, d_0, \alpha, \gamma}$ , with  $\gamma$  at most 1. Moreover, the defining polynomials that appear in [2] essentially correspond to the lower end of the classes  $\mathcal{D}$  where  $\alpha + \gamma = 1$ . Indeed, considering the modified classes  $\mathcal{C}'$ , we see that every  $\mathbb{K}$  in  $\mathcal{C}'_{n_0, d_0, \alpha}$  also belongs to  $\mathcal{D}_{n_0, d_0, \alpha, 1-\alpha}$ . Thus, our broader definition of the classes include more number fields, not only those whose minimal defining polynomials satisfy  $\alpha + \gamma = 1$ .

Note that, in the context of class group computations, algorithms with complexity higher than  $L_{|\Delta_{\mathbb{K}}|}(\frac{1}{2})$  are not of great interest. As a consequence, since the algorithm we propose is exponential in  $n$ , we can restrict ourselves to considering classes with  $\alpha \leq \frac{1}{2} \leq \gamma$ .

#### 4. The algorithm of Cohen and Diaz y Diaz for polynomial reduction

To the best of our knowledge, the only previous algorithm for reducing defining polynomials is a folklore algorithm, mentioned in particular by Cohen and Diaz y Diaz in [5] and in [14, Chapter V]. In the sequel, we call it the algorithm of Cohen and Diaz y Diaz.

They define the *size* of a monic degree- $n$  polynomial  $T = \prod(X - \tau_j)$  as

$$S(T) = \sum_{j=1}^n |\tau_j|^2$$

and it is the quantity they choose to minimize. Since for all  $k \in \{0, \dots, n\}$ ,

$$|t_{n-k}| \leq \binom{n}{k} \left( \frac{S(T)}{n} \right)^{\frac{k}{2}},$$

the size of  $T$  is related to the size of  $\max(|t_{n-k}|^{2/k})$  and hence to the height of the polynomial. Yet this relation is both more complicated and weaker than the inequalities given by (2.2) and (2.3).

The main reason for their choice is that the size is not too hard to minimize. Indeed, computing a polynomial with the smallest size is equivalent to finding  $T$  such that the vector formed of its roots has the smallest  $L_2$ -norm. Remembering that this vector belongs to the lattice formed of the embeddings of  $\mathcal{O}_{\mathbb{K}}$  in  $\mathbb{C}^n$ , this can be done using a lattice reduction algorithm. In particular, the LLL algorithm is a good option since it finds a basis of short vectors. However, while it solves the problem for small values of  $n$ , when the dimension grows, it is not guaranteed to find the shortest vector. If needed, it is possible to use stronger lattice reduction algorithms such as BKZ whose complexity is exponential in  $n$  but remains polynomial in the size of the entries.

Thus, the first step of their method is to compute an integral basis  $\omega_1, \dots, \omega_n$  of  $\mathcal{O}_{\mathbb{K}}$ , so that every integer  $\theta \in \mathcal{O}_{\mathbb{K}}$  can be expressed as

$$\theta = \sum_{i=1}^n x_i \omega_i,$$

where the  $x_i$  are in  $\mathbb{Z}$ . Assuming that  $\theta$  generates  $\mathbb{K}$ , then its minimal polynomial coincides with its characteristic polynomial

$$T_{\theta} = \prod_{j=1}^n \left( X - \sum_{i=1}^n x_i \sigma_j(\omega_i) \right).$$

This turns the correspondence between algebraic integers of degree  $n$  and monic polynomials defining the field  $\mathbb{K}$  into a correspondence between polynomials and vectors in the lattice of embeddings of  $\mathcal{O}_{\mathbb{K}}$  in  $\mathbb{C}^n$ . This lattice is generated by the  $n$  vectors:

$$\Omega_i = [\sigma_1(\omega_i), \dots, \sigma_n(\omega_i)]. \quad (4.1)$$

When the field  $\mathbb{K}$  is imprimitive, care should be taken to avoid polynomials that generate a strict subfield<sup>†</sup> of  $\mathbb{K}$ . For an integer  $\theta$ , the size of its polynomial  $T_{\theta}$  is given by a positive definite quadratic form in the  $x_i$ :

$$S(T_{\theta}) = \sum_{j=1}^n \left| \sum_{i=1}^n x_i \sigma_j(\omega_i) \right|^2 = \sum_{i,j} \left( \sum_{k=1}^n \sigma_k(\omega_i) \overline{\sigma_k(\omega_j)} \right) x_i x_j.$$

---

<sup>†</sup>This issue is taken care of in practice in PARI/GP by restricting the enumeration in order to avoid elements that only generate subfields. In theory, since they can be an extremely large number of short bad elements, bounding the resulting complexity is difficult.

Equivalently, one can remark that the coefficients  $\sum_{k=1}^n \sigma_k(\omega_i) \overline{\sigma_k(\omega_j)}$  are the entries of the Gram matrix of the above lattice. Since lattice reduction, including the LLL algorithm, only requires this Gram matrix as input, one can find a basis of short vectors in  $\mathbb{Z}^n$  corresponding to small values of  $S(T_\theta)$ . Finally, the algorithm of Cohen and Diaz y Diaz selects the best value of  $\theta$  it can find, breaking ties in any arbitrary manner when several values are equally good.

From an implementation perspective, the lattice generated by vectors  $\Omega_i$  (and/or the Gram matrix of the lattice) cannot be represented exactly. Instead, it is explained in [6] that, in general, it should be replaced by an approximation with sufficiently high precision. In fact, there is an interesting special case that occurs when  $\mathbb{K}$  is totally real. In that case, the Gram matrix is integral and the lattice reduction can thus be performed on the exact lattice, represented by its Gram matrix. For the general case, an analysis of the necessary precision is given in [1].

To summarize, the existing algorithm reduces the above lattice in order to find primitive integer of  $\mathbb{K}$  with smallest size. However, as already noticed by Cohen in [6, Remark Algorithm 4.4.12], having the smallest size does not necessarily imply having the smallest height.

EXAMPLE 1. To illustrate this fact, here are a few examples of polynomials where the application of the algorithm of Cohen and Diaz y Diaz does not return a smallest-height polynomial:

Minimal-height polynomials	Cohen/Diaz y Diaz output polynomials
$x^3 + 6381x^2 + 4378x - 1216$	$x^3 - x^2 - 3537064x + 2193757452$
$x^3 - 9681x^2 - 5434x - 6901$	$x^3 - 31246021x - 67226458585$
$x^3 - 6665x^2 - 4318x - 2977$	$x^3 + 336681x - 419200237$
$x^3 - 6018x^2 - 1387x + 6161$	$x^3 - 12073495x - 16147208593$

Our implementation (see Section 7) certifies that the four polynomials appearing in the table have minimal heights.

### 5. An optimal algorithm for NF defining polynomial reduction

We know from Section 3 that existing bounds do not guarantee, for an arbitrary number field, the existence of a defining polynomial with coefficients small enough to allow application of Theorem 3.1. In the present section, we thus aim at the next best thing: given an arbitrary defining polynomial for a number field, try to find a monic polynomial of smallest height that defines the same number field.

Throughout this section, we assume that we are given as input a monic<sup>†</sup> irreducible polynomial  $T_{\text{in}}$  and a factorization of the discriminant  $\Delta(T_{\text{in}})$  into prime factors. This allows us to precompute the discriminant  $\Delta_{\mathbb{K}}$  of the number field  $\mathbb{K}$  defined by  $T_{\text{in}}$  and an integral basis  $\omega_1, \dots, \omega_n$  of  $\mathcal{O}_{\mathbb{K}}$  using for instance Round 2 algorithm [6, Algorithm 6.1.8].

With the motivations of Section 3 in mind, we now wish to find a monic polynomial of smallest height that defines  $\mathbb{K}$ . Since such a polynomial clearly exists, let  $T_F$  denote a minimal polynomial<sup>‡</sup>. Let  $\theta_F$  be a root of  $T_F$ ; since  $T_F$  is monic,  $\theta_F$  belongs to  $\mathcal{O}_{\mathbb{K}}$ . Thus, we can write  $T_F(X) = \prod (X - \sigma_j(\theta_F))$ . As in the algorithm of Cohen and Diaz y Diaz, we remark that the

<sup>†</sup>If the number field is initially described by a non-monic polynomial, it is possible using a linear change of variables to make it monic. The transformation can increase the height by a large factor, but it is mostly irrelevant for our purpose.

<sup>‡</sup>Uniqueness of  $T_F$  is not guaranteed. Indeed,  $T_F(-X)$  also defines  $\mathbb{K}$  and there might be others. However, any of these polynomials achieves our goal and more importantly, the minimal height  $H(T_F)$  is well-defined.



vector corresponding to  $\theta_F$  in the lattice associated to  $\mathcal{O}_{\mathbb{K}}$  should be small in a certain sense, since the coefficients of  $T_F$ , i.e. the symmetric polynomials in the coordinates of the vector are small. However, we need to replace the notion of smallness by something more adapted than the  $L_2$ -norm of the vector.

This is precisely the idea behind our algorithm, consider all small enough vectors in the lattice until we find one that defines a minimal polynomial and can prove that this polynomial is indeed of minimal height. In order to do that, let first focus on the target solution  $T_F$  and the corresponding vector  $v(\theta_F) = [\sigma_1(\theta_F), \dots, \sigma_n(\theta_F)]$ . Remark that if all entries of  $T_F$  have roughly the same size, then its  $L_2$ -norm is also small. In that case, the algorithm of Cohen and Diaz y Diaz succeeds in finding  $T_F$ . However, when the sizes of the entries are unbalanced, success is no longer guaranteed. For a more precise analysis, let us introduce a parameter  $c > 1$  whose value is determined later to minimize algorithmic complexity<sup>†</sup>. Now, consider the vector  $[\log_c |\sigma_1(\theta_F)|, \dots, \log_c |\sigma_n(\theta_F)|]$  that describes the relative sizes of the coordinates of  $v(\theta_F)$ . Then, round each value up to the smallest possible non negative integer, this gives a vector  $b^F = [b_1^F, \dots, b_n^F]$ . Since  $T_F$  is monic, by definition, the Mahler measure  $M(T_F)$  is the product of the values  $\max(1, |\sigma_i(\theta_F)|)$  and we can write the following inequality between  $M(T_F)$  and the vector  $b^F$ :

$$c^{\sum b_j^F - n} \leq M(T_F) \leq c^{\sum b_j^F}.$$

In particular, this guarantees that:

$$\sum_{j=1}^n b_j^F \leq \log_c M(T_F) + n \leq \log_c H(T_F) + \frac{1}{2} \log_c(n+1) + n, \quad (5.1)$$

thanks to Equation (2.3).

Given the vector of weights  $b^F$ , we can introduce a *weighted* copy of the lattice corresponding to  $\mathcal{O}_{\mathbb{K}}$  in  $\mathbb{C}^n$ . The weighted lattice is generated by the vectors:

$$\widetilde{\Omega}_i = \left[ \frac{\sigma_1(\omega_i)}{c^{b_1^F}}, \dots, \frac{\sigma_n(\omega_i)}{c^{b_n^F}} \right].$$

In this new lattice,  $\theta_F$  is associated to the vector:

$$\tilde{v}(\theta_F) = \left[ \frac{\sigma_1(\theta_F)}{c^{b_1^F}}, \dots, \frac{\sigma_n(\theta_F)}{c^{b_n^F}} \right].$$

Note that each coordinate of  $\tilde{v}(\theta_F)$  has norm at most 1 so that its  $L_2$ -norm is bounded by  $\sqrt{n}$ .

As in Section 4, we could work with the Gram matrix of the lattice. However, as an alternative, we prefer to replace the lattice in  $\mathbb{C}^n$  by a related lattice<sup>‡</sup> in  $\mathbb{R}^n$ . We assume that the complex embeddings are ordered with  $r_1$  real embeddings first and  $r_2$  pairs of conjugate complex embeddings after that. More precisely,  $\sigma_i$  is a real embedding for  $1 \leq i \leq r_1$  and the complex embeddings are related by  $\sigma_{r_1+i} = \overline{\sigma_{r_1+i+r_2}}$  for  $1 \leq i \leq r_2$ . Using a classical idea of replacing pairs of complex numbers by their real and imaginary parts, we obtain a lattice of  $\mathbb{R}^n$  generated by  $n$  real vectors:

$$\Omega_i^{\mathbb{R}} = [\sigma_1(\omega_i), \dots, \sigma_{r_1}(\omega_i), \sqrt{2} \operatorname{Re}(\sigma_{r_1+1}(\omega_i)), \sqrt{2} \operatorname{Im}(\sigma_{r_1+1}(\omega_i)), \dots, \sqrt{2} \operatorname{Im}(\sigma_{r_1+r_2}(\omega_i))].$$

We also remark that, thanks to the factor  $\sqrt{2}$ , the  $L_2$ -norm of each real vector is identical to the  $L_2$ -norm of the corresponding complex vector. Indeed, the real and imaginary parts are common to the two conjugates embeddings and it is necessary to count their squares twice in the  $L_2$ -norm.

---

<sup>†</sup>It is done in Appendix C and it turns out that the optimal choice is  $c = \exp(1)$ .

<sup>‡</sup>A third option would be to work directly with the complex lattice as explained in [7].

We can also do the same thing for the weighted lattice. First, remark that since  $\sigma_{r_1+i} = \overline{\sigma_{r_1+i+r_2}}$ , we necessarily have  $b_{r_1+i}^F = b_{r_1+i+r_2}^F$ . As a consequence, only the first  $r = r_1 + r_2$  coefficients of the weight vector  $b$  are needed to describe the weighted lattice, which is generated by:

$$\widetilde{\Omega}_i^{\mathbb{R}} = \left[ \frac{\sigma_1(\omega_i)}{c^{b_1^F}}, \dots, \frac{\sigma_{r_1}(\omega_i)}{c^{b_{r_1}^F}}, \frac{\sqrt{2} \operatorname{Re}(\sigma_{r_1+1}(\omega_i))}{c^{b_{r_1+1}^F}}, \frac{\sqrt{2} \operatorname{Im}(\sigma_{r_1+1}(\omega_i))}{c^{b_{r_1+1}^F}}, \dots, \frac{\sqrt{2} \operatorname{Im}(\sigma_{r_1+r_2}(\omega_i))}{c^{b_{r_1+r_2}^F}} \right].$$

In  $\widetilde{\Omega}_i$ , the vector corresponding to the integer  $\theta_F$  has all its entries of individual norm at most 1. Thus its  $L_2$ -norm is bounded by  $\sqrt{\widetilde{n}}$ . Due to the equality of the  $L_2$ -norm, this bound also holds in the lattice generated by the  $\widetilde{\Omega}_i^{\mathbb{R}}$ .

As a consequence, our algorithm simply aims at enumerating all the possible weighted lattices and all vectors of  $L_2$ -norm at most  $\sqrt{n}$  in these lattices. However, doing that directly would be sub-optimal, because for any integer  $\theta$  in  $\mathcal{O}_{\mathbb{K}}$ , we would also consider many shifted copy  $\theta + j$  with  $j \in \mathbb{Z}$ . To avoid that, we work instead with a lattice projected orthogonally with respect to the vector  $\widetilde{\Omega}_1^{\mathbb{R}}$  to find a candidate  $\theta$ . Then, finding the best  $\theta + j$  has a polynomial runtime in  $n$ . The resulting algorithm can be described as follows:

**ALGORITHM 1.** *Search for a minimal-height defining polynomial.*

**Input:** Integral basis  $\omega_1, \dots, \omega_n$  of  $\mathcal{O}_{\mathbb{K}}$  and a priori bound  $B_F$  on  $\log_c H(T_F)$ .

- (i) Fix  $k = 0$ , let  $\text{Bound} = B_F + \frac{1}{2} \log_c(n+1) + n$
- (ii) For all  $(b_1, \dots, b_n) \in \mathbb{N}^n$  such that  $b_{r_1+i} = b_{r_1+r_2+i}$  and  $\sum_{j=1}^n b_j = k$ :
  - (a) Construct the weighted lattice generated by the vectors:

$$\widetilde{\Omega}_i^{\mathbb{R}} = \left[ \frac{\sigma_1(\omega_i)}{c^{b_1}}, \dots, \frac{\sigma_{r_1}(\omega_i)}{c^{b_{r_1}}}, \frac{\sqrt{2} \operatorname{Re}(\sigma_{r_1+1}(\omega_i))}{c^{b_{r_1+1}}}, \dots, \frac{\sqrt{2} \operatorname{Im}(\sigma_{r_1+r_2}(\omega_i))}{c^{b_{r_1+r_2}}} \right].$$

Since the lattice has real entries, algorithmically we replace it by a sufficiently precise fixed point approximation and scale it up to an integer lattice.

- (b) Project the vectors  $\widetilde{\Omega}_i^{\mathbb{R}}$  for  $i \in \{2, \dots, n\}$  orthogonally with respect to  $\widetilde{\Omega}_1^{\mathbb{R}}$ .
- (c) Enumerate all vectors of  $L_2$ -norm at most<sup>†</sup>  $1.1\sqrt{n}$  in this projected lattice and for each short vector  $v^\perp$ :
  - (1) Lift  $v^\perp$  to the shortest possible vector  $v$  in the full lattice. (This can be done efficiently by using Gauss' algorithm on the two-dimensional lattice spanned by  $\widetilde{\Omega}_1^{\mathbb{R}}$  and an arbitrary lift of  $v^\perp$ .)
  - (2) Reconstruct the corresponding polynomial

$$T_v(X) = \prod_{j=1}^{r_1} (X - v_j) \cdot \prod_{j=1}^{r_2} ((X - v_{r_1+2j-1} - i v_{r_1+2j}) \cdot (X - v_{r_1+2j-1} + i v_{r_1+2j})).$$

- (3) If  $T_v$  is irreducible:
  - Find the integer  $j$  that minimizes the height of  $T_v(X + j)$
  - Store  $T_v(X + j)$  if it has the smallest height encountered until this point.
  - When memorizing  $T_v(X + j)$ ,  
update  $\text{Bound}$  to  $\lfloor \log_c H(T_v(X + j)) \rfloor + \frac{1}{2} \log_c(n+1) \rfloor + n$
- (iii) Increment  $k$ . If  $k \leq \text{Bound}$ , goto (ii) else **stop**.

**Output:** The stored polynomial, whose height is minimal among all the defining polynomials.

---

<sup>†</sup>With an exact representation of the lattice, it would suffice to enumerate vectors of norm at most  $\sqrt{n}$ . The 1.1 factor comes from the fact that we are dealing with an approximation of the lattice. See the paragraph on precision.

If the *a priori* bound  $B_F$  is correct, we know that  $T_F$  belongs to one of the weighted lattice we have encountered. Thus, the last memorized polynomial is – one of the possible –  $T_F$ .

Otherwise, if the input bound does not hold, the algorithm asserts this fact.

*Finding  $j$ .* In the algorithm, we need to find the integer  $j$  that minimizes the height of  $T_v(X + j)$ . This can be done in polynomial time. Indeed, each coefficient of  $T_v(X + j)$  (viewed as a polynomial in  $X$ ) is a polynomial in  $j$ . We thus want to minimize (over the integers) the maximum of the absolute values of  $n$  polynomials<sup>†</sup>. This can be restated as minimizing the maximum of  $2n$  polynomials: the coefficients and their opposites. This function is a continuous positive piecewise-polynomial real function. The transition from one piece to next necessarily occurs at a root of a difference of two of the  $2n$  polynomials. Thus, there are at most  $O(n^3)$  pieces. In each piece, it is easy to minimize over the reals and then over the integers by considering the two integers<sup>‡</sup> closest to this real minimum. As a consequence, the best value for  $j$  can be found in polynomial time.

*Precision.* At the beginning of the algorithm, the lattice generated by the vectors  $\widetilde{\Omega}^{\mathbb{R}}$  is replaced by a “sufficiently” precise fixed point approximation and then scaled up to an integer lattice. In order to make the description of the algorithm complete, we need to specify the corresponding precision. The key point is that the precision should be high enough to ensure that any short vector of the exact lattice corresponds to a short vector of the approximate lattice. Indeed, we are essentially enumerating all vectors of length bounded by  $\sqrt{n}$  and we want to make sure that none of those can be overlooked. Let  $v$  be a vector of norm at most  $\sqrt{n}$  of the exact lattice, we can write:

$$v = \sum_{i=1}^n \nu_i \widetilde{\Omega}_i^{\mathbb{R}} \quad \text{with integer coefficients } \nu_i.$$

In order to make sure that the vector of the approximate lattice is also short, i.e. shorter than  $1.1\sqrt{n}$ , the precision should be good enough for the sum of products of the coefficients  $\nu_i$  by the approximation error to be smaller than a constant. In particular, it is enough to require that the maximum of the  $\nu_i$  times the approximation error is smaller than  $\frac{1}{10n}$ .

As a consequence, in order to bound the necessary precision, it suffices to upper bound the coefficients  $\nu_i$ . A classical technique to obtain such a bound is to start from a lower bound on the orthogonalized vectors of the initial basis and remark that the coefficients  $\nu_i$  are upper bounded in absolute value by  $\sqrt{n}$  times the inverse of this bound. To obtain a lower bound on a given  $\|b_i^*\|$ , we can divide the determinant of the lattice by the norms of all the vectors but the corresponding  $b_i$ . The determinant of the scaled lattice of iteration  $k$  of the algorithm is  $\sqrt{\Delta_{\mathbb{K}}} c^{-k}$ . Thanks to Lemma 1.1 of [14, Chapter V], we know that in the basis of the ring of integers of  $\mathbb{K}$  computed from the input polynomial  $T$ ,  $\omega_i$  is a polynomial of degree  $i - 1$  in the roots of  $T$ . Moreover, the coefficients of this polynomial are in the interval  $[-1, 1]$ . As a consequence, the norm of  $b_i$  – the lattice element corresponding to  $\omega_i$  – can be upper bounded by  $i\sqrt{n}Z$  where  $Z$  denotes the largest complex modulus among the roots of  $T$ . In particular,  $Z \leq M(T)$ .

Putting all this together, we can upper bound the coefficients  $\nu_i$  at iteration  $k$  by:

$$\frac{c^k M(T)^{n^2} n^{1.5n}}{\Delta_{\mathbb{K}}}.$$

---

<sup>†</sup>Since  $T_v(X + j)$  is monic, we do not need to include the coefficient in front of  $X^n$  in the minimization.

<sup>‡</sup>Ignoring the integer values lying outside of the current piece.

Thus, it suffices to work with a precision  $k \log_2(c) + n^2 \log_2 M(T) + 1.5n \log_2(n) + \log_2(10n)$ , which is polynomial in the size of the input polynomial. This precision is also sufficient for performing the reconstruction of  $T_v$  from the complex embeddings, rounding each coefficient to the nearest integer (see [1, Section 3.2] for more details).

*Pseudo-canonical polynomial.* For some applications, such as the construction of tables of number fields, it is useful to have a canonical polynomial that represents a given number field. In that case, it is not difficult to add in the above algorithm an arbitrary criteria such as lexicographic order on the coefficients to single out one of the possible minimal-height polynomials. Indeed, all of them are encountered during the enumeration.

### 6. Complexity analysis

Now we have described our algorithm, in order to understand its algorithmic complexity, we need to count the involved lattices and to study the cost of short vector enumeration in each of those lattices. We restrict ourselves to primitive number fields for this analysis. For imprimitive fields, the algorithm still works but its complexity may be much higher due the possibility of encountering an extremely large number of integral elements that generate subfields when doing the short vector enumerations. We leave as an open problem the adaptation of the techniques used in PARI/GP for the algorithm of Cohen and Diaz y Diaz with imprimitive fields to our method. In the complexity analysis, we ignore<sup>†</sup> the cost of computing an integral basis for the maximal order of the field  $\mathbb{K}$ . Recall that according to [6, Section 6.1], the complexity of this step is dominated by the factorization of the discriminant of the input polynomial. Thanks to the NFS factoring algorithm, this can be done in heuristic time  $L_{\Delta(T_{\text{in}})}\left(\frac{1}{3}, \sqrt[3]{\frac{64}{9}}\right)$  and depending on  $T_{\text{in}}$ , it may be the most costly part of the complete computation.

#### 6.1. Number of lattices

When the input bound  $B_F$  is incorrect, the number of lattices that are explored is a function on  $B_F$ . When  $B_F$  is correct, thanks to the updating of the bound when a new polynomial is found, the number of lattices depends on the highest value of  $k$  that is reached and this value is at most  $\log_c H(T_F) + \frac{1}{2} \log_c(n+1) + n$  according to Equation (5.1). Thus, the exact running time depends on the quality of the output: it is faster to find polynomials with a smaller height.

Let us consider each iteration of the outer loop: for the iteration numbered  $k$ , the number of lattices is simply the number of vectors  $(b_1, \dots, b_n) \in \mathbb{N}^n$  that satisfy the two constraints  $b_{r_1+i} = b_{r_1+r_2+i}$  and  $\sum_{j=1}^n b_j = k$ . To obtain an upper bound, we can forget the first constraint<sup>‡</sup> and just count the number of vectors such that  $\sum_{j=1}^n b_j = k$ . It is a classical combinatorial result that this number is  $\binom{k+n-1}{n-1}$ . Hence, the total number of lattices considered by the algorithm is:

$$\sum_{k=0}^{k_F} \binom{k+n-1}{n-1} = \binom{k_F+n}{n} \leq \frac{(k_F+n)^n}{n!}, \tag{6.1}$$

where  $k_F$  is the value of  $k$  in the last iteration, i.e.

$$k_F = \lfloor \min(B_F, \log_c H(T_F)) + \frac{1}{2} \log_c(n+1) \rfloor + n.$$

---

<sup>†</sup>The reason for ignoring this step is that it is already required for the class group computation itself.

<sup>‡</sup>Note that, for totally real fields the first constraint is always trivially satisfied. Thus, the bound is tight.

Bound on  $k_F$  and classes  $\mathcal{D}_{n_0, d_0, \alpha, \gamma}$ . Given as input a number field known to be in a class  $\mathcal{D}_{n_0, d_0, \alpha, \gamma}$ , we can simply set the value of  $B_F$  given to the algorithm to be:

$$B_F = \left\lfloor \frac{d_0}{\log c} (\log |\Delta_{\mathbb{K}}|)^\gamma (\log \log |\Delta_{\mathbb{K}}|)^{1-\gamma} \right\rfloor.$$

Moreover, since we are limiting ourselves to the cases  $\alpha \leq \frac{1}{2} \leq \gamma$ ,  $n$  (and  $\log_c(n+1)$ ) is negligible compared to  $B_F$  and thus  $k_F + n = (1 + o(1))B_F$ .

We conclude that the number of lattices in the enumeration is upper bounded by  $\frac{((1+o(1))B_F)^n}{n!}$  and thus by  $L_{|\Delta_{\mathbb{K}}|}(\alpha, \gamma n_0 - \frac{\alpha}{n_0})$ .

## 6.2. Cost of each enumeration

The second analysis treats of the enumeration phase for each weighted lattice. We recall that we are interested in all vectors whose euclidean norm is below  $\sqrt{n}$ . A slight modification from the Kannan's SVP algorithm brings us the enumeration process and all its analysis is detailed in [9]. Note that this cost of enumeration includes the HKZ reduction of the basis.

**PROPOSITION 6.1.** *Enumerating all vectors of norm below  $\sqrt{n}$  in one of our weighted lattice can be done in at most  $\text{Poly}(\log |\Delta_{\mathbb{K}}|) \cdot n^{\frac{n}{2e} + o(n)}$  binary operations.*

*Proof.* The proof is a consequence of [9] and is provided in Appendix B. Note that the  $n^{o(n)}$  in particular hides a  $c^n$  term which is considered in Appendix C when optimizing  $c$ .  $\square$

We remark that  $n^{\frac{n}{2e} + o(n)} = L_{|\Delta_{\mathbb{K}}|}(\alpha, \frac{\alpha n_0}{2e})$ . Multiplying by the number of lattices, we find that the final complexity can be expressed as

$$L_{|\Delta_{\mathbb{K}}|} \left( \alpha, \gamma n_0 - \frac{\alpha}{n_0} + \frac{\alpha n_0}{2e} \right),$$

which becomes arbitrarily close to  $L_{|\Delta_{\mathbb{K}}|}(\alpha, \gamma - \frac{2e-1}{2e} \alpha)$  when  $n_0$  is taken close to 1.

## 7. Application to class group computation

Let us get back to class group computation. We recall that Biassé and Fieker find a way to bring down the complexity of their algorithm for certain classes of  $\mathfrak{o}$ -orders in  $\mathfrak{o}$ -number fields. Given a number field  $\mathbb{K} \in \mathcal{C}_{n_0, d_0, \alpha}$  – defined by a polynomial  $T$  satisfying (3.1) – they are able to compute the class group and a compact representation of a fundamental system of units of this order in time  $L_{|\Delta_{\mathbb{K}}|}(a, c)$  for  $\frac{1}{3} \leq a < \frac{1}{2}$  depending on the parameters of the class and  $c > 0$ .

We now want to investigate the theoretical impact of using a minimal-height defining polynomial for class group computations. Note that the existence of such a polynomial ensures that  $\mathbb{K}$  lies in a certain class  $\mathcal{D}_{n_0, d_0, \alpha, \gamma}$  with  $\gamma \in [1 - \alpha, 1]$ . If  $\gamma$  reaches the lower bound  $\gamma = 1 - \alpha$ , then  $\mathbb{K} \in \mathcal{D}_{n_0, d_0, \alpha, 1 - \alpha}$  and we can apply the conditional improvement<sup>†</sup> of [2] to this field. Our idea is then to add our algorithm as a precalculation to theirs in order to find the parameters required to fit  $\mathbb{K}$  in such a class, if possible.

Doing this enables to extend the applicability of the algorithm of Biassé and Fieker to every number field in a class  $\mathcal{D}_{n_0, d_0, \alpha, 1 - \alpha}$ , even if the number field is not initially given by a small-height defining polynomial. It remains to check that the cost of this precalculation does not

---

<sup>†</sup>More precisely the adaptation of this improvement to the classes  $\mathcal{C}'$ .

outweigh the global complexity of the rest of the class group computation. We know from Section 6 that our algorithm runs in time  $L_{|\Delta_{\mathbb{K}}|}(\alpha, O(1))$  in that case. This never dominates the cost of their class group algorithm, thus we can state that:

**THEOREM 7.1** ([2, Theorem 6.2]). *Under GRH and smoothness heuristics about ideals, for every number field  $\mathbb{K} \in \mathcal{D}_{n_0, d_0, \alpha, 1-\alpha}$ , there exists an  $L_{\Delta_{\mathbb{K}}}(a, c)$  algorithm for class group and unit group computation for some  $c > 0$  and  $a$  satisfying  $a \geq \max(\alpha, \frac{1-\alpha}{2})$ .*

Furthermore, our new classes  $\mathcal{D}$  built from four parameters allow us to generalize the work of Biasse and Fieker to all number fields having  $\alpha \leq \frac{1}{2} \leq \gamma$ :

**THEOREM 7.2.** *Under GRH and smoothness heuristics about ideals, for every number field  $\mathbb{K} \in \mathcal{D}_{n_0, d_0, \alpha, \gamma}$ , there exists an  $L_{\Delta_{\mathbb{K}}}(a, c)$  algorithm for class group and unit group computation for some  $c > 0$  and  $a = \max(\alpha, \frac{\gamma}{2})$ .*

*Proof.* This is only a generalization of the work of Biasse and Fieker. Their final result mainly depends on [2, Theorem 3.1]. In fact we need to adapt the bounding of norms in the proof of this theorem to our more general classes. More precisely, we find that:

For all number fields  $\mathbb{K}$  in  $\mathcal{D}_{n_0, d_0, \alpha, \gamma}$ , we can find a  $B$ -smooth ideal equivalent to a  $|\Delta_{\mathbb{K}}|$ -ideal  $\mathfrak{a} \subseteq \mathcal{O}_{\mathbb{K}}$  with a decomposition in degree 1 prime ideals in time  $L_{\Delta_{\mathbb{K}}}(b, \mu)$  for some  $\mu \geq 0$  where  $B = L_{\Delta_{\mathbb{K}}}(a, \rho)$  for some  $\rho \geq 0$  provided that  $a$  and  $b$  satisfy:

$$(i) \ b \leq \gamma \leq a + b; \quad (ii) \ \alpha + \gamma \leq a + 2b; \quad (iii) \ \alpha + \gamma \leq 2a + b; \quad (iv) \ \alpha \leq b.$$

The proof is the same except that we modify the exponent appearing in the definition of  $k$ . The value  $1 - \beta - \frac{\tau}{2}$  in their proof becomes  $\alpha + \gamma - \beta - \frac{\tau}{2}$ .  $\square$

We thus conclude from this theoretical analysis that our algorithm for reducing number field defining polynomial widens the set of number fields whose class group and unit group can be computed with runtime  $L_{|\Delta_{\mathbb{K}}|}(a, O(1))$ ,  $\frac{1}{3} \leq a < \frac{1}{2}$ . The experimental results below illustrate that it also offers a good behavior on practical examples.

### 7.1. Experimental results

We implemented a prototype of our algorithm under Magma 2.21-6. The precision and the parameter  $c$  are fixed by the user. This slightly differs from the description of Section 5 since our code enumerate all the elements of norm below  $\sqrt{n}$  in each weighted lattice, instead of doing the minimization of the height of  $T_v(X + j)$ . Indeed, with the practical examples we considered, it is faster to proceed in that way. Furthermore, when a polynomial  $T$  (better than the input polynomial) is found, our code offers two options. The first is to stop immediately, since it appears that in practice the first polynomial to be found usually has minimal-height. Despite the fact that this early abort does not certify the minimality, it is a good practical option for class group computations. Another approach when a first polynomial is found is to directly increment  $k$  to the value  $\lfloor \log H(T) + \frac{1}{2} \log_c(n+1) \rfloor + n$ . This skips intermediate computations which are often useless and directly goes to checking that the polynomial  $T$  has minimal height.

In our implementation, we do not fix the parameter  $c$  to  $\exp(1)$  as in the asymptotic analysis. Instead, we remark that using a large  $c$  allows for smaller values of  $k$  and makes the code run

faster for finding a first candidate polynomial with small height. However, to find a minimal-height polynomial, smaller values of  $c$  are better. This can be seen on the examples given in Table 1.

EXAMPLE 2. As a first benchmark, we consider the polynomial  $t^{12} + 4t^{11} - 17t^{10} - 68t^9 + 108t^8 + 416t^7 - 314t^6 - 1129t^5 + 358t^4 + 1353t^3 - 36t^2 - 540t - 72$  given as an example in [14, Section V.3]. This polynomial is obtained by using the algorithm of Cohen and Diaz y Diaz to a polynomial with huge coefficients; it defines a suborder of index 670150656 of the ring of integers. More precisely, the vector corresponding to the above polynomial appears as the second vector in the reduced basis after LLL reduction. The fourth vector of the same basis yields a better polynomial with smaller height (505) and smaller index (439826112).

With our algorithm, we find an even better polynomial:

$$t^{12} - 14t^{11} + 25t^{10} + 62t^9 - 155t^8 - 50t^7 + 263t^6 - 50t^5 - 155t^4 + 62t^3 + 25t^2 - 14t + 1,$$

whose associated order has index 419904 – and whose height is 263. In addition, this polynomial is palindromic and, thus, reveals an underlying symmetry of the corresponding number field which was not apparent from the other defining polynomials.

EXAMPLE 3. To illustrate practically what changes in class group computation, we choose an example having smaller degree and larger coefficients:

$$x^5 - 5843635x^4 + 931633x^2 + 6577x - 8570.$$

Our implementation of the reduction algorithm certifies that this polynomial has minimal height.

If we give it to Magma V2.21-6 in order to compute the class group of the associated number field, it first “reduces” the polynomial as

$$x^5 - 2x^4 - 8001397580x^3 - 31542753393650x^2 + 3636653302451131875x + 4818547529425280067500$$

and then finds the class group – assuming GRH – in about 285 seconds, on the laptop we used for all experiments. More precisely, according to the output of its verbose mode, it seems that Magma derives the relations by sieving on different polynomials. Those polynomials appear to be chosen as the minimal polynomials of algebraic integers of the form  $\frac{\theta+b}{c}$ , where  $\theta$  is the second element of the LLL-reduced integral basis of  $\mathcal{O}_{\mathbb{K}}$ . We reconstructed the “reduced” polynomial from this information. Magma uses 2306 different sieving polynomials: 663 lead to 0 relations, 748 to 1 relation, 498 to 2 relations and 397 to at least 3 relations. None of them produce more than 8 relations.

In Magma V2.19-10, class group computation is not as optimized as in V2.21, but there is no reducing algorithm: Magma directly works with the input polynomial. Thus, we can compare the efficiency of using either of the two polynomials:

- with  $x^5 - 5843635x^4 + 931633x^2 + 6577x - 8570$ , it takes about 140 seconds,
- with the “reduced” one, it takes about 3240 seconds.

We see that using the old version with a minimal-height polynomial even makes faster than the new version, despite the huge optimization of the class group computation code. For a more detailed comparison, using the best polynomial as input, the old version only sieves on 58 different sieving polynomials and the first already leads to 784 relations.

Note that our implementation in Magma finds this minimal-height polynomial from the “reduced” one in less than 1.5 second. It also certifies that it is indeed of minimum height. Thus, in this practical case, the reduction algorithm takes negligible time and using it as a precomputation can greatly speed up the class group computation.

EXAMPLE 4. Finally, we run our implementation on sets of number fields stored in the online Class Group Database [4]. For each degree  $n \in \{3, 4, 5, 7\}$ , we pick 100 number fields having discriminant about  $b$  bits and we try to reduce the polynomial  $T_0$  given in the table. For each degree, we consider two different sizes of discriminants in order to observe how the algorithm behaves as the discriminant grows.

To illustrate the improvement compare to the previously tabulated polynomial, we compute the ratio  $r = \frac{\log H(T_F)}{\log H(T_0)}$ ,  $T_F$  denoting the output polynomial.

TABLE 1. Gain in the size of the height after reduction,  $r = \frac{\log H(T_F)}{\log H(T_0)}$

$n$	$\log_2  \Delta_{\mathbb{K}} $	Early abort					Proven minimal height				
		$c$	avg. $r$	min $r$	max $r$	avg. time	$c$	avg. $r$	min $r$	max $r$	avg. time
3	20	10	0.652	0.435	0.860	7 ms	2	0.651	0.435	0.860	49 ms
	40	10	0.635	0.381	0.741	17 ms	3	0.634	0.381	0.736	131 ms
4	25	10	0.607	0.393	0.868	18 ms	2	0.604	0.393	0.868	1.0 s
	40	10	0.547	0.446	0.806	108 ms	3	0.544	0.446	0.806	6.6 s
5	50	10	0.724	0.567	0.955	109 ms	3	0.715	0.567	0.955	4.8 s
	80	10	0.698	0.555	0.852	820 ms	4	0.695	0.555	0.852	14.7 s
7	80	20	0.712	0.519	0.970	1.7 s	4	0.701	0.519	0.970	749 s
	100	20	0.700	0.537	0.843	6.3 s	—	—	—	—	—

Appendix A. Proof of Theorem 3.2

THEOREM. Let  $\mathbb{K}$  be a number field of degree  $n$ . There exists  $\theta \in \mathcal{O}_{\mathbb{K}} \setminus \mathbb{Z}$  whose minimal polynomial  $P_{\theta}$  satisfies

$$H(P_{\theta}) \leq 3^n \left( \frac{|\Delta_{\mathbb{K}}|}{n} \right)^{\frac{n}{2n-2}}.$$

*Proof.* Given the integral basis  $\omega_1, \dots, \omega_n$  of  $\mathcal{O}_{\mathbb{K}}$  produced by the Round 2 algorithm [6, Algorithm 6.1.8], we consider the lattice  $L$  generated by  $(\Omega_1, \dots, \Omega_n)$ , where  $\Omega_i$  denotes the vector  $[\sigma_1(\omega_i), \dots, \sigma_n(\omega_i)]$ . We know that  $\Omega_1$  is the all ones vector and that  $\|\Omega_1\| = \sqrt{n}$ . In other words,  $\Omega_1$  corresponds to the integer 1 in the number field  $\mathbb{K}$  and it generates the trivial subfield  $\mathbb{Q}$ . Thus, elements of  $\mathcal{O}_{\mathbb{K}} \setminus \mathbb{Z}$  are in a one-to-one correspondence with lattice vectors not on the line through 0 and  $\Omega_1$  – i.e. with vectors that involve at least one of  $\Omega_2, \dots, \Omega_n$  with a non-zero coefficient.

We now consider the lattice  $L^{\perp}$  spanned by the vectors  $\Omega_i^{\perp} = \Omega_i - k_i \Omega_1$ , where  $k_i = \frac{\langle \Omega_1, \Omega_i \rangle}{n}$  so that  $\langle \Omega_1, \Omega_i^{\perp} \rangle = 0$ . Hence,  $L^{\perp}$  is the orthogonal projection of  $L$  with respect to  $\Omega_1$ , and its determinant<sup>†</sup> satisfies

$$\det L = \|\Omega_1\| \det L^{\perp}.$$

In this new lattice  $L^{\perp}$ , of dimension  $n - 1$ , Minkowski’s theorem implies the existence of an element  $\theta^{\perp}$  of small uniform norm, namely

$$\exists \theta^{\perp} \in L^{\perp} \quad \text{s.t.} \quad \|\theta^{\perp}\|_{\infty} \leq (\det L^{\perp})^{\frac{1}{n-1}}.$$

---

<sup>†</sup>We recall that the determinant of a lattice  $L$  is the absolute value of the determinant of any matrix that defines  $L$  and this value is the same for any basis.



Let us denote this vector by

$$\theta^\perp = \sum_{i=2}^n t_i \Omega_i^\perp = \sum_{i=2}^n t_i \Omega_i + \sum_{i=2}^n t_i k_i \Omega_1,$$

with  $t_i \in \mathbb{Z}$ .

Consider the affine line going through  $\sum_{i=2}^n t_i \Omega_i$  and directed by  $\Omega_1$ . On this line, there exists a lattice vector  $\theta = \sum_{i=1}^n t_i \Omega_i$  of minimal length. We know that  $\theta - \theta^\perp = \beta \Omega_1$  for a real number  $\beta$  with  $|\beta| \leq \frac{1}{2}$ . As a consequence,  $\|\theta - \theta^\perp\|_\infty \leq \frac{1}{2}$ . This allows us to conclude that

$$\exists \theta \in L \quad \text{s.t.} \quad \|\theta\|_\infty \leq \left( \frac{\det L}{\sqrt{n}} \right)^{\frac{1}{n-1}} + \frac{1}{2} = \left( \frac{|\Delta_{\mathbb{K}}|}{n} \right)^{\frac{1}{2n-2}} + \frac{1}{2}.$$

Finally, thanks to the Minkowski's bound, we know that  $|\Delta_{\mathbb{K}}| \geq n$  so that  $\left( \frac{|\Delta_{\mathbb{K}}|}{n} \right)^{\frac{1}{2n-2}} \geq 1$  and  $\left( \frac{|\Delta_{\mathbb{K}}|}{n} \right)^{\frac{1}{2n-2}} + \frac{1}{2} \leq \frac{3}{2} \left( \frac{|\Delta_{\mathbb{K}}|}{n} \right)^{\frac{1}{2n-2}}$ . Then we deduce an upper bound for the height of the minimal polynomial  $P_\theta$  of  $\theta$ , using (2.2) and  $M(T) = \prod \max(1, |\tau_i|) \leq (\max_i |\tau_i|)^n$ :

$$H(P_\theta) \leq 2^n \left( \frac{3}{2} \left( \frac{|\Delta_{\mathbb{K}}|}{n} \right)^{\frac{1}{2n-2}} \right)^n = 3^n \left( \frac{|\Delta_{\mathbb{K}}|}{n} \right)^{\frac{n}{2n-2}}.$$

□

### Appendix B. Proof of Proposition 6.1

**PROPOSITION.** *Enumerating all vectors of norm below  $\sqrt{n}$  in one of our weighted lattice can be done in at most  $\text{Poly}(\log |\Delta_{\mathbb{K}}|) \cdot n^{\frac{n}{2e} + o(n)}$  binary operations.*

*Proof.* This is mostly a consequence of the results of [9], which gives a two steps strategy for enumerating the short vectors of a lattice. The first step is to compute an HKZ-reduced basis of the lattice and the second to enumerate the short vectors using this HKZ-reduced basis. Thanks to the high quality of the reduced basis, the enumeration process is greatly speeded up. Note that this second step dominates the complexity of the process.

The complexity of the enumeration process is analyzed in [9, Section 4.1]. Assume that we have a dimension  $d$  lattice given by a basis  $(b_1, \dots, b_d)$  with Gram-Schmidt orthogonalization denoted by  $(b_1^*, \dots, b_d^*)$  and that we wish to enumerate all vectors of  $L_2$ -norm at most  $A$ . Then the complexity expressed in number of arithmetic operations is:

$$2^{O(d)} \cdot \max_{I \subset [1 \dots d]} \left( \frac{A^{|I|}}{\sqrt{d}^{|I|} \prod_{i \in I} \|b_i^*\|} \right).$$

Moreover, [9, Theorem 3] states that for an HKZ-reduced basis, for all subsets  $I$  of  $[1 \dots d]$ , we have:

$$\frac{\|b_1\|^{|I|}}{\prod_{i \in I} \|b_i^*\|} \leq \sqrt{d}^{|I| + \frac{d}{e}}.$$

As a consequence, in terms of arithmetic operations, enumerating all vectors of  $L_2$ -norm at most  $\lambda \|b_1\|$  costs at most:

$$2^{O(d)} \cdot \sqrt{d}^{\frac{d}{e}} \cdot \lambda^d.$$

Note that the preliminary step of computing an HKZ-reduction of the lattice before doing the enumeration does not dominate the complexity of the enumeration itself (see [9, Theorem 2]).

In [9], the complexity is then expressed in terms of bit operations for the case of integer lattices given a bound on the larger integers in the lattice basis. In our case, as already explained after Algorithm 1, we work with an integer lattice approximation whose entries have size polynomial in the size of the input polynomial.

Finally, to apply the result in our case, we need to note that whenever we are enumerating on the lattice corresponding to a weight vector  $(b_1, \dots, b_n)$ , we know that the weight vector  $(\max(b_1 - 1, 0), \dots, \max(b_n - 1, 0))$  did not lead to a successful short vector during the enumeration. Thus, due to the primitivity of the field, no vector of  $L_2$ -norm smaller than  $\sqrt{n}$  exists in this lattice. As a consequence, there is no non-zero vector of  $L_2$ -norm smaller than  $\sqrt{n}/c$  in the current lattice. Thus, we are performing enumeration with a ratio factor  $\lambda$  compared to the short vector at most  $c$ . We conclude that the cost of enumeration in bit operations is at most:

$$\text{Poly}(\log |\Delta_{\mathbb{K}}|) \cdot 2^{O(n)} \cdot n^{\frac{n}{2e}} \cdot c^n \leq \text{Poly}(\log |\Delta_{\mathbb{K}}|) \cdot n^{\frac{n}{2e} + o(n)}.$$

Note that for the first lattice in the enumeration process, i.e. the Cohen and Diaz y Diaz lattice, it is also directly known that no vectors of  $L_2$ -norm smaller than  $\sqrt{n}$  exist in the lattice.  $\square$

### Appendix C. Optimal choice for $c$

Once the complexity analysis is done, it remains to more precisely study the dependence on  $c$  to minimize the constant term. Assuming that the output polynomial defines a number field  $\mathbb{K}$  in  $\mathcal{D}_{n_0, d_0, \alpha, \gamma}$  such that  $\gamma \geq \alpha$  – namely the case of application to class groups –, then the global complexity we found is bounded by

$$\frac{\left(\frac{d_0}{\log c} (\log |\Delta_{\mathbb{K}}|)^\gamma (\log \log |\Delta_{\mathbb{K}}|)^{1-\gamma} (1 + o(1))\right)^n}{n!} \cdot \text{Poly}(\log |\Delta_{\mathbb{K}}|) \cdot 2^{O(n)} \cdot n^{\frac{n}{2e}} \cdot c^n.$$

As we want to find the best  $c$ , we only look at the dependence on  $c$  which can be approximated by:

$$\left(\frac{1}{\log c}\right)^n \cdot c^n = \left(\frac{c}{\log c}\right)^n,$$

and whose minimum is achieved by fixing  $c$  at  $\exp(1)$ .

### References

1. K. BELABAS, ‘Topics in computational algebraic number theory’, *Journal de Théorie des Nombres de Bordeaux* 16 (2004) 19–63.
2. J.-F. BIASSE and C. FIEKER, ‘Subexponential class group and unit group computation in large degree number fields’, *LMS Journal of Computation and Mathematics* 17 (2014) 385–403.
3. J. BUCHMANN, ‘A subexponential algorithm for the determination of class groups and regulators of algebraic number fields’, *Séminaire de Théorie des Nombres, Paris 1988-1989* (1990) 27–41.
4. ‘Class Group Database’, <http://www.mathematik.uni-kl.de/~numberfielddbtables/>, maintained by G. MALLE.
5. H. COHEN and F. DIAZ Y DIAZ, ‘A polynomial reduction algorithm’, *Journal de Théorie des Nombres de Bordeaux* 3 (1991) 351–360.
6. H. COHEN, ‘A course in computational algebraic number theory’, *Graduate Texts in Mathematics* 183 (1991).
7. Y. H. GAN, C. LING and W. H. MOW, ‘Complex lattice reduction algorithm for low-complexity full-diversity MIMO detection’, *IEEE Trans. Signal Processing* 57-7 (2009) 2701–2710.
8. J. L. HAFNER and K. S. MCCURLEY, ‘A rigorous subexponential algorithm for computation of class groups’, *Journal of American Mathematical Society* 2 (1989) 839–850.
9. G. HANROT and D. STEHLÉ, ‘Improved analysis of Kannan’s shortest lattice vector algorithm’, *CRYPTO 2007* (2007) 170–186.
10. D. H. LEHMER, ‘Factorization of certain cyclotomic functions’, *Annals of Mathematics* 34 (1933) 461–479.

11. K. MAHLER, 'An application of Jensen's formula to polynomials', *Journal of the London Mathematical Society* 7 (1960) 98–100.
12. K. MAHLER, 'An inequality for the discriminant of a polynomial', *Michigan Mathematical Journal* 11 (1964) 257–262.
13. M. MIGNOTTE AND P. GLESSER, 'Landau's inequality via Hadmard's', *Journal of Symbolic Computation* 18 (1994) 379–383.
14. M. E. POHST, 'Computational algebraic number theory', *DMV Lecture Notes* 21 (1993).
15. D. SHANKS, 'Class number, a theory of factorization, and genera', *Proceedings of Symposia in Pure Mathematics* 20 (1969) 415–440.
16. D. SHANKS, 'The infrastructure of a real quadratic field and its applications', *Proceedings of the 1972 Number Theory Conference* (1972) 217–224.

Alexandre Gélín  
Sorbonne Universités, UPMC Paris 6,  
UMR 7606, LIP6, 75005, Paris  
France  
alexandre.gelin@lip6.fr

Antoine Joux  
Chaire de Cryptologie, Fondation UPMC,  
Sorbonne Universités, UPMC Paris 6,  
UMR 7606, LIP6, 75005, Paris  
France  
antoine.joux@m4x.org