



HAL
open science

Computing e -th roots in number fields

Olivier Bernard, Pierre-Alain Fouque, Andrea Lesavourey

► **To cite this version:**

Olivier Bernard, Pierre-Alain Fouque, Andrea Lesavourey. Computing e -th roots in number fields. ALENEX 2024 - SIAM Symposium on Algorithm Engineering and Experiments, Jan 2024, Alexandria, United States. pp.207-219, 10.1137/1.9781611977929.16 . hal-04832783

HAL Id: hal-04832783

<https://hal.science/hal-04832783v1>

Submitted on 12 Dec 2024


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.




Distributed under a Creative Commons Attribution 4.0 International License

Computing e -th roots in number fields

Olivier Bernard 
olivier.bernard@normalesup.org
Zama
Paris, France

Pierre-Alain Fouque 
pierre-alain.fouque@irisa.fr
Univ Rennes, IRISA
Rennes, France

Andrea Lesavourey 
andrea.lesavourey@irisa.fr
Univ Rennes, CNRS, IRISA
Rennes, France

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.

KEYWORDS

Roots, Number fields, CRT, p -adic lifting, Couveignes

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 [9, 12] (NFS). This problem also intervenes during saturation processes while computing the class group or S -units of a number field [7]. Recently, such computations were found to be important to study the Ideal-SVP problem used in Lattice-based Cryptography [4].

Generally speaking, elements are given in a "factored" 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 ones to each other. Indeed, for the NFS, the length r of the product 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.

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 power. We aim at designing a workable method for *large* exponents e and dimension $[K : \mathbb{Q}]$ for all cases.

- When K and e are such that there exist infinitely many prime integers p such that $\forall \mathfrak{p} \mid p, p^{f_{\mathfrak{p}}} \not\equiv 1 \pmod{e}$ (condition $(*)$), we reconstruct x from $(x \bmod p_1, \dots, x \bmod p_k)$ using the Chinese Remainder Theorem (CRT), where each $(x \bmod p_i)$ is itself computed through a CRT procedure on prime ideals of K . We call this generalisation of Thomé's square-roots algorithm [24] the Double-CRT (Algorithms 2 and 3).
- Generically, we can use p -adic lifting when there is an inert prime, or the 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 §3 (see Algorithm 1). While both methods work for any e , inert primes do not always exist and p -adic reconstruction scales poorly with the dimension of K . However, sometimes they can be very useful and we indeed 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 [12] to relative extensions of number fields K/L , provided $[K : L]$ is coprime to e and sufficiently many prime integers p verify that each prime ideal \mathfrak{p} of \mathcal{O}_L above p is inert in K . We then turn this strategy into a recursive procedure, calling the previous algorithms until the smallest possible subfield is reached (Algorithms 4 and 5).

This leads to the following global strategy to compute an e -th root of an element $y \in K$, which we use in our implementation:

- (1) if condition $(*)$ is satisfied, use the Double-CRT (Algorithm 3);
- (2) else, if inert primes exist in K , use a p -adic lifting (see §3.1);
- (3) otherwise, try to use the relative Couveignes' strategy, going back to step (2) for computing an e -th root of $N_{K/L}(y)$ in L ;
- (4) finally, resort to the p -adic reconstruction given in §3.2.

Experimental results

We ran experiments to evaluate the performances of our algorithms when compared to standard methods and implementations, especially PARI/GP nroots. 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 SAGE-MATH [20] 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 an optimised C implementations. All our codes are freely available at <https://github.com/ob3rnard/eth-roots>.

Over "good" cases, our CRT generalisation algorithm is clearly more efficient than PARI/GP nroots, see e.g. Figure 1, and the gap explodes when the exponent e increases. Our algorithm scale well

The authors would like to thank Dr. Razvan Barulescu and Dr. Aurel Page for suggesting using Schirokauer maps to detect huge e -th powers. The authors would also like to thank the anonymous reviewers for their interesting and useful comments. Most of this work was done while Olivier Bernard was employed by Thales, Gennevilliers, France. Andrea Lesavourey is funded by the Direction Générale de l'Armement (Pôle de Recherche CYBER), with the support of Région Bretagne.

and we can use it without any problems for e of 94-bits prime (see Table 1), which would be completely unrealistic for PARI/GP.

Over “bad” cases, experiments show that our generalization of Couveignes’ algorithm is also more efficient than PARI/GP nroots [19], see Figures 2 and 3. Our algorithm is always faster, and the gap with nroots becomes larger when e and n increase.

Finally, we tested our algorithms in a real-life situation, namely saturating full-rank multiplicative sets of S -units arising from Stickelberger’s theorem using the code of [4], see §6, Table 1 and Figure 4. Again, our implementation of our algorithms is more efficient than PARI/GP nroots for all ranges of exponents or dimensions.

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. Let us denote by $M(s)$ the complexity of multiplying two s -word integers, and by $M(d, s)$ the complexity of multiplying two polynomials of degree d whose coefficients are taken modulo an s -word integer, i.e., using fast arithmetic (see e.g., [15, Cor. 11.8 and 11.10]),

$$M(d, s) \leq O(d \log d \log \log d \cdot s \log^2 s \log \log s).$$

2.1 Bounds on root coefficients

In all the methods of this paper, we need a bound on the coefficients of the sought 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 precisely, let $\Omega = (b_{ij}) \in \mathcal{M}(\mathbb{Q})$ be s.t. $\omega_i = \sum_j b_{ij} \alpha^{j-1}$, and $V_\alpha = (\sigma_j(\alpha)^{i-1})$ 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 $\Sigma(x) = C(x) \cdot (\Omega V_\alpha)$ 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 , and let $C_\infty = \|\Omega^{-1} V_\alpha^{-1}\|_\infty$. Then we have*

$$\|C(x)\|_\infty \leq \|\Sigma(x)\|_\infty \cdot C_\infty.$$

PROOF. This is a direct adaptation of [3, Lem. 3.3] (see also [13, Lem. 6]), noting that the given definition of the infinity norm for matrices is equivalent to $\|A\|_\infty = \sup_{y \neq 0} (\|yA\|_\infty / \|y\|_\infty)$. \square

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 *factored* form, i.e., there exist $(u_i)_{1 \leq i \leq r} \in K^*$ and $(a_i)_{1 \leq i \leq r} \in \llbracket 0, e-1 \rrbracket^r$ 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 factored form as above, and let x be such that $y = x^e$. Then*

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

PROOF. For any embedding σ , we have

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

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

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 $p > 2$ with $q = p^d$. For the local-global methods of this paper, an important tool consists in computing an e -th root in \mathbb{F}_q , where e is given modulo $q-1$.

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

In very rare cases, we will have no choice but to consider $q \equiv 1 \pmod{e^2}$, thus we resort to the Adleman-Manders-Miller algorithm [1, Th. IV], which is an adaptation of Tonelli-Shanks algorithm to the case $e > 2$, and has complexity $O((e \log q + \log^2 q) \cdot M(d, \log p))$ in the worst case. Alternatively, we can use generic factorisation methods such as the Cantor-Zassenhaus algorithm [10, Alg. 3.4.6], whose complexity in $O(e^2 \log^{1+e} e \log(qe) \cdot M(d, \log p))$ [22] might be competitive for small $e < \log q$.

For simplicity’s sake, we will denote by $R(e, q)$ the complexity of computing an e -th root in \mathbb{F}_q in all cases.

3 GENERIC LOCAL-GLOBAL METHODS

In this section, we present two local-global methods that apply unconditionally w.r.t. e , namely p -adic lifting (e.g., [24, §1.1]), and 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 \mathfrak{p} in K . It relies on the fact that $K_{\mathfrak{p}}$, the p -adic completion of K at \mathfrak{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 $K_{\mathfrak{p}}$ is injective. Thus, a sufficiently good approximation in $K_{\mathfrak{p}}$ 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 $K_{\mathfrak{p}}$.

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} :

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

Let $B > 0$ be an upper bound on the coefficients of x , then this iteration is performed $\kappa = \lceil \log_2 \max\{1, \log_p 2B\} \rceil$ 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 $y^{1/2} = y \cdot y^{-1/2}$ [8, §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

$$x_{i+1} = x_i - \frac{1}{e} x_i (y^{e-1} x_i^e - 1). \quad (1)$$

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 factored 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. $\|C(x)\|_\infty \leq B$ ▷ Using Lemmas 1 and 2
 - 2: $\kappa \leftarrow \lceil \log_2 \max\{1, \log_p 2B\} \rceil$
 - 3: $a \leftarrow (\prod_i u_i^{a_i})^{e-1} \pmod{p^{2^\kappa}}$ ▷ Reduce u_i 's first
 - 4: $x_0 \leftarrow (a \pmod{p})^{1/e}$ in $\mathbb{F}_p^n \simeq \mathcal{O}_{/p\mathcal{O}}$ ▷ Using §2.2
 - 5: **for** $0 \leq i < \kappa$ **do**
 - 6: $x_{i+1} \leftarrow x_i - \frac{1}{e} x_i (ax_i^e - 1)$ ▷ Work in \mathcal{O} modulo $p^{2^{i+1}}$
 - 7: **end for**
 - 8: $x \leftarrow a \cdot x_\kappa \pmod{p^{2^\kappa}}$
 - 9: **return** x with coefficients mapped in $[-B, B]$
-

Proposition 1. *Algorithm 1 is correct, and runs in time at most $O(\log e \cdot (r + \log s) \cdot M(n, s) + R(e, p^n))$, where s is the total input size, i.e., $s = O(\sum_i \log \|\Sigma(u_i)\|_\infty)$.*

PROOF. Let x be a root of $g(z) = y^{e-1} z^e - 1$. At any stage $i \geq 0$, 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

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

As for the complexity, note that, by Lemma 2, $\log B = O(s)$, thus $\kappa = O(\log s)$. Computing the degree n polynomial $(a \pmod{p^{2^\kappa}})$ costs at most $O(r \log e \cdot M(n, s))$ using that $a_i < e$, and all polynomials $(a \pmod{p^{2^i}})$ for the loop can be iteratively deduced in negligible $O(nM(s) \log s)$ time. Likewise, computing all $(1/e \pmod{p^{2^i}})$ costs $O(s \log e \log s)$. Computing $(a \pmod{p})^{1/e}$ in \mathbb{F}_p^n costs $R(e, p^n)$ by §2.2. Finally, using these values, each of the $O(\log s)$ Newton iterations at precision $k = 2^i$ costs at most $O(\log e \cdot M(n, k \log p))$ for a total of $O(\log e \log s \cdot M(n, s))$. \square

3.2 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 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 $\mathcal{O}_{K/\mathfrak{p}} \simeq \mathbb{F}_{p^{f(\mathfrak{p}|p)}}$, Equation (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 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 [14], we start from the smallest $a = 2^\kappa$ such that

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

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 verifies

$$\exists_\infty \text{ primes } q \text{ s.t. } \forall q \mid q, q^{f_q} \not\equiv 1 \pmod{e}. \quad (2)$$

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: An unramified prime q in K verifying (2), and $y \in (K^*)^e$ in factored form $y = \prod_{i \leq r} u_i^{a_i}$, where the u_i 's are given mod q .

Ensure: $x \equiv y^{1/e} \pmod{q}$.

- 1: $S \leftarrow \{q_1, \dots, q_g\} := \{q; q \mid q\mathcal{O}\}$ ▷ Using Cantor-Zassenhaus
- 2: Compute all $(u_i \pmod{q_j})_{i,j}$ ▷ Use a product tree
- 3: **for** $q \in S$ **do**
- 4: $y_q \leftarrow \prod_i (u_i \pmod{q})^{a_i}$
- 5: $x_q \leftarrow y_q^{1/e} \pmod{q}$ ▷ Using §2.2
- 6: **end for**
- 7: **return** $\text{CRT}_K(\{x_q\}_{q \in S}, \{q\}_{q \in S})$

Algorithm 3 Double CRT for e -th root

Require: $y \in (K^*)^e$ in factored form $y = \prod_{i \leq r} u_i^{a_i}$.

Ensure: $x \in K^*$ such that $y = x^e$.

- 1: Compute B s.t. $\|C(x)\|_\infty \leq B$ ▷ Using Lemmas 1 and 2
- 2: Choose primes q_1, \dots, q_k verifying (2) s.t. $\prod_j q_j \geq 2B$.
- 3: Compute all $(u_i \pmod{q_j})_{i,j}$ ▷ Use a product tree
- 4: **for** $1 \leq j \leq k$ **do**
- 5: $x_j \leftarrow (y \pmod{q_j})^{1/e}$ ▷ Using Algorithm 2
- 6: **end for**
- 7: $x \leftarrow \text{CRT}_{\mathbb{Z}}(\{x_j\}_j, \{q_j\}_j)$ ▷ Coefficient by coefficient
- 8: **return** x with coefficients mapped in $[-B, B]$

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

PROOF. Since q is unramified in K , i.e., $(q) = \prod_{q \in S} q$, one has

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

For all $q \in S$, the condition $q^f \not\equiv 1 \pmod{e}$ implies $\mathbb{F}_{q^{f_q}}^* = (\mathbb{F}_{q^{f_q}}^*)^e$, so that any element of $O_{K/q}$ has a *unique* e -th root. Consequently, step ?? of Algorithm 2 is properly defined and $x \equiv x_q \pmod{q}$. Thus the output of step ?? is indeed congruent to x modulo q .

As for the complexity, the first step, factorising a degree n polynomial over \mathbb{F}_q , can be done in $O(n^2 \log^{1+\epsilon} n \log qn \cdot M(\log q))$ using Cantor-Zassenhaus. Computing the product tree for the q_j 's and reducing $\{u_i\}_{1 \leq i \leq r}$ modulo all q_j 's cost $(r+1) \cdot O(\log q \cdot M(n, \log q))$ [15, Lem. 10.4 and Th. 10.15]. For a given $q \mid q$, computing y_q costs at most $O(r f_q \log q \cdot M(f_q, \log q))$ (mapping each a_i modulo $q^{f_q} - 1$), and the e -th root in $\mathbb{F}_{q^{f_q}}$ costs $O(f_q \log q \cdot M(f_q, \log q))$ since by hypothesis $q^{f_q} \not\equiv 1 \pmod{e}$. Since all f_q sum to n , the whole loop costs at most $O(rn \log q \cdot M(\max f_q, \log q))$. 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 an extra $O(r \log e \cdot n \log q)$ term in the worst scenario. The last CRT step can be done in $O(\log n \cdot M(n, \log q))$ [15, Cor. 10.23], which is negligible. \square

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 primes with small maximum inertia degrees. If there are sufficiently many totally split primes, e.g., in cyclotomic fields, the complexity even drops to $O(rn \log^{2+\epsilon} q + r \log q \log e)$.

We now explain how to compute x *via* computations modulo primes q_1, \dots, q_k that all verify (2). This is done in Algorithm 3.

Proposition 3. *Algorithm 3 is correct, and runs in time at most $O(rns \cdot (M(n, \kappa) + \log^{2+\epsilon} s + \log e))$, where $s = O(\sum_i \log \|\Sigma(u_i)\|_\infty)$ is the total input size.*

PROOF. Step ?? computes the solution modulo the ideal generated by q_j , for each $j \in \llbracket 1, k \rrbracket$. Thus the CRT in step ?? computes $x \pmod{Q}$, where $Q = \prod_j q_j$, and $Q > 2B$ ensures that there is a unique element $z \equiv x \pmod{Q}$ with coefficients in $[-B, B]$.

Let $\kappa = \max_j \log q_j$. To fix ideas, κ is taken as to fit a machine word and all q_j 's are chosen evenly, so $k \cdot \kappa \approx \sum \log q_j = O(s)$ and $\kappa = O(1)$. Computing the product tree for the q_j 's and reducing one coefficient of the u_i 's modulo each q_j costs $O(M(\log Q) \cdot \log k)$ by [15, Th. 10.24], hence in total $O(rn \cdot M(s) \log s)$. Using Proposition 2, the loop has a total cost of $O(rn \log Q \cdot (M(n, \kappa) + \log e))$. By [15, Th. 10.25], the final CRT has a negligible $O(n \cdot M(s) \log s)$ cost. \square

Remark 2. Likewise, using Remark 1, the overall complexity drops to $O(rnM(s) \log s + rs \log e)$ when many totally split primes exist.

4.2 Bad cases: existence of cyclotomic subfield

In this section, we give the following result, which characterises number fields which are “bad” fields for e . This result can also be found in [17, §C, Pr. 2.1], and we give the proof for completeness.

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

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

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

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

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

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 $p \equiv 1 \pmod{e}$, 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 $P(K/\mathbb{Q}) \subseteq P(\mathbb{Q}(\zeta_e)/\mathbb{Q})$ and Lemma 3 gives (ii). \square

In all generality, the class of fields for which Algorithm 3 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 `nroots`. 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 factored forms. The results obtained for $\log B = 100$ are displayed in Figure 1.

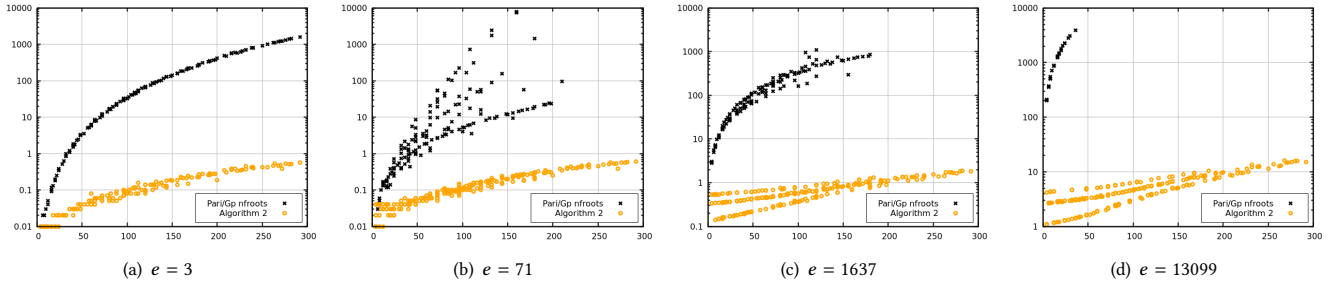


Figure 1: Timings (s) for nroots and Algorithm 3 plotted against the degree, for various prime e over cyclotomic fields.

Conductor m	113	256	137	149	169	243	249	167	173	179	181	235	197	199
Degree $\varphi(m)$	112	128	136	148	156	162	164	166	172	178	180	184	196	198
$\log_2(e)$, with $e \mid h_m^-$	43.4	44.7	45.4	68.8	47.5	39.7	53.9	82.1	66.0	93.5	62.3	57.1	92.9	41.5
$\log_2(\ x\ _\infty)$, $x = y^{1/e}$	215.0	263.9	311.9	346.9	381.9	395.6	393.4	469.1	320.7	410.0	461.5	488.4	546.3	492.7
Double CRT (Alg. 3)	2.5	3.4	5.7	9.3	8.7	8.6	24.7	12.3	9.1	14.0	14.6	24.5	17.3	16.2
Schirokauer maps	176	15	748	1035	100	20	822	1526	2408	4913	286	998	5426	1864

Table 1: Timings (s) for e -th roots within the saturation of Stickelberger S -units [4] for selected cyclotomic fields

Remark 3. For small exponents, i.e., such that $3e \leq [K : \mathbb{Q}]$, the function nroots from PARI/GP uses Trager’s method [25]; in these experiments, this is the case only when $e = 3$. Otherwise it follows the ideas developed in [3, 13].

From Figure 1, we see that our implementation of Algorithm 3 using SAGEMATH is, in all cases, much more efficient than nroots when the dimension increases. Further, the gap increases with the exponent. Remark, e.g., in Figure 1(b), that Algorithm 3 is more stable than PARI/GP nroots.

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, namely the computation of S -units in cyclotomic fields, by saturating a full-rank multiplicative S -units subset arising from Stickelberger’s theorem, as suggested in [4]. Concretely, our biggest example here is for $e = 14458667392334948286764635121$ in $\mathbb{Q}(\zeta_{179})$, a 94-bits prime dividing h_{179}^- , for which an e -th root in dimension $n = 178$ is computed in only 14 seconds.

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 $\mathfrak{p} \mid p$ such that $p^{f_{\mathfrak{p}}} \equiv 1 \pmod{e}$.

5.1 A relative Couveignes’ method

Couveignes’ method for square-root computation [12, 24] allows identifying together the roots modulo a set of inert primes p_1, \dots, p_k , assuming that the degree of the number field $[K : \mathbb{Q}]$ is odd. In the following, we show how to generalise this method to $e \geq 3$.

A key ingredient of Couveignes’ method is the fact that the norm maps $N_K : K \rightarrow \mathbb{Q}$ and $N_{\mathbb{F}_{p^n}} : \mathcal{O}_{K/(p)} \simeq \mathbb{F}_{p^n} \rightarrow \mathbb{F}_p$ are coherent

when p is an inert prime integer, meaning that for any $x \in K$ we have $N_K(x) \pmod{p} = N_{\mathbb{F}_{p^n}}(x \pmod{p})$. We will use a generalisation of this property to relative extensions.

Lemma 4. Consider K/L a number field extension, \mathfrak{p} an inert prime ideal of \mathcal{O}_L and \mathfrak{P} the prime ideal of \mathcal{O}_K above \mathfrak{p} . Then the following diagram is commutative.

$$\begin{array}{ccc}
 \mathcal{O}_K & \twoheadrightarrow & \mathcal{O}_{K/\mathfrak{P}} \\
 \downarrow N_{K/L} & \curvearrowright & \downarrow N \\
 \mathcal{O}_L & \twoheadrightarrow & \mathcal{O}_{L/\mathfrak{p}}
 \end{array}$$

PROOF. Elements of \mathcal{O}_K can be seen as polynomials with coefficients in L with degree less than $[K : L]$, as $\mathcal{O}_K \cong \mathcal{O}_L[T]/(P_{K/L}(T))$ for some irreducible polynomial $P_{K/L} \in L[T]$. Similarly, elements of $\mathbb{F}_{\mathfrak{P}} := \mathcal{O}_{K/\mathfrak{P}}$ are class of polynomials in $\mathbb{F}_{\mathfrak{p}} := \mathcal{O}_{L/\mathfrak{p}}$. Since $\mathfrak{p} \mid \mathfrak{P}$ is inert, one can define $\mathbb{F}_{\mathfrak{P}}/\mathbb{F}_{\mathfrak{p}}$ using the same polynomial $P_{K/L}$, i.e., $\mathbb{F}_{\mathfrak{P}} \cong \mathbb{F}_{\mathfrak{p}}[T]/(P_{K/L}(T))$. Consequently, if $\alpha \in \mathcal{O}_K$ is written as $\sum_{i=0}^{[K:L]-1} \alpha_i T^i$, with $\alpha_i \in \mathcal{O}_L$, then its embedding into $\mathbb{F}_{\mathfrak{P}}$ is expressed as $\sum_{i=0}^{[K:L]-1} \bar{\alpha}_i T^i$ where $\bar{\cdot} : \mathcal{O}_L \rightarrow \mathbb{F}_{\mathfrak{p}}$. Now, recall that the relative norm of an element α in a field extension E/F is the determinant of the multiplication map $[\alpha]$, seen as a F -linear map of E . Since both extensions K/L and $\mathbb{F}_{\mathfrak{P}}/\mathbb{F}_{\mathfrak{p}}$ are given by the same defining polynomial, one can consider the action $[\alpha]$ and $[\bar{\alpha}]$ over the “same” basis, i.e., $[\alpha]$ and $\bar{\cdot}$ commute. This transfers to the determinant, so that $\det[\alpha] \pmod{\mathfrak{p}} = \det[\alpha \pmod{\mathfrak{P}}]$. \square

Another key ingredient of Couveignes’ method for $e = 2$ is that, as soon as the degree of the field is odd, $N(\pm x) = \pm N(x)$. 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 $\zeta_e \in F$. Assume additionally that $\gcd([E : F], e) = 1$. Then,

for any $y \in (E^*)^e$, the norm map $N_{E/F}$ induces a bijection between the zeroes $Z_E(X^e - y)$ in E and $Z_F(X^e - N_{E/F}(y))$ in F .

PROOF. The set $Z_E(X^e - y)$ is of the form $\{\zeta_e^i x \mid i \in \llbracket 0, e-1 \rrbracket\}$ where x is a fixed e -th root of y in E . Similarly, we have

$$Z_F(X^e - N_{E/F}(y)) = \{\zeta_e^i \cdot N_{E/F}(x) \mid i \in \llbracket 0, e-1 \rrbracket\}.$$

Now, since $\zeta_e \in F$, $N_{E/F}(\zeta_e^i \cdot x) = \zeta_e^{i[E:F]} \cdot N_{E/F}(x)$. Thus, it suffices to prove the result for the e -th roots of unity, which follows from the fact that $\gcd([E:F], e) = 1$. \square

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

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

Require: A number field extension K/L of degree prime to $e \geq 3$, $y \in (K^*)^e$ in factored form, a fixed e -th root $a = N_{K/L}(y)^{1/e}$, a prime integer p s.t. each \mathfrak{p} above p in \mathcal{O}_L is inert in K/L .

Ensure: $x \equiv y^{1/e} \pmod{(p)}$ in K .

- 1: Compute $\{\mathfrak{p}_1, \dots, \mathfrak{p}_g \mid p\mathcal{O}_L = \prod_i \mathfrak{p}_i\}$
 - 2: Compute $\{\mathfrak{P}_1, \dots, \mathfrak{P}_g \mid \mathfrak{p}_i\mathcal{O}_K = \mathfrak{P}_i\}$
 - 3: **for** $1 \leq i \leq g$ **do**
 - 4: $x_i \leftarrow y^{1/e} \pmod{\mathfrak{P}_i}$ \triangleright Pick one root in the residue field
 - 5: $a_i \leftarrow a \pmod{\mathfrak{p}_i}$ \triangleright Expected relative norm mod \mathfrak{p}_i
 - 6: $z_i \leftarrow a_i/N_{K/L}(x_i)$ \triangleright e -th root of unity mod \mathfrak{p}_i
 - 7: $x_i \leftarrow x_i \cdot z_i^{[K:L]^{-1} \pmod{e}}$
 - 8: **end for**
 - 9: **return** $\text{CRT}_L(\{x_i\}_{i \in \llbracket 1, g \rrbracket}, \{\mathfrak{P}_i\}_{i \in \llbracket 1, g \rrbracket})$.
-

Lemma 6. Consider K/L a number field extension and $e \in \mathbb{N}^*$ such that $\zeta_e \in L$. Assume additionnally that $\gcd([K:L], e) = 1$. Then Algorithm 4 is correct and runs in polynomial time.

PROOF. The output is correct if for each $i \in \llbracket 1, g \rrbracket$, x_i is the embedding in $\mathbb{F}_{\mathfrak{P}_i} := \mathcal{O}_{K/\mathfrak{P}_i}$ of a fixed e -th root of y in K , denoted by x . This is ensured by the combination of Lemmas 4 and 5, which state that in each residue field $\mathbb{F}_{\mathfrak{P}_i}$ there is exactly one element of $Z_{\mathbb{F}_{\mathfrak{P}_i}}(X^e - x_i)$ whose relative norm over $\mathcal{O}_{L/\mathfrak{p}_i}$ is a_i . \square

Fixing a root in L of the relative norm of y allows us to make a consistent choice of residues in L modulo each \mathfrak{P} from their relative norms modulo \mathfrak{p} in L . As shown by Lemma 6, this allows us to compute a root $x \pmod{p} \in K$ for a given prime integer *without* the enumeration process that would otherwise be needed to guess the correct combination of residues in the CRT. The same idea applies across several prime integers, see Algorithm 5.

Theorem 2. Consider K/L a number field extension and $e \geq 3$ such that $\zeta_e \in L$. Assume also that $\gcd([K:L], e) = 1$. Then, Algorithm 5 is correct and runs in polynomial time in $[K:\mathbb{Q}]$ and e .

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 $L \hookrightarrow K$ ensuring the existence of infinitely many prime integers with the right splitting condition mentioned. If K/L

Algorithm 5 Relative Couveignes' method for e -th root

Require: A number field extension K/L of degree prime to $e \geq 3$, $y \in (K^*)^e$ in factored form.

Ensure: $x \in K^*$ such that $y = x^e$.

- 1: Compute B s.t. $\|C(x)\|_\infty \leq B$ \triangleright Using Lemmas 1 and 2
 - 2: $a \leftarrow N_{K/L}(y)^{1/e}$ \triangleright Using §3 or this algorithm in L
 - 3: Choose prime integers p_1, \dots, p_k s.t. each prime ideal above p_j in \mathcal{O}_L is inert in K/L and $\prod_j p_j \geq 2B$.
 - 4: **for** $1 \leq j \leq k$ **do**
 - 5: $x_j \leftarrow \text{RelativeCouvMod}_p(e, y, a, p_j)$ \triangleright Algorithm 4
 - 6: **end for**
 - 7: $x \leftarrow \text{CRT}_{\mathbb{Z}}(\{x_j\}_j, \{p_j\}_j)$ \triangleright Coefficient by coefficient
 - 8: **return** x with coefficients mapped in $[-B, B]$.
-

is Galois, one can deduce from Chebotarev density theorem that this is equivalent to K/L being cyclic [11, 18].

One of the most costly steps is the computation of $N_{K/L}(y)$, whose cost is mitigated by using a factored form for y . The overall complexity depends on the algorithm used to compute its e -th root.

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 $N_{K/L}(y)$. 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 $\mathbb{Q}(\zeta_e) \hookrightarrow K$, as we are in the “bad” case. Let L be a subfield of K such that Algorithm 5 can be applied, i.e., $\gcd([K:L], e) = 1$, $\text{Gal}(K/L)$ is cyclic and $\mathbb{Q}(\zeta_e) \hookrightarrow L$. In order to apply Algorithm 5 recursively for the computation of $N_{K/L}(y)^{1/e}$, one needs the existence of a subfield of L satisfying the same conditions. We thus get Proposition 4, which describes Galois extensions for which a recursive version of Algorithm 5 is applicable.

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

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

$$L = L_0 \hookrightarrow L_1 \hookrightarrow \dots \hookrightarrow L_{r-1} \hookrightarrow L_r = K$$

such that for all $i \in \llbracket 0, r-1 \rrbracket$ the extension L_{i+1}/L_i is cyclic and $\gcd([L_{i+1}:L_i], e) = 1$. Since $\text{Gal}(K/L) \cong \bigoplus_{i=0}^{r-1} \text{Gal}(L_{i+1}/L_i)$ and $[K:L] = \prod_{i=0}^{r-1} [L_{i+1}:L_i]$, K/L is Abelian with $\gcd([K:L], e) = 1$. Conversely, denoting by G the Galois group of K/L , if this extension is Abelian then G admits a cyclic refinement [16]. The part on the dimension is clear as well. \square

5.3 Experimental results

We report now on experimental results we obtained from our implementation of Algorithm 5 in SAGEMATH [20]. 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 $[K:L] = p$ thus $e = q$

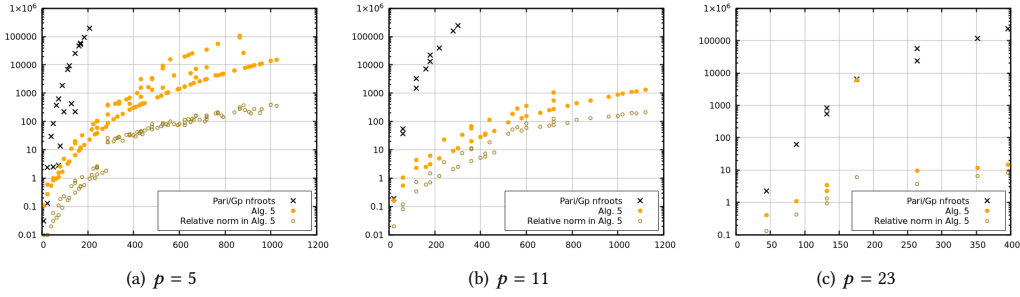


Figure 2: Timings (s) for nroots and Alg. 5 plotted against n , over fields $\mathbb{Q}(\zeta_{pq})$ with constant $[K : L] = p - 1$ and $e = q$.

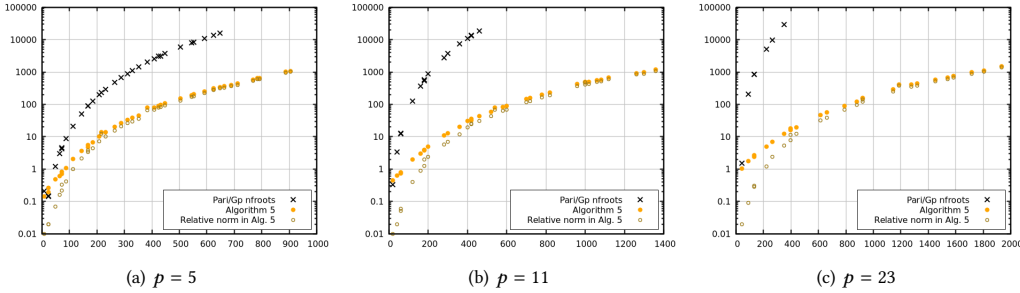


Figure 3: Timings (s) for nroots and Alg. 5 plotted against n , over fields $\mathbb{Q}(\zeta_{pq})$ with constant $e = p$ and $[K : L] = \varphi(q)$.

and $L = \mathbb{Q}(\zeta_q)$. In the second, $e = p$ so one consider $L = \mathbb{Q}(\zeta_e)$ and $[K : L] = q$. We chose to compare the performances of our implementation to the ones of Pari/Gp nroots. Results can be found in Figures 2 and 3.

One can see that Algorithm 5 is more efficient than nroots in all cases. Recall that Pari/Gp nroots uses Trager's method [25] when $3e \leq [K : \mathbb{Q}]$, which is the case in our experiments for which $e = p$. Otherwise, Pari/Gp nroots uses the ideas developed in [3, 13]. 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 $N_{K/L}(y)$ when $[K : L]$ 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 (and its implementation) even more impactful. Indeed we compute these norms as products of polynomials which might not be the most efficient in all cases. Instead, one could use a half-gcd version of resultants to be quasi-linear.

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, 7]. 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 $E = \{y_1, \dots, y_s\}$ a generating set of H , a subgroup of a multiplicative group G . To fix ideas, G is the

group K, S of S -units of a number field K for some set S of prime ideals, and H a full-rank subgroup [4, 5]. We also assume elements of E are given in *factored representation*, i.e., their factorisation on a given multiplicative basis $U = \{u_1, \dots, u_r\}$ is known as:

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

The overall saturation mechanism 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 [7].

6.1 Detecting e -th powers

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

- (1) select characters $\chi_{\mathfrak{Q}} : (H, \times) \rightarrow (\mathbb{Z}/e\mathbb{Z}, +)$ such that $y \in H^e \implies \chi_{\mathfrak{Q}}(y) = 0$ for sufficiently enough primes \mathfrak{Q} ;
- (2) compute the kernel 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 $\phi_{\mathfrak{Q}} : \mathcal{O}_K \rightarrow \mathcal{O}_K/\mathfrak{Q}$ 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 $i \in \llbracket 1, s \rrbracket$. 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 $Q \equiv 1 \pmod{e}$, where $Q = N_{K/\mathbb{Q}}(\mathfrak{Q})$.

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 $t \in \mathbb{F}_{\mathfrak{Q}}^*$, its power $t^{(Q-1)/e}$ is an e -th root of unity, and is equal to 1 if, and only if, $t \in (\mathbb{F}_{\mathfrak{Q}}^*)^e$. 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.

Actually, this infinite family of characters can be completed with easier-to-compute alternative characters, but of which we only have finitely many. For instance, when considering multiplicative subsets of S -units for some set S of prime ideals, we can include the p -adic valuations modulo e , for each $p \in S$. Another very important finite family of characters is given by *Schirokauer maps* [21], which can be viewed as an approximation of the e -adic logarithm. Namely, in our large e 's experiments, we compute

$$\lambda : y \in K \mapsto \frac{y^{\rho_e} - 1 \pmod{e^2}}{e},$$

where ρ_e is the least common multiple of the $|\mathcal{O}_K/\mathfrak{e}|$ for all prime ideals \mathfrak{e} dividing $e\mathcal{O}_K$. This yields $[K : \mathbb{Q}]$ characters – one per coefficient – modulo e [21, Pr. 3.8].

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 $s' \leq s$. If we consider the $\chi_{\mathfrak{Q}}$ to be uniformly distributed in the dual then [9, Lem. 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 $e \mid N_{K/\mathbb{Q}}(\mathfrak{Q}) - 1$. As we have to compute a discrete logarithm in $\mathbb{F}_{\mathfrak{Q}}$, 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 $q \equiv 1 \pmod{e}$ 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 $q \equiv 1 \pmod{e}$ 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, $q \equiv 1 \pmod{m}$. Thus

one needs to find k primes that directly verify the congruence $q \equiv 1 \pmod{\text{lcm}(e, m)}$, 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 $S = \{\mathfrak{Q}_1, \dots, \mathfrak{Q}_k\}$ above suitable prime integers q , one needs to compute the values $\chi_{\mathfrak{Q}}(y_i)$ for all \mathfrak{Q} and $i \in \llbracket 1, s \rrbracket$, 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 $\chi_{\mathfrak{Q}}(u_j) \in \mathbb{Z}/e\mathbb{Z}$. As the u_j are expected to be small, it is generally more efficient to compute first all $\chi_{\mathfrak{Q}}(u_j)$ 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

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

Therefore, the left kernel of M in $\mathbb{Z}/e\mathbb{Z}$ yields (after lifting to \mathbb{Z}) vectors $(\alpha_1, \dots, \alpha_t) \in \llbracket 0, e-1 \rrbracket^t$ 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 [23]. 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 mechanically happens when e is large. The bottleneck is the computation of discrete logarithms in subgroups of order e , and we need to do it rk times. The cost is then $O(rk\sqrt{e})$ using generic techniques. It can be mitigated by using index calculus methods, which can also benefit from a precomputation phase when computing many (r here) discrete logarithms.

Instead, in our practical case of interest, p -adic valuations and Schirokauer maps give just enough characters to be able to detect e -th powers. Considering that p -adic valuations are known (this is arguably often the case), the cost of detecting e -th powers amounts to computing those Schirokauer maps. In cyclotomic fields, ρ_e is at most $e^n - 1$, so the total cost of detection is $O(r \cdot n \log e \cdot M(n, 2 \log e))$. Timings for the computation of Schirokauer maps are given in Table 1, and unexpected variations directly relate to the size of ρ_e .

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]. In particular, elements are always given in factored form, and we do *never* require to compute the product. Timings can be found in Figure 4 and Table 1. We separated “good” cases where $e \nmid m$ and “bad” cases where $\mathbb{Q}(\zeta_e) \subseteq K$, and we selected e to be the largest prime factor of h_m^- (resp. s.t. $\text{gcd}(m, h_m^-)$).

We note a significant gain using Algorithm 3 in good cases, especially when n is large. In bad cases, Algorithm 5 also outperforms PARI/GP nfroots when n is large, though its impact is smaller in this case. The running time of nfroots is greatly influenced by the size of e , whereas Algorithm 3 is relatively stable with respect to this parameter. Finally, we stress that much data is missing for PARI/GP nfroots, due to its asymptotical limitations.

In order to get a more precise picture of the power of the Double-CRT algorithm, we gathered in Table 1 its performances for very large exponents e .

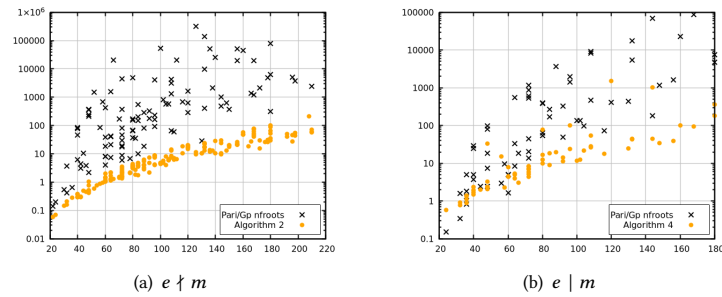


Figure 4: Timings (s) for nroots and Alg. 4 plotted against the dimension for saturation process.

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- 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*, Shihho Moriai and Huaxiong Wang (Eds.). Springer, 349–380.
- [6] Jean-François Biasse, Muhammed R. 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. arXiv:math.NT/2002.12332
- [8] Richard P. Brent. 1975. Multiple-precision zero-finding methods and the complexity of elementary function evaluation. *Analytic Computational Complexity* (1975), 151–176.
- [9] Joe P. Buhler, Hendrik 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.
- [10] Henri Cohen. 1993. *A Course in Computational Algebraic Number Theory*. Springer.
- [11] Henri Cohen. 2012. *Advanced Topics in Computational Number Theory*. Springer.
- [12] Jean-Marc Couveignes. 1997. Computing A Square Root For The Number Field Sieve. 1554 (06 1997). <https://doi.org/10.1007/BFb0091540>
- [13] Claus Fieker and Carsten Friedrichs. 2000. On Reconstruction of Algebraic Numbers. In *Algorithmic Number Theory*, Wieb Bosma (Ed.). Springer, 285–296.
- [14] Nicolas Gama and Phong Q. Nguyen. 2008. Predicting Lattice Reduction. In *EUROCRYPT (LNCS)*, Vol. 4965. Springer, 31–51.
- [15] Joachim von zur Gathen and Jürgen Gerhard. 2013. *Modern Computer Algebra* (3 ed.). Cambridge University Press.
- [16] Serge Lang. 2012. *Algebra*. Vol. 211. Springer Science & Business Media.
- [17] Pascal Molin. 2010. *Intégration numérique et calculs de fonctions L*. Ph.D. Dissertation. L’Université Bordeaux I.
- [18] Jürgen Neukirch. 1999. *Algebraic Number Theory*. Grundlehren der mathematischen Wissenschaften, Vol. 322. Springer Berlin, Heidelberg.
- [19] The PARI Group 2022. *PARI/GP version 2.13.4*. The PARI Group, Univ. Bordeaux. available from <http://pari.math.u-bordeaux.fr/>.
- [20] The Sage Developers. 2023. *SageMath, the Sage Mathematics Software System (Version x.y.z)*. <https://www.sagemath.org>.
- [21] Oliver Schirokauer. 1993. Discrete logarithms and local units. *Philosophical Transactions: Physical Sciences and Engineering* 345, 1676 (1993), 409–423.
- [22] Victor Shoup. 1993. Factoring polynomials over finite fields: Asymptotic complexity vs. reality. In *Proceedings of the IMACS Symposium*. 124–129.
- [23] Arne Storjohann. 2013. *Algorithms for Matrix Canonical Forms*. Ph.D. Dissertation. Swiss Federal Institute of Technology, Zurich.
- [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.