Computing e-th roots in number fields

ABSTRACT

We describe several algorithms for computing e-th roots of elements in a number field K, where e is an odd prime integer. In particular we generalize Couveignes' and Thomé's algorithms originally designed to compute square-roots in the Number Field Sieve algorithm for integer factorization. Our algorithms cover most cases of e and K and allow to obtain reasonable timings even for large degree number fields and large exponents e. The complexity of our algorithms is better than general root finding algorithms and our implementation compared well in performance to these algorithms implemented in well-known computer algebra softwares. One important application of our algorithms is to compute the saturation phase in the Twisted-PHS algorithm for computing the Ideal-SVP problem over cyclotomic fields in post-quantum cryptography.

CCS CONCEPTS

• **Mathematics of computing → Solvers**; • **Computing methodologies → Computer algebra systems**; • **Security and privacy** → Mathematical foundations of cryptography.

ACM Reference Format:

. 2023. Computing e-th roots in number fields. In *Proceedings of (ISSAC '23).* ACM, New York, NY, USA, 10 pages. https://doi.org/XXXXXXX.XXXXXXX

1 INTRODUCTION

Computing roots of elements is an important step when solving various tasks in computational number theory. It arises for example during the final step of the General Number Field Sieve [10, 13] (NFS). This problem also intervenes during saturation processes while computing the class group or S-units of a number field [5, 7]. Recently, such computations we found to be important to study the Ideal-SVP problem used in Lattice-based Cryptography [14, 19].

Generally speaking, elements are given in a "compact" form, meaning that an element y for which a root needs to be computed is given as a product of relatively small elements $y = \prod_{i=1}^{r} u_i^{e_i}$. Note that the two contexts mentioned previously are somewhat orthogonal one to each other. Indeed for the NFS, the length of the product r is very large while the degree of the number field is typically small, about 10, and one needs to compute square roots ($e = 2$). In saturation processes, such as the ones we are interested in (see [4] for practical cases), r is typically a few times the degree of the field, which is potentially large, say between 100 and 200. Moreover the exponent e may be very large as well, 90 bits. Thus, most strategies to compute an e-th root developed in the NFS context become intractable in this setting if not carefully adapted.

ISSAC '23, June 03–05, 2023, Woodstock, NY

ACM ISBN 978-1-4503-XXXX-X/18/06…$15.00
https://doi.org/XXXXXXX.XXXXXXX

Our contributions

In this article, we explain how to "efficiently" compute an e-th root of an element $y \in K$, where K is a number field and e is an odd prime. We aim at designing a workable method for *large* exponents e and dimension $[K : \mathbb{Q}]$.

- When K and e are such that there are infinitely many prime integers p such that $\forall \mathfrak{p} \mid p, p^{f(\mathfrak{p}|p)} \not\equiv 1 \bmod e$, we reconstruct x from $x \pmod{p_1}, \ldots, x \pmod{p_r}$ using the Chinese Remainder Theorem (CRT), and where each $x \pmod{p_i}$ is itself computed through a CRT procedure. This generalisation of Thomé's square-roots algorithm [24] will be called Double CRT(Algorithms 2 and 3).
- Generically, we can use p-adic lifting when there is an inert prime, or the $\mathfrak{p}$-adic reconstruction of Belabas [3]. Both use a Hensel's lifting, that we adapt to compute e-th roots while avoiding inverse computations in section 3 (Algorithm 1). While these methods work for any e, inert primes do not always exist and $\mathfrak{p}$-adic reconstruction scales badly with the dimension of K. However, in some cases, they can be very efficient and we will use them in our last recursive algorithm.
- When good conditions on K and e are not satisfied, we show how one can adapt Couveignes' approach for square roots [13] to relative extensions of number fields K/k provided $[K : k]$ is coprime to e and infinitely many prime integers p are such that each prime ideal $\mathfrak{p}$ of O_k above p is inert in K. We then turn this strategy into a recursive procedure, calling previous algorithms until reaching the smallest subfield possible (Algorithms 4 and 5).

Experimental results

We ran experiments to evaluate the performances of our algorithms when compared to standard methods and implementations, especially Pari/Gp nfroots [1]. We focused on cyclotomic fields, as they are the main fields used in our application domain, i.e., lattice-based cryptography, but our algorithms extend to other number fields in most cases. All of our implementations are done using SageMath [23] with few optimisations. Meanwhile, we compare with Pari/Gp, and still achieve several orders of performance (between 10 to 10000+ when n and e increase). Thus, our timings could be further improved with optimised C implementations.

Over "good" cases, our CRT generalisation algorithm seems to be clearly more efficient than Pari/Gp nfroots, see Figure 1, and the gap seems to widen when the exponent e increases. We have graphs up to $e = 2^{16}$ but our algorithm scale well and we can use it without any problems for e with 60-bit prime, which would be completely out of scope for Pari/Gp.

Over "bad" cases, experiments show that our generalization of Couveignes' algorithm is more efficient than Pari/Gp nfroots [22], see Figures 2 and 3. Our algorithm is always faster, and the gap with nfroots becomes larger when e and n increase.

[1] Code publicly available at https://github.com/anonymousroots/roots-crt.

Finally, we tested our algorithms in concrete experiments for the saturation using the code of [4].

2 PRELIMINARIES

Let $K = \mathbb{Q}(\alpha)$ be a number field defined by a monic irreducible polynomial $f \in \mathbb{Z}[t]$ of degree n s.t. $f(\alpha) = 0$. In this paper, we shall suppose that we know a reasonable (e.g., LLL-reduced) basis (ω_i) of some order $\mathbb{Z}[\alpha] \subseteq \mathcal{O} \subseteq \mathcal{O}_K$ and $f_0 \geq 1$ such that $f_0\mathcal{O}_K \subset \mathcal{O}$.

Complexities. We use the standard $O(\cdot)$ notation, and let $\mathsf{M}(d, s)$ be the complexity of multiplying two polynomials of degree d with coefficients modulo an s-bit integer, i.e., $\mathsf{M}(d, s) \leq O(d^2 s^2)$ naively.

2.1 Bounds on root coefficients

In all the methods of this paper, we need a bound on the coefficients of the seeked e-th root on the basis (ω_i). Such bounds are usually obtained by computing a lazy estimation of the inverse of a Vandermonde-like matrix linking complex embeddings of elements of K to their coefficients corresponding to (ω_i).

More precisally, let $\Omega = (b_{ij}) \in \mathcal{M}(\mathbb{Q})$ be s.t. $\omega_i = \sum_j b_{ij}\alpha^{j-1}$, and $V_\alpha = \left(\sigma_j(\alpha)^{i-1}\right)$ the Vandermonde matrix corresponding to f. For $x = \sum_i c_i\omega_i$, let $C(x) = (c_i)$ be the coefficient embedding of x and $\Sigma(x) = C(x) \cdot \left(\Omega V_\alpha\right)$ be its canonical embedding to $\mathbb{C}$.

Lemma 1. *Define by $\|A\|_\infty = \max_j \sum_i |a_{ij}|$ the infinity norm of a matrix A. Then we have*

$$\|C(x)\|_\infty \leq \|\Sigma(x)\|_\infty \cdot \left\|V_\alpha^{-1}\Omega^{-1}\right\|_\infty .$$

Proof. This is a direct adaptation of [3, Lem. 3.3], noting that the given definition of the infinity norm for matrices is equivalent to $\|A\|_\infty = \sup_{y\neq 0}\left(\|yA\|_\infty / \|y\|_\infty\right)$. □

Hence, for any $x \in K$, the coefficient norm $\|C(x)\|$ is only a constant factor away from the usual norm $\|\Sigma(x)\|$, and in practice only a loose estimation of this factor is necessary. In particular, for $y \in (K^*)^e$, it is easy to evaluate the size of the (canonical) embedding norm of its e-th root x as $\ln\|\Sigma(x)\|_\infty = \frac{1}{e}\ln\|\Sigma(y)\|_\infty$, and lemma 1 gives the intuition that generically, the coefficients of x are roughly e times smaller than those of y, which holds well in practice.

Moreover, in our particular case of interest, y is given in *compact* form, i.e., there exist $(u_i)_{1\leq i\leq r} \in K^*$ and $(a_i)_{1\leq i\leq r} \in [\![0, e-1]\!]^r$ s.t. $y = \prod_{1\leq i\leq r} u_i^{a_i}$. In this situation, the following lemma shows that the size of the coefficients of $x = y^{1/e}$ does *not* depend on e, but only on the total size of the u_i's.

Lemma 2. *Let $y = \prod_i u_i^{a_i}$ be an element of $(K^*)^e$ in compact form as above, and let x be such that $y = x^e$. Then*

$$\ln\|\Sigma(x)\|_\infty < \textstyle\sum_{1\leq i\leq r} \ln\|\Sigma(u_i)\|_\infty .$$

Proof. For any embedding σ, we have

$$\ln|\sigma(x)| = \tfrac{1}{e}\cdot\ln|\sigma(y)| = \tfrac{1}{e}\cdot\textstyle\sum_{1\leq i\leq r} a_i \ln|\sigma(u_i)|.$$

As $a_i < e$ for all $1 \leq i \leq r$, the lemma follows. □

This motivates the design of algorithms that *never* compute y globally, which is absolutely crucial when e is very large.

2.2 Computing e-th roots in finite fields

Let $\mathbb{F}_q$ be a finite field of characteristic $p > 2$. An important tool for the local-global methods of this paper consists in computing an e-th root in $\mathbb{F}_q$. Let $y \in (\mathbb{F}_q^*)^e$ and let x s.t. $y = x^e$.

If $q \not\equiv 1 \bmod e$, then every element y is an e-th power and we can simply compute x as $y^{e^{-1} \bmod (q-1)}$. If $q \equiv 1 \bmod e$, then we need to work in the subgroup of order $(q-1)/e$ of $\mathbb{F}_q^*$. If $q \not\equiv 1 \bmod e^2$, we can again compute x as $y^{e^{-1} \bmod (q-1)/e}$. Both cases can be treated efficiently in $O(\log^3 q)$ operations.

In very rare cases, we will have no choice but to consider $q \equiv 1 \bmod e^2$, thus we resort to the Adleman-Manders-Miller algorithm [1, Th. IV], which is an adaptation of Tonnelli-Shanks algorithm to the case $e > 2$, and has complexity $O(e\log^4 q)$. Alternatively, we can use generic factorisation methods such as the Cantor-Zassenhaus algorithm [11, Alg. 3.4.6], whose complexity in $O\big(e^2\log^2 q\log^{1+\epsilon} e\cdot(\log e + \log q)\big)$ [20] might be competitive for small $e < \log q$.

3 GENERIC LOCAL-GLOBAL METHODS

In this section, we present two local-global methods that apply unconditionnally w.r.t. e, namely p-adic lifting (e.g., [24, §1.1]), and $\mathfrak{p}$-adic reconstruction [3]. Both methods rely on a p-adic (resp. $\mathfrak{p}$-adic) Newton iteration, or Hensel's lift, that we tweak in the particular case of e-th roots to avoid p-adic (resp. $\mathfrak{p}$-adic) inversions.

Despite their genericity, both methods have severe drawbacks. The p-adic lifting relies on the existence of inert primes, which rules out many families of number fields, e.g., cyclotomic fields of composite conductors. The $\mathfrak{p}$-adic reconstruction does not scale well as the degree of the number field grows, since it requires LLL-reducing a possibly badly-skewed ideal lattice.

Nevertheless, the techniques developed here will be used as a base case for our (recursive) relative Couveignes' method §5.2, which aims at reducing the dimension of the e-th root computation.

3.1 p-adic lifting

The classical p-adic lifting approach applies whenever there exist inert prime ideals p in K. It relies on the fact that K_p, the p-adic completion of K at p, is then a degree $n = [K : \mathbb{Q}]$ unramified extension of $\mathbb{Q}_p$, and the morphism sending a root in K of the defining polynomial f to a root of f in K_p is injective. Thus, a sufficiently good approximation in K_p of the image of $x \in K$ allows us to retrieve directly the coefficients of x. This is done by means of Newton iterations in K_p.

Let $h(z) = z^e - y$, with $y = x^e$ for some $x \in K^*$, and let p be an inert prime of K with $\gcd(p, e) = 1$. The Newton iteration for h writes as follows. At each step $i \geq 0$, suppose that we know a p-adic approximation x_i of x at precision $k = 2^i$, i.e., $x_i = x + O(p^k)$. For $i = 0$, this approximation is found by usual methods in $\mathbb{F}_{p^n}$ (§2.2). Then, $h(x_i) = O(p^k)$ and an approximation x_{i+1} at precision $2k$ is obtained by computing the following iteration modulo p^{2k}:

$$x_{i+1} = x_i - \tfrac{1}{e}\cdot\left(x_i - \tfrac{y}{x_i^{e-1}}\right).$$

Let $B > 0$ be an upper bound on the coefficients of x, then this iteration is performed $\kappa = \left\lceil \log_2 \max\{1, \log_p 2B\}\right\rceil$ times. In particular, only the knowledge of y at precision p^{2^κ} is needed. This is

especially useful when e is large, since in that case the size of the coefficients of x is roughly e times smaller than for y (see §2.1).

In practice, each iteration above requires an inverse computation. For $e = 2$, a known trick to avoid this consists in first computing the *inverse* square root, then using $y^{1/2} = y \cdot y^{-1/2}$ [9, §2]. We adapt this trick in our case by seeking first a root x of the polynomial

$$g(z) = y^{e-1}z^e - 1.$$

Then $(y \cdot x)$ verifies $(yx)^e = y \cdot (y^{e-1}x^e) = y$ as expected. The Newton iteration for g now writes without inverses as

$$x_{i+1} = x_i - \tfrac{1}{e}x_i\big(y^{e-1}x_i^e - 1\big). \tag{1}$$

The complete p-adic lifting using this new iteration is summarized in algorithm 1.

Algorithm 1 p-adic lifting for e-th root

Require: $y \in (K^*)^e$ in compact form $y = \prod_i u_i^{a_i}$, p an inert prime in K with $\gcd(p, e) = 1$.
Ensure: $x \in K^*$ such that $y = x^e$.
1: Compute B s.t. $\|C(x)\|_\infty \le B$ ▷ Using lemmas 1 and 2
2: $\kappa \leftarrow \lceil \log_2 \max\{1, \log_p 2B\} \rceil$
3: $a \leftarrow y^{e-1} \mod p^{2^\kappa}$ ▷ Reduce u_i's first
4: $x_0 \leftarrow (a \mod p)^{1/e}$ in $\mathbb{F}_{p^n} \simeq {}^{O}\!/_{pO}$ ▷ Using §2.2
5: **for** $0 \le i < \kappa$ **do**
6: $\quad x_{i+1} = x_i - \frac{1}{e}x_i\big(ax_i^e - 1\big)$ ▷ Work in O modulo $p^{2^{i+1}}$
7: **end for**
8: $x \leftarrow a \cdot x_\kappa \mod p^{2^\kappa}$
9: **return** x with coefficients mapped in $[-B, B]$

Proposition 1. *Algorithm 1 is correct, and runs in time at most* $O\big(\log e \cdot (r + \log s)\mathsf{M}(n, s) + e \log^4 p^n\big)$*, where s is the total input size.*

Proof. Let x be a root of $g(z) = y^{e-1}z^e - 1$. At any stage $i \ge 0$, let x_i be a p-adic approximation of x at precision $k = 2^i$, i.e., $x = x_i + O(p^k)$, and let $\varepsilon_i = g(x_i) = O(p^k)$. We shall show that $\varepsilon_{i+1} = g(x_{i+1}) = O(p^{2k})$, which implies correctness. Working modulo p^{2k}, and plugging the Newton iteration formula for x_{i+1} into g, we get

$$\begin{aligned}\varepsilon_{i+1} &= y^{e-1}\big(x_i - \tfrac{1}{e}x_i g(x_i)\big)^e - 1 = y^{e-1}\big(x_i - \tfrac{1}{e}x_i\varepsilon_i\big)^e - 1\\ &= -1 + y^{e-1}x_i^e\big(1 - e\tfrac{1}{e}\varepsilon_i + O(\varepsilon_i^2)\big)\\ &= -1 + (\varepsilon_i + 1) - \varepsilon_i(\varepsilon_i + 1) + O\big(\varepsilon_i^2\big) = O\big(\varepsilon_i^2\big) = O\big(p^{2k}\big).\end{aligned}$$

As for the complexity, note that, by lemma 2, $\log B = O(s)$ so that $\kappa = O(\log s)$. Computing a using multi-exponentiation and a subproduct tree costs at most $O\big(r \log e \cdot \mathsf{M}(n, s)\big)$ using $a_i < e$. By §2.2, computing $a^{1/e}$ in $\mathbb{F}_{p^n}$ costs $O(e \log^4 p^n)$ in the worst case. Each of the $O(\log s)$ Newton iterations at precision $k = 2^i$ costs at most $O\big(\log e \cdot \mathsf{M}(n, k \log p)\big)$ for a total of $O\big(\log e \log s \cdot \mathsf{M}(n, s)\big)$. □

3.2 $\mathfrak{p}$-adic reconstruction

When the field K contains no inert primes, the above method can still be used, for any unramified prime ideal $\mathfrak{p}$ of inertia degree $f(\mathfrak{p}|p) < n$, to obtain a $\mathfrak{p}$-adic approximation in $K_\mathfrak{p}$, the completion of K at $\mathfrak{p}$, which is an unramified extension of $\mathbb{Q}_p$ of degree $f(\mathfrak{p}|p)$.

Starting from a low-precision e-th root in ${}^{O_K}\!/_{\mathfrak{p}} \simeq \mathbb{F}_{p^{f(\mathfrak{p}|p)}}$, eq. (1) allows for its lifting modulo $\mathfrak{p}^a$ for any a. If a is large enough, it is possible to reconstruct the root in K from this $\mathfrak{p}$-adic embedding approximation, as is done in [3, §3], by solving a Bounded Distance Decoding problem in the LLL-reduced lattice corresponding to $\mathfrak{p}^a$. The main drawback of this method is that the ideal $\mathfrak{p}^a$ is all the more badly-skewed that the inertia degree of $\mathfrak{p}$ is small and n is large, so that the LLL-reduction quickly dominates [3, §3.7].

In practice, we estimate a using the analysis of [3]. Suppose that we have a bound B' on the coefficient norm of x. By [3, Lem. 3.7], the reconstruction succeeds when $r_{\max} > B'$, where $r_{\max}$ is explicitly given by [3, Lem. 3.8], which provides a lower bound on a as in [3, Lem. 3.12]. Hence, using $\gamma \approx 1.022$ as the root-Hermite factor achieved by LLL [16], we start from the smallest $a = 2^\kappa$ such that

$$a > \frac{n}{f(\mathfrak{p}|p) \cdot \ln p}\Big(\ln 2B' + \big(\tfrac{n(n-1)}{4} - 1\big)\ln\gamma\Big).$$

This value of a is controlled *a posteriori* by computing the corresponding $r_{\max}$, checking whether $r_{\max} > B'$ and doubling a while necessary. In practice, the above estimation is rarely invalidated.

4 USING CHINESE REMAINDER THEOREM: THE EASY CASES

In this section, we describe how one can compute e-th roots using the Chinese Remainder Theorem in number fields. when e is such that

$$\exists_\infty \text{ primes } q \text{ s.t. } \forall \mathfrak{q} \mid q, q^{f_\mathfrak{q}} \not\equiv 1 \mod e. \tag{2}$$

This condition ensures that none of the residue fields contain a primitive e-th root of unity. In cyclotomic fields of conductor m, this condition is equivalent to assuming $\gcd(e, m) = 1$, since in that case all primes verify the assumption and conversely.

In the context of the Number Field Sieve, Thomé described a CRT-based method to compute square-roots [24, §4]. A major problem for $e = 2$ is to guess the correct signs modulo each prime ideal, which is handled by solving a knapsack problem. However, as n grows as well as e, this approach quickly becomes intractable.

4.1 A CRT-based method for e-th roots

First, we show in algorithm 2 how to retrieve an e-th root modulo q, where q is a prime integer verifying (2), using the Chinese Remainder Theorem in number fields.

Algorithm 2 Number field CRT for e-th root mod q

Require: $y \in (K^*)^e$ in compact form $y = \prod_i u_i^{a_i}$, and an unramified prime q in K verifying (2).
Ensure: $x \equiv y^{1/e} \mod (q)$.
1: $S \leftarrow \{\mathfrak{q}_1, \ldots, \mathfrak{q}_r\} := \{\mathfrak{q}, \mathfrak{q} \mid (q)\}$
2: **for** $\mathfrak{q} \in S$ **do**
3: $\quad y_\mathfrak{q} \leftarrow y \mod \mathfrak{q}$ ▷ Reduce u_i's modulo $\mathfrak{q}$ first
4: $\quad x_\mathfrak{q} \leftarrow y_\mathfrak{q}^{1/e} \mod \mathfrak{q}$ ▷ Using §2.2
5: **end for**
6: **return** $\mathrm{CRT}_K((x_\mathfrak{q})_{\mathfrak{q}\in S}, (\mathfrak{q})_{\mathfrak{q}\in S})$

Proposition 2. *Algorithm 2 is correct, and runs in time at most* $O\big(rn \log q \cdot \mathsf{M}(n, \log q) + r \log^2 e\big)$.

Proof. We first prove correctness. One has $(q) = \prod_{\mathfrak{q}\in S} \mathfrak{q}$, since q is unramified in K, which gives

$$O_{K/(q)} \cong \prod_{\mathfrak{q}|(q)} O_{K/\mathfrak{q}} \cong \prod_{\mathfrak{q}|(q)} \mathbb{F}_{q^{f(\mathfrak{q}|q)}}.$$

For all $\mathfrak{q} \in S$, the condition $q^{f_\mathfrak{q}} \not\equiv 1 \bmod e$ ensures that $\mathbb{F}^*_{q^{f_\mathfrak{q}}} = (\mathbb{F}^*_{q^{f_\mathfrak{q}}})^e$, so any element of $O_{/\mathfrak{q}}$ has a *unique* e-th root. Consequently, step 4 of algorithm 2 is properly defined and $x \equiv x_\mathfrak{q} \bmod \mathfrak{q}$. Thus the output of step 6 is indeed congruent to x modulo q.

As for the complexity, the first step factorises a polynomial of degree n modulo q which can be done in $O\big(n^2 \log^2 q \log^{1+\epsilon} n \cdot (\log q + \log n)\big)$ using Cantor-Zassenhaus. For a given $\mathfrak{q} \mid q$, computing $y_\mathfrak{q}$ costs at most $O\big(r f_\mathfrak{q} \log q \cdot \mathsf{M}(f_\mathfrak{q}, \log q)\big)$ (mapping each a_i modulo $q^{f_\mathfrak{q}}-1$), and the e-th root in $\mathbb{F}_{q^{f_\mathfrak{q}}}$ costs $O\big(f_\mathfrak{q} \log q \cdot \mathsf{M}(f_\mathfrak{q}, \log q)\big)$. Since all $f_\mathfrak{q}$ sum to n, the whole loop costs at most $O\big(rn \log q \cdot \mathsf{M}(n, \log q)\big)$, and the last CRT step is negligible, as it can be done in $O\big(n^2 \log^2 q\big)$ [26, Th. 5.7]. Finally, note that if e is large, we can reduce it as well as the a_i's once modulo $q^n - 1$, and use these smaller versions thereafter, yielding the extra $O(r \log^2 e)$ term. □

Remark 1. Note that the complexity of algorithm 2 does not depend heavily on e. Furthermore, the proof shows that it is best to consider totally split primes in K, which always exist. In this case, the complexity drops to $O\big(rn \log^3 q + r \log^2 e\big)$.

We now explain how to compute x *via* computations modulo primes $q_1, \ldots, q_r$ that all verify (2).

Algorithm 3 Double CRT for e-th root

Require: $y \in (K^*)^e$ in compact form $y = \prod_i u_i^{a_i}$.
Ensure: $x \in K^*$ such that $y = x^e$.
1: Compute B s.t. $\|C(x)\|_\infty \le B$ ▷ Using lemmas 1 and 2
2: Choose primes $q_1, \ldots, q_k$ verifying (2) s.t. $\prod_i q_i \ge 2B$.
3: **for** $1 \le i \le k$ **do**
4: $\quad x_i \leftarrow y^{1/e}$ in $O_{/q_i O}$ ▷ Using algorithm 2
5: **end for**
6: $x \leftarrow \mathrm{CRT}_\mathbb{Z}\big((x_i)_i, (q_i)_i\big)$ ▷ Coefficient by coefficient
7: **return** x with coefficients mapped in $[-B, B]$

Proposition 3. *Algorithm 3 is correct, and runs in time at most* $O\big(rs(\mathsf{M}(n,\kappa) + \log^2 e) + n \cdot s^2\big)$, *where s is the total input size.*

Proof. Step 2 computes the solution modulo the ideal generated by q_i, for each $i \in [\![1, k]\!]$. Thus the CRT in step 6 computes x mod (Q), where $Q = \prod_i q_i$, and $Q > 2B$ ensures that there is a unique element $z \equiv x \bmod (Q)$ with all coefficients in $[-B, B]$.

Let $\kappa = \max_i \log q_i$. To fix ideas, κ is taken as to fit a machine word, and all q_i's are chosen evenly, thus $\kappa k \approx \sum \log q_i = O(\log B)$. Using proposition 2, computing the roots modulo all q_i's has a total cost of $O\big(rn \log B \cdot \mathsf{M}(n,\kappa) + rk \log^2 e\big)$ and by [26], the final CRT costs $O(nk^2)$. We conclude using $k = O(\log B) = O(s/n)$ by lemma 2. □

Remark 2. One can improve the complexity of Algorithm 3 as well as its practical efficiency by using product trees both for interpolation and product steps encountered. Our implementation take advantage of such constructions.

4.2 Bad cases: existence of cyclotomic subfield

In this section, we prove the following result, which characterises number fields which are bad fields for e.

Theorem 1. *Let K be a number field. The two following assertions are equivalent :*

(*i*) *For almost all prime $p \in \mathbb{N}$, $\forall \mathfrak{p} \mid p$, $p^{f(\mathfrak{p}|p)} \equiv 1 \bmod e$;*
(*ii*) *$\mathbb{Q}(\zeta_e)$ is a subfield of K.*

For this we will need a result due du Bauer mentioned in [18].

Notation. Given L/K a number field extension we denote by $P(L/K)$ the set $\{\mathfrak{p}$ unramified prime of $K \mid \exists \mathfrak{P}, f(\mathfrak{P}|\mathfrak{p}) = 1\}$.

Lemma 3 (Bauer in [18])**.** *If L/K is Galois and M/K is an arbitrary finite extension, then $P(M/K) \dot\subseteq P(L/K) \iff L \subseteq M$.*

Proof of Theorem 1. The first assertion is true for $\mathbb{Q}(\zeta_e)$, so $(ii) \implies (i)$ is clear by multiplicativity of the inertia degree. Now assume (i). Let $p \in P(K/\mathbb{Q})$ and $\mathfrak{p}$ s.t. $f(\mathfrak{p}|p) = 1$. Condition (i) implies that almost all such prime satisfies $p \equiv 1 \bmod e$, which implies p is completely split in $\mathbb{Q}(\zeta_e)$, i.e., $p \in P(\mathbb{Q}(\zeta_e)/\mathbb{Q})$. Consequently, we have $P(K/\mathbb{Q}) \dot\subseteq P(\mathbb{Q}(\zeta_e)/\mathbb{Q})$ and Lemma 3 gives (ii). □

In all generality, the class of fields for which Algorithm 2 cannot be used is larger than the one described by Theorem 1. However, if one considers Galois fields only, the two are equivalent.

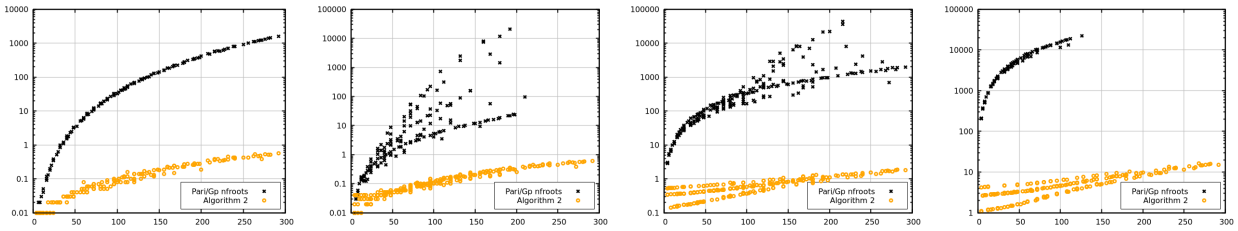
4.3 Experimental results

We compared in practice algorithm 3 to standard algorithms and implementations such as Pari/Gp `nfroots`. For specific exponents $e \in \{3, 71, 1637, 13099\}$, and focusing on suitable cyclotomic fields, we computed the average time taken to compute the e-th roots of $y = x^e$ where x is a random element with coefficients of bit size $\log B \in \{1, 50, 100\}$. We stress that with this protocol, algorithm 3 does *not* take advantage of being designed for treating compact forms. The results we obtained for $\log B = 100$ can be found in fig. 1.

Remark 3. For small exponents such that $3e \leqslant [K : \mathbb{Q}]$, the function nfroots from Pari/Gp uses Trager's method [25]. This is typically the case in our experiments when $e = 3$. Otherwise, it follows ideas developed in [3, 15] for example. This is the case when $e = 71, 1637, 13099$ in these experiments.

From fig. 1, we see that our implementation of algorithm 3 using SageMath is, in all cases, much more efficient than nfroots when the dimension increases. Further, the gap increases with the exponent. Remark, e.g., in fig. 1(b), that Algorithm 3 is more stable than Pari/Gp nfroots.

Finally, as expected, the performances of our algorithm are not much influenced by the size of the exponent e, to the point where it is perfectly fine to compute e-th roots for very large e's. For example, table 1 shows timings for large e's in a real-life application, the biggest of which being $e = 207293548177$, a 37-bits prime for which the root in dimension $n = 198$ is computed in half an hour.



(a) $e = 3$ (b) $e = 71$ (c) $e = 1637$ (d) $e = 13099$

Figure 1: Timings (s) for `nfroots` and Algorithm 3 plotted against the dimension, over cyclotomic fields and $e \in \{3, 71, 1637, 13099\}$.

5 A RELATIVE COUVEIGNES' METHOD: THE BAD CASES

In this section we will describe how one can compute e-th roots in the bad cases, i.e., when for all primes $p \in \mathbb{N}$, there exist at least one $\mathfrak{p} \mid p$ s.t. $p^{f_\mathfrak{p}} \equiv 1 \mod e$.

5.1 A relative Couveignes' method

Couveignes' method for square-root computation [13, 24] allows one to identify together the roots modulo a set of inert primes $p_1, \ldots, p_r$, assuming that the degree of the number field $[K : \mathbb{Q}]$ is odd. In the following, we show how to generalise this method to $e \geqslant 3$.

A key ingredient of Couveignes' method is the fact that the norm maps $\mathrm{N}_K : K \to \mathbb{Q}$ and $\mathrm{N}_{\mathbb{F}_{p^n}} : {}^{O_K}\!/_{(p)} \simeq \mathbb{F}_{p^n} \to \mathbb{F}_p$ are coherent when p is an inert prime integer, meaning that for any $x \in K$ we have $\mathrm{N}_K(x) \bmod p = \mathrm{N}_{\mathbb{F}_{p^n}}(x \bmod p)$. We will use a generalisation of this property to relative extensions.

Lemma 4. *Consider K/k a number field extension, $\mathfrak{p}$ an inert prime ideal of O_k and $\mathfrak{P}$ the prime ideal of O_K above $\mathfrak{p}$. Then the following diagram is commutative.*

$$\begin{array}{ccc} O_K & \twoheadrightarrow & {}^{O_K}\!/_{\mathfrak{P}} \\ {\scriptstyle \mathrm{N}_{K/k}}\downarrow & \circlearrowright & \downarrow{\scriptstyle \mathrm{N}} \\ O_k & \twoheadrightarrow & {}^{O_k}\!/_{\mathfrak{p}} \end{array}$$

Proof. When the extension K/k is Galois, this can be deduced by the fact that the norm of an element (in O_K or ${}^{O_K}\!/_{\mathfrak{P}}$) is the product of its conjugates and that the Galois group of K/k is canonically isomorphic to the Galois group over the residue fields extension.

If we are not in the Galois case, we can proceed as follows. Elements of O_K can be seen as polynomials with coefficients in k with degree less than $[K : k]$ as we have the isomorphism $O_K \cong O_k[T]/(P_{K/k}(T))$ for some irreducible polynomial $P_{K/k} \in k[T]$. Similarly elements of ${}^{O_K}\!/_{\mathfrak{P}}$ are class of polynomials in ${}^{O_k}\!/_{\mathfrak{p}}$. We will denote these residue fields by $\mathbb{F}_\mathfrak{P}$ and $\mathbb{F}_\mathfrak{p}$ respectively. Since $\mathfrak{P} \mid \mathfrak{p}$ is inert, one can choose the same polynomial $P_{K/k}$ as defining polynomial of $\mathbb{F}_\mathfrak{P}/\mathbb{F}_\mathfrak{p}$, i.e., one has $\mathbb{F}_\mathfrak{P} \cong \mathbb{F}_\mathfrak{p}[T]/(P_{K/k}(T))$. Consequently, if α is an element of K written as $\sum_{i=0}^{[K:k]-1} \alpha_i T^i$ with $\alpha_i \in O_k$ for each $0 \leqslant i < [K : k]$, then its embedding into $\mathbb{F}_\mathfrak{P}$ is $\sum_{i=0}^{[K:k]-1} \overline{\alpha}_i T^i$ where $\bar{\cdot} : O_k \to \mathbb{F}_\mathfrak{p}$. Now recall that the relative norm of an element α in a field extension E/F consists in the determinant of the multiplication map $[\alpha]$ when seen as F-linear map of E. Because both extensions K/k and $\mathbb{F}_\mathfrak{P}/\mathbb{F}_\mathfrak{p}$ are given by the same defining polynomial, one can consider the action $[\alpha]$ and $[\overline{\alpha}]$ over the "same" basis. In short, $[\alpha]$ and $\bar{\cdot}$ commute. This transfers to the determinant so we have $\det[\alpha] \bmod \mathfrak{p} = \det[\alpha \bmod \mathfrak{P}]$. □

Another key ingredient of Couveignes' method for $e = 2$ is that, as soon as the degree of the field is odd, $\mathrm{N}(\pm x) = \pm \mathrm{N}(x)$. We present an extension of this property to e-th roots of unity and relative extensions.

Lemma 5. *Let E/F be a field extension of finite degree and $e \in \mathbb{N}^*$ such that ζ_e belongs to both E and F. Assume additionnally that $\gcd([E : F], e) = 1$. Then for any $y \in (E^*)^e$, the norm map $\mathrm{N}_{E/F}$ induces a bijection between $Z_E(X^e - y)$ and $Z_F\big(X^e - \mathrm{N}_{E/F}(y)\big)$.*

Proof. The set $Z_E(X^e - y)$ is of the form $\{\zeta_e^i x \mid i \in [\![0, e-1]\!]\}$ where x is a fixed e-th root of y in E. Similarly, we have $Z_F\big(X^e - \mathrm{N}_{E/F}(y)\big) = \{\zeta_e^i \mathrm{N}_{E/F}(x) \mid i \in [\![0, e-1]\!]\}$. Now, since $\zeta_e \in F$,

$$\mathrm{N}_{E/F}\big(\zeta_e^i x\big) = \zeta_e^{i[E:F]} \mathrm{N}_{E/F}(x).$$

Thus, it suffices to prove the result for the e-th roots of unity. This follows from the fact that $\gcd([E : F], e) = 1$. □

Remark 4. Note that Lemma 5 can be applied to number field extensions or finite field extensions. Since both O_K and $O_K/\mathfrak{p}$ contain a primitive e-th root of unity, Lemma 5 applies to both sides of the commutative diagram from Lemma 4.

Lemma 6. *Consider K/k a number field extension and $e \in \mathbb{N}^*$ such that ζ_e belongs to both K and k. Assume additionnaly that $\gcd([K : k], e) = 1$. Then Algorithm 4 is correct and its running time is in*

$$O\big(rn \log p \cdot \mathsf{M}(n, \log p) + e^2 \log^2 e \cdot (\log e + n \log p) \cdot \mathsf{M}(n, \log p)\big). \quad (3)$$

Proof. The output is correct if for each $i \in [\![1, r]\!]$, x_i is the embedding in ${}^{O_K}\!/_{\mathfrak{P}_i}$ of a fixed e-th root of y in K, denoted by x. This is ensured by the combination of Lemma 4 and Lemma 5 which state that in each residue field ${}^{O_K}\!/_{\mathfrak{P}_i}$ there exactly one element of X_i whose norm over ${}^{O_k}\!/_{\mathfrak{p}_i}$ is a_i.

As for the complexity, the two first steps factorise a polynomial of degree smaller than n modulo p which can be done in

Algorithm 4 Relative Couveignes' method for e-th root modulo p

Require: A number field extension K/k, e, $y \in (K^*)^e$, $a = \mathrm{N}_{K/k}(y)^{1/e}$ a fixed e-th root of $\mathrm{N}_{K/k}(y)$ in O_k, and p a prime integer s. t. each $\mathfrak{p}$ above p in O_k is inert in K/k.
Ensure: $x \equiv y^{1/e} \bmod (p)$ in K.
1: $S \leftarrow \{\mathfrak{p}_1, , \ldots, \mathfrak{p}_g \mid pO_k = \prod_i \mathfrak{p}_i\}$
2: $T \leftarrow \{\mathfrak{P}_1, , \ldots, \mathfrak{P}_g \mid \mathfrak{p}_i O_k = \mathfrak{P}_i\}$
3: **for** $i = 1 \ldots g$ **do**
4: $\quad x_i \leftarrow y^{1/e} \bmod \mathfrak{P}_i$ ▷ Pick one root in the residue field
5: $\quad a_i \leftarrow a \bmod \mathfrak{p}_i$ ▷ Expected relative norm mod $\mathfrak{p}_i$
6: $\quad z_i \leftarrow a_i / \mathrm{N}_{K/k}(x_i)$ ▷ e-th root of unity mod $\mathfrak{p}_i$
7: $\quad x_i \leftarrow x_i \cdot z_i^{[K:k]^{-1} \bmod e}$
8: **end for**
9: **return** $\mathrm{CRT}((x_i)_{i \in [\![1,g]\!]}, (\mathfrak{P}_i)_{i \in [\![1,g]\!]})$.

$O\big(n^2 \log^2 p \log^{1+\epsilon} n \cdot (\log p + \log n)\big)$ using Cantor-Zassenhaus. Fix $i \in [\![1, g]\!]$. Then computing $y \bmod \mathfrak{P}_i$ costs at most $O\big(rn^2 \cdot \mathsf{M}(\log p) + rf(\mathfrak{P}_i|p) \log p \cdot \mathsf{M}(f(\mathfrak{P}_i|p), \log p)\big)$ (mapping each a_i modulo $p^{f(\mathfrak{P}_i|p)} - 1$ and where the first term account for the embedding of each u_i in the residue field), and the e-th root in $\mathbb{F}_{\mathfrak{P}_i}$ costs less than $O\big(e^2 f(\mathfrak{P}_i|p)^2 \log^2 p \log^{1+\epsilon} e \cdot (\log e + f(\mathfrak{P}_i|p) \log p)\big)$ using again Cantor-Zassenhaus since we cannot ensure that e can be inverted modulo $p^{e(\mathfrak{P}_i|p)} - 1$. Now step 5 can be computed in $O([k : \mathbb{Q}]^2 M(\log p))$ and step 6 costs at most $O(f(\mathfrak{P}_i|p) \log p \cdot M(n, \log p))$ (the norm being computed as an exponentiation in the residue field). Step 7 can be done in $O\big(f(\mathfrak{P}_i|p) \cdot \log p \cdot \mathsf{M}(f(\mathfrak{P}_i|p), \log p)\big)$. The whole loop then costs

$$\begin{aligned} O\big(&grn^2 \cdot \mathsf{M}(\log p) + rn \log p \cdot \mathsf{M}(n, \log p) \\ &+ e^2 n^2 \log^2 p \cdot \log^{1+\epsilon} e \cdot (\log e + n \log p) \\ &+ gn^2 \cdot \mathsf{M}(\log p) + n \log p \cdot \mathsf{M}(n, \log p)\big), \end{aligned}$$

which simplifies (?) to

$$\begin{aligned} O\big(&rn \log p \cdot \mathsf{M}(n, \log p) \\ &+ e^2 \log^2 e \cdot (\log e + n \log p) \cdot \mathsf{M}(n, \log p)\big). \end{aligned}$$

The last CRT step is again negligible, as it can be done in $O\big(n^2 \log^2 p\big)$ [26, Th. 5.7].

□

Considering a fixed root in $k \bmod \mathfrak{p}$ allows us to choose the corresponding roots in K, removing the potential enumeration processes due to CRT. In particular, lemma 6 shows that this allows computing a root $x \bmod p \in K$ for a given prime integer. The same idea applies to several primes, see algorithm 5.

Algorithm 5 Relative Couveignes' method for e-th root

Require: A number field extension K/k, $e \geqslant 3$, $y \in (K^*)^e$ in compact form $y = \prod_i u_i^{a_i}$.
Ensure: $x \in K^*$ such that $y = x^e$.
1: $a \leftarrow \mathrm{N}_{K/k}(y)^{1/e}$ ▷ Using §3 or this algorithm in k
2: Choose primes $p_1, \ldots, p_l$ prime integers s.t. each prime ideal above p_i in O_k is inert in K/k and $\prod_i q_i \geq 2B$.
3: **for** $i = 1..l$ **do**
4: $\quad x_i \leftarrow \mathsf{RelativeCouvMod}_{\mathfrak{p}}(e, y, a, p_i)$ ▷ Algorithm 4
5: **end for**
6: **return** $\mathrm{CRT}_{\mathbb{Z}}((x_i)_{i \in [\![1,l]\!]}, (p_i)_{i \in [\![1,l]\!]})$.

Notation. We will denote by $R(K, s, e)$ the cost of computing the eth root of an element of size s in the number field K.

Theorem 2. *Consider K/k a number field extension and $e \in \mathbb{N}^*$ such that ζ_e belongs to both K and k. Assume additionnally that* $\gcd([K : k], e) = 1$. *Then algorithm 5 is correct and runs in*

$$R(k, ns, e) + O\big(r \cdot \mathsf{M}(n, ns) + (r + e^2 \log^2 e) \cdot ns \cdot \mathsf{M}(n, \kappa) + se^2 \log^3 e\big). \tag{4}$$

Proof. Using Lemma 6 we can conclude that for each $i \in [\![1, l]\!]$, we have $x_i \equiv x \bmod p_i$. Thus, the correctness is ensured. Regarding the complexity, step 1 can be computed in less than $O(r \cdot \mathsf{M}(n, ns)) + R(k, ns, e)$ where the first term is the cost of the relative norm. Let $\kappa = \max_i \log p_i$. To fix ideas, κ is taken as to fit a machine word, and all p_i's are chosen evenly, thus $\kappa \cdot l \approx \sum \log p_i = O(\log B)$. Using lemma 6, computing the roots modulo all p_i's has a total cost of $O\big(rn \log B \cdot \mathsf{M}(n, \kappa) + le^2 \log^3 e + e^2 \log^2 e \cdot \log B \cdot n \cdot \mathsf{M}(n, \kappa)\big)$ and by [26], the final CRT costs $O(nl^2)$. We conclude using $l = O(\log B) = O(s/e)$ by lemma 2 that we have a final cost of

$$R(k, ns, e) + O(r \cdot \mathsf{M}(n, ns)) + O\big((r + e^2 \log^2 e) \cdot ns \cdot \mathsf{M}(n, \kappa) + se^2 \log^3 e\big).$$

□

Remark 5. Under the specific conditions required on the field extension K/k and the prime decomposition in it, note that fixing one e-th root of $\mathrm{N}_{K/k}(y)$ allows us to remove the cost of both types of CRT.

Note that for Algorithm 5 to be usable as a generic method over a fixed field K together with an exponent e, it is necessary that there is a subfield $k \hookrightarrow K$ ensuring the existence of infinitely many prime integers with the right splitting condition mentioned. If K/k is Galois, one can deduce from Chebotarev density theorem that this is equivalent to K/k being cyclic [12, 18].

Note that one of the most costly steps is the computation of $\mathrm{N}_{K/k}(y)$. This cost can is mitigated by the use of a compact representation. The complexity $\mathsf{R}(K, B, e)$ will depend on the algorithm used to compute the e-th root. In this setting, computing the full product $\prod_i \mathrm{N}_{K/k}(u_i)^{e_i}$ in k shall always be avoided in order to avoid coefficient explosion.

5.2 A recursive relative Couveignes' method

Algorithm 5 can be turned into a recursive algorithm, noting that one important step is to compute the e-th root of $\mathrm{N}_{K/k}(y)$. We will

focus on the Galois case. Thus, let us fix a Galois number field K and denote by G its Galois group.

Recall that we assume that $\mathbb{Q}(\zeta_{e^s}) \hookrightarrow K$ for some s as we are in bad cases. Let k be a subfield of K such that Algorithm 5 can be applied, i.e. $\gcd([K:k], e) = 1$ and $\mathrm{Gal}(K/k)$ is cyclic. In order to apply Algorithm 5 recursively for the computation of $\mathrm{N}_{K/k}(y)^{1/e}$, one needs the existence of a subfield $l \hookrightarrow k$ satisfying the two same conditions. We thus get Proposition 4 which describes Galois extensions for which one can apply a recursive version of Algorithm 5.

Proposition 4. *Consider K/k a Galois extension of number fields. Then one can apply a recursive version of Algorithm 5 with respect to K/k and $e \in \mathbb{N}$ if, and only if, K/k is Abelian and* $\gcd([K:k], e) = 1$.

Proof. If K/k is suitable for a recursive version of Algorithm 5, then there is a tower of subfields

$$k = K_0 \hookrightarrow K_1 \hookrightarrow \cdots \hookrightarrow K_{r-1} \hookrightarrow K_r = K$$

such that for all $i \in [\![0, r-1]\!]$ the extension K/K_{i+1} is cyclic and $\gcd([K_{i+1} : K_i], e) = 1$. Since one has $G = \mathrm{Gal}(K/k) \cong \bigoplus_{i=0}^{r-1} \mathrm{Gal}(K_{i+1}/K_i)$ and $[K:k] = \prod_{i=1}^{r-1} [K_{i+1} : K_i]$, K/k is clearly Abelian with $\gcd([K:k], e) = 1$. Conversely, denoting by H the Galois group of K/k, if this extension is Abelian then H admits a cyclic refinement [17]. The part on the dimension is clear as well. □

5.3 Experimental results

We report now on experimental results we obtained from our implementation of Algorithm 5 in SageMath [23]. In these simple experiments we chose to consider cyclotomic fields of the form $\mathbb{Q}(\zeta_{pq})$ where p is a prime integer and $q \in \mathbb{N}$ satisfies suitable conditions. The prime p is constant in each experiment.

We considered two sets of experiments. In the first, we chose q to be a prime integer as well, and fixed $[K:k] = p$ thus $e = q$ and $k = \mathbb{Q}(\zeta_q)$. In the second, $e = p$ so one consider $k = \mathbb{Q}(\zeta_e)$ and $[K:k] = q$. We chose to compare the performances of our implementation to the ones of Pari/Gp `nfroots`. Results can be found in figs. 2 and 3.

One can see that Algorithm 5 is more efficient than `nfroots` in all cases but $p = e = 3$. In this last case, the gap between the two performances tends to diminish when $[K:\mathbb{Q}]$ is increasing while it seems to widen in otherwise. Recall that Pari/Gp `nfroots` uses Trager's method [25] when $3e \leqslant [K:\mathbb{Q}]$, which is the case in our experiments for which $e = p$. Otherwise, Pari/Gp `nfroots` uses the ideas developped in [3, 15]. All of these observations tend to show that our generalisation of Couveignes' method is more effective than previous algorithms used to compute e-th roots.

An important constatation is that most of the running time of algorithm 5 comes either from the computation of $\mathrm{N}_{K/k}(y)$ when $[K:k]$ is large (as predicted by the theoretical complexity), see Figure 3. In practice it amounts for more than 90% of the computation time. Improving the efficiency of this task would render our relative Couveignes' method even more impactful.

6 SATURATION: A REAL-LIFE EXAMPLE

In this section, we first briefly describe the saturation process that can be used during the computation of S-unit groups, when a subgroup of finite index is already known [2, 4, 8]. Usually, the saturation is performed for small primes e dividing this index. We show here the efficiency of our methods to handle much larger values of prime-power e's.

Assume that we have access to $E = \{y_1, \ldots, y_s\}$ a generating set of H, a subgroup of a multiplicative group G. To fix ideas, G is the group $\mathcal{O}_{K,S}^{\times}$ of S-units of a number field K for some set S of prime ideals, and H a full-rank subgroup [4, 5]. Additionally we assume elements of E are given in *compact representation*, i.e., their factorisation on a given multiplicative basis $U = \{u_1, \ldots, u_r\}$ is known as:

$$\forall i \in [\![1, s]\!], \exists (e_{i,j})_{1 \leq j \leq r} \in \mathbb{Z}^r \;\Big|\; y_i = \prod_{j=1}^{r} u_j^{e_{i,j}}.$$

The overall saturation can be summarized as follows:

(1) detect elements of $H \cap (K^*)^e$, as we will see, this is actually the most costly part;
(2) compute the corresponding e-th roots, this is exactly the subject of this paper;
(3) compute a multiplicative basis of a subgroup of index divided by e, this part benefits from being realized only once for all possible e's dividing the index of H in G, in order to contain the size of the elements, and won't be discussed further.

A precise theoretical analysis can be found in [8].

6.1 Detecting e-th powers

Let e be a prime-power. In this section we will describe how one can efficiently detect e-th powers. At a very high level, it goes as follows:

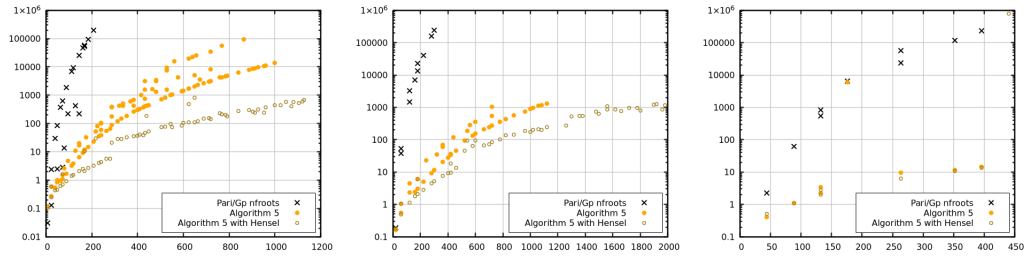
(1) select characters $\chi_{\mathfrak{Q}} : (H, \times) \to (\mathbb{Z}/\mathbb{Z}_e, +)$ such that $y \in H^e \implies \chi_{\mathfrak{Q}}(y) = 0$ for sufficiently enough primes $\mathfrak{Q}$;
(2) compute the kernel of of $\chi : y \in H \mapsto (\chi_{\mathfrak{Q}}(y))_{\mathfrak{Q}}$.

The characters selected in step (1) have to be numerous enough so that the morphism χ contains non trivial e-th powers. Theoretically, the validity of this step is controlled by the Grunwald-Wang theorem, and a practical instantiation of the problematic cases where the Grunwald-Wang theorem does not directly apply, can be found in [6, §4.2]. We shall only keep in mind that problematic cases only arise when e is a power of 2.

6.1.1 Conditions on the primes. Prime integers q below suitable $\mathfrak{Q}$ verify some conditions. The reduction map $\phi_{\mathfrak{Q}} : \mathcal{O}_K \twoheadrightarrow \mathcal{O}_K/\mathfrak{Q}$ needs to be extendable to H which is not included in $\mathcal{O}_K$ in general. Thus, q should not divide the numerator or the denominator of any y_i for $i \in [\![1, s]\!]$. We will write $\phi_{\mathfrak{Q}}$ as well for this extended map.

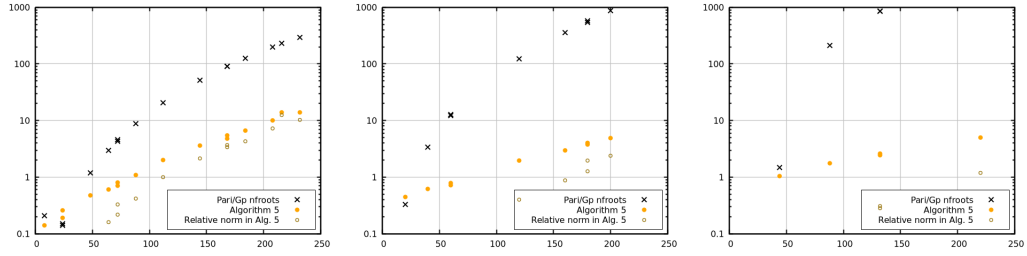
Then recall that we wish to detect non trivial e-th powers, so the residue field should contain elements which are not e-th powers. This is equivalent to $Q \equiv 1 \mod e$, where $Q = \mathrm{N}_{K/\mathbb{Q}}(\mathfrak{Q})$.

6.1.2 Definition of the characters. Once an element y has been embedded into a residue field $\mathcal{O}/\mathfrak{Q}$, one needs to detect whether $\phi_{\mathfrak{Q}}(y)$ is an e-th power or not. One can note that for any element $t \in \mathbb{F}_Q^*$, its power $t^{(Q-1)/e}$ is an e-th root of unity, and is equal to 1 if, and



(a) $p = 5$ (b) $p = 11$ (c) $p = 23$

Figure 2: Timings (s) for `nfroots` and Alg. 5 plotted against n, over fields $\mathbb{Q}(\zeta_{pq})$ with constant $[K : k] = p - 1$ and $e = q$.



(a) $p = 5$ (b) $p = 11$ (c) $p = 23$

Figure 3: Timings (s) for `nfroots` and Alg. 5 plotted against n, over fields $\mathbb{Q}(\zeta_{pq})$ with constant $e = p$ and $[K : k] = \varphi(q)$.

only if, $t \in (\mathbb{F}_Q^*)^e$. Thus one puts $\chi_{\mathfrak{Q}} : y \mapsto \log_{\zeta_e}(\phi_{\mathfrak{Q}}(y)^{(Q-1)/e})$, with ζ_e is a primitive e-th root of unity.

6.1.3 Number of characters. In order to detect non trivial powers, i.e., elements in $G^e \setminus H^e$, one only has to intersect $\ker \chi_{\mathfrak{Q}}$ for sufficiently many $\mathfrak{Q}$. If s is the cardinal of a generating family E of H, then the rank of $H/(H \cap (K^*)^e)$ is $s' \leqslant s$. If we consider the $\chi_{\mathfrak{Q}}$ to be uniformly distributed in the dual then [10, lemma 8.2] can be modified to show that $s' + r$ characters generate the dual with probability at least $1 - e^{-r}$.

6.2 Practical considerations

6.2.1 Computing suitable primes. Discarding the condition regarding the denominators of elements of E, suitable prime integers $q \mid \mathfrak{Q}$ only need to verify $e \mid \mathrm{N}_{K/\mathbb{Q}}(\mathfrak{Q}) - 1$. As we have to compute a discrete logarithm in $\mathbb{F}_Q$, it is desirable to restrict to primes of inertia degree 1. Hence, instead of drawing random primes of a given bit-length, it is best to test primality on integers $q \equiv 1 \mod e$ until sufficiently many primes of inertia degree 1 are found. Note that, contrary to the CRT case, small primes give as much information as big primes for the characters.

In the Galois case, this comes down to find completely split primes q, and Chebotarev density theorem assures us that the density of those primes in the set of all prime integers is $1/[K : \mathbb{Q}]$. Hence, we expect to find $k \geq s' + r$ suitable primes $q \equiv 1 \mod e$ of a given bitsize in $O(k[K : \mathbb{Q}])$ trials.

For cyclotomic fields $K_m = \mathbb{Q}(\zeta_m)$ the situation is even better, as a prime q completely splits in K if, and only if, $q \equiv 1 \bmod m$. Thus one needs to find k primes that directly verify the congruence $q \equiv 1 \mod \mathrm{lcm}(e, m)$, which can be done at a given bitsize in $O(k)$ trials.

6.2.2 Computing non trivial relations. Once characters have been selected as a set of prime ideals $S = \{\mathfrak{Q}_1, \ldots, \mathfrak{Q}_k\}$ above suitable prime integers q, one needs to compute the values $\chi_{\mathfrak{Q}}(y_i)$ for all $\mathfrak{Q}$ and $i \in [\![1, s]\!]$, then identify the kernel of χ yielding $H \cap (K^*)^e$.

Recall that each element y_i of the generating set E is known through its decomposition in the multiplicative basis U. Since a character is a morphism, the image of $\chi_{\mathfrak{Q}}$ is determined by the collection of elements $\chi_q(u_j) \in \mathbb{Z}/e\mathbb{Z}$. As the u_j are expected to be small, it is generally more efficient to compute first all $\chi_q(u_j)$ and reconstruct each $\chi_{\mathfrak{Q}}(y_i)$ as $\sum_{j=1}^{r} e_{i,j}\chi_{\mathfrak{Q}}(u_j)$.

The image $\chi(H)$ is then generated by the rows of the matrix

$$M = \left(\chi_{\mathfrak{Q}}(y_i)\right)_{\substack{i \in [\![1,s]\!] \\ \mathfrak{Q} \in S}} \in \mathcal{M}_{s,k}\left(\mathbb{Z}/e\mathbb{Z}\right).$$

Therefore, computing the left kernel of M in $\mathbb{Z}/e\mathbb{Z}$ yields vectors $(\alpha_1, \ldots, \alpha_t) \in [\![0, e-1]\!]^t$ (after lifting to the integers) such that $\prod_{i=1}^{s} y_i^{\alpha_i} \in (K^*)^e$.

Note that when e is a prime-power, we can use Howell's normal form [21]. This form might give redundant solutions, but those will be sorted out in the final reconstruction phase.

6.2.3 Complexity analysis. Actually, detecting of e-th powers can be problematically long if the sizes of the different finite fields are too large. This mecanically happens when e is large. The bottleneck is the computation of discrete logarithms in subgroups of order $e - 1$, and we need to do it sk times. The cost is then $O(sk\sqrt{e})$ using generic techniques. It can be mitigated by using index calculus

methods, which can also benefit from a precomputation phase when computing many discrete logarithms.

6.3 Experiments

In this section, we report the impact of the algorithms presented here on the running time in saturation processes while computing S-units in the manner of [4] [2]. In particular, note that elements are given in compact representation $y = \prod_i u_i^{e_i}$ and that one does not require to compute the product. This helps for example mitigating the drawback of some costly operations such as the relative norm in Algorithm 5. Timings can be found in Figure 4. We separated "good" cases where $e \nmid m$ and bad cases where $\mathbb{Q}(\zeta_e) \subseteq K$. We selected e to be the largest prime factor with of h_m^- and $\gcd(m, h_m^-)$ respectively. Note that we restricted our experiments to e with bit-size smaller than 40 in the first case, and that we used different machines which can explain differences in running times.

Remark that we gain a significant advantage using Algorithm 3 in good cases, especially when n is large. In bad cases, the performances of Algorithm 5 has less impact. However it outperforms Pari/Gp nfroots when n is large as well.

Note that much data is missing for Pari/Gp nfroots, which is due to its asymptotical limitations. To draw a better (more precise) picture of the situation, we gathered selected data in Table 1 for large exponents e.

Remark that the running time of nfroots seems to be greatly influenced by the size of e, whereas Algorithm 3 is relatively stable with respect to this parameter.

REFERENCES

[1] Leonard Adleman, Kenneth Manders, and Gary L. Miller. 1977. On Taking Roots in Finite Fields. In *18th Annual Symposium on Foundations of Computer Science*. IEEE Computer Society, 175–178.

[2] Jens Bauch, Daniel J. Bernstein, Henry de Valence, Tanja Lange, and Christine van Vredendaal. 2017. Short Generators Without Quantum Computers: The Case of Multiquadratics. In *Advances in Cryptology – EUROCRYPT 2017*, Jean-Sébastien Coron and Jesper Buus Nielsen (Eds.). Springer International Publishing, Cham, 27–59.

[3] Karim Belabas. 2004. A relative van Hoeij algorithm over number fields. *J. Symb. Comput.* 37 (05 2004), 641–668. https://doi.org/10.1016/j.jsc.2003.09.003

[4] Olivier Bernard, Andrea Lesavourey, Tuong-Huy Nguyen, and Adeline Roux-Langlois. 2022. Log-$\mathcal{S}$-unit lattices using Explicit Stickelberger Generators to solve Approx Ideal-SVP. In *Advances in Cryptology – ASIACRYPT 2022 (LNCS, Vol. 13793)*. Springer, 677–708.

[5] Olivier Bernard and Adeline Roux-Langlois. 2020. Twisted-PHS: Using the Product Formula to Solve Approx-SVP in Ideal Lattices. In *Advances in Cryptology – ASIACRYPT 2020*, Shiho Moriai and Huaxiong Wang (Eds.). Springer, 349–380.

[6] Jean-François Biasse, Muhammed Rashad Erukulangara, Claus Fieker, Tommy Hofmann, and William Youmans. 2022. Mildly Short Vectors in Ideals of Cyclotomic Fields Without Quantum Computers. *Mathematical Cryptology* 2, 1 (Nov. 2022), 84—-107. https://journals.flvc.org/mathcryptology/article/view/132573

[7] Jean-François Biasse, Claus Fieker, Tommy Hofmann, and Aurel Page. 2020. Norm relations and computational problems in number fields.

[8] Jean-François Biasse, Claus Fieker, Tommy Hofmann, and Aurel Page. 2020. Norm relations and computational problems in number fields. arXiv:2002.12332 [math.NT]

[9] Richard P. Brent. 1975. Multiple-precision zero-finding methods and the complexity of elementary function evaluation. *Analytic Computational Complexity* (1975), 151–176.

[10] J. P. Buhler, H. W. Lenstra, and Carl Pomerance. 1993. Factoring integers with the number field sieve. In *The development of the number field sieve*, Arjen K. Lenstra and Hendrik W. Lenstra (Eds.). Springer, 50–94.

[11] Henri Cohen. 1993. *A Course in Computational Algebraic Number Theory*. Springer.

[12] H. Cohen. 2012. *Advanced Topics in Computational Number Theory*. Springer.

[13] Jean-Marc Couveignes. 1997. Computing A Square Root For The Number Field Sieve. 1554 (06 1997). https://doi.org/10.1007/BFb0091540

[14] R. Cramer, L. Ducas, and B. Wesolowski. 2017. Short Stickelberger Class Relations and Application to Ideal-SVP. In *EUROCRYPT*.

[15] Claus Fieker and Carsten Friedrichs. 2000. On Reconstruction of Algebraic Numbers. In *Algorithmic Number Theory*, Wieb Bosma (Ed.). Springer, 285–296.

[16] Nicolas Gama and Phong Q. Nguyen. 2008. Predicting Lattice Reduction. In *EUROCRYPT (LNCS, Vol. 4965)*. Springer, 31–51.

[17] Serge Lang. 2012. *Algebra*. Vol. 211. Springer Science & Business Media.

[18] J. Neukirch. 1999. Algebraic Number Theory.

[19] Alice Pellet-Mary, Guillaume Hanrot, and Damien Stehlé. 2019. Approx-SVP in Ideal Lattices with Pre-processing. In *Advances in Cryptology – EUROCRYPT 2019*, Yuval Ishai and Vincent Rijmen (Eds.). Springer, 685–716.

[20] Victor Shoup. 1993. Factoring polynomials over finite fields: Asymptotic complexity vs. reality. In *Proceedings of the IMACS Symposium*. 124–129.

[21] Arne Storjohann. 2013. *Algorithms for Matrix Canonical Forms*. Ph.D. Dissertation. Swiss Federal Institute of Technology, Zurich.

[22] The PARI Group 2022. *PARI/GP version* `2.13.4`. The PARI Group, Univ. Bordeaux. available from http://pari.math.u-bordeaux.fr/.

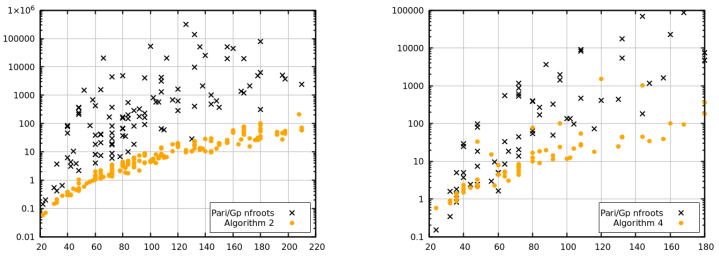
[23] The Sage Developers. 2023. *SageMath, the Sage Mathematics Software System (Version x.y.z)*. `https://www.sagemath.org`.

[24] Emmanuel Thomé. 2012. Square Root Algorithms for the Number Field Sieve. In *Arithmetic of Finite Fields*, Ferruh Özbudak and Francisco Rodríguez-Henríquez (Eds.). Springer, 208–224.

[25] Barry M. Trager. 1976. Algebraic Factoring and Rational Function Integration. In *Proceedings of the Third ACM Symposium on Symbolic and Algebraic Computation (SYMSAC '76)*. ACM, 219–226.

[26] Joachim von zur Gathen and Jürgen Gerhard. 2013. *Modern Computer Algebra* (3 ed.). Cambridge University Press.

[2] Code publicly available at https://github.com/anonymousroots/roots-crt



(a) $e \nmid m$ (b) $e \mid m$

Figure 4: Timings (s) for `nfroots` and Alg. 4 plotted against the dimension for saturation process.

Table 1: Timings (s) for e-th roots within computation of S-units for selected cyclotomic fields

Conductor m	53	61	67	107	109	151	199
Dim. $\varphi(m)$	52	60	66	106	108	150	198
e ($\log_2(e)$)	4889 (12)	1861 (10)	12739 (13)	2886593 (21)	9431866153 (33)	312885301(28)	25645093 (24)
Algorithm 3	0.6	1.1	1.3	6.5	7.2	20.4	54.9
nfroots	1471.9	421.7	20921.2	n/a	n/a	n/a	n/a