

# Sparse representation for cyclotomic fields

Claus Fieker<sup>1</sup>

School of Mathematics and Statistics, University of Sydney, NSW 2006, Australia  
claus@math.usyd.edu.au <http://magma.maths.usyd.edu.au/users/claus/>

**Abstract.** Currently, all major implementations of cyclotomic fields as well as number fields, are based on a dense model where elements are represented either as dense polynomials in the generator of the field or as coefficient vectors with respect to a fixed basis. While this representation allows for the asymptotically fastest arithmetic for general elements, it is unsuitable for fields of degree  $> 10^4$  that arise in certain applications such as character theory for finite groups. We propose instead a *sparse* representation for cyclotomic fields that is particularly tailored towards representation theory. We implemented our ideas in Magma and used it for fields of degree  $> 10^6$  over  $\mathbb{Q}$ .

## 1 Introduction

Currently, most implementations of number fields are based on explicitly known primitive elements and use a dense representation for the elements: A number field  $K/\mathbb{Q}$  is given by specifying a (monic) irreducible polynomial over the integers,  $f \in \mathbb{Z}[x]$  and  $K$  is constructed as the quotient ring

$$K := \mathbb{Q}[x]/f.$$

Based on this setting, elements are easily represented as polynomials of degree less than  $\deg f = [K : \mathbb{Q}]$ . Since univariate polynomials mostly have a *dense* representation on a computer, ie. are represented as a vector of length equal to the degree +1 of the polynomial, number field elements inherit this property. The primary motivation for this project was an application in the computation of characters of finite groups [9] where one needed to work in the  $n$ th cyclotomic field for

$$n = 2^6 \cdot 3 \cdot 5^2 \cdot 1201 = 5,764,800 = 7^8 - 1$$

of degree 1,536,000 over  $\mathbb{Q}$ . In particular, we need to evaluate the sum of eigenvalues of matrices over a finite field lifted into a cyclotomic field. Suppose the eigenvalues lie in a finite field  $\text{GF}(p^k)$ , then the lift of a multiplicative generator of  $\text{GF}(p^k)^\times$  is  $\zeta_{p^k-1}$ . Thus the above example originated from a matrix over  $\text{GF}(7)$  with eigenvalues in  $\text{GF}(7^8)$ .

A second motivation arose in computational class field theory ([4]). Where one starts with a finite Abelian group and computes a number field where the automorphism group is isomorphic to the given group. The field decomposes

naturally into a compositum of cyclic fields of prime power degree parallel to the decomposition of the group.

Lastly, a large number of applications start with constructions like:

Let  $K$  be a field containing a root of  $f$  and  $g$  and a  $n$ th root of unity.

One would like to preserve this information if possible, especially since usually the size of a defining polynomial for the resulting field as a simple extension of  $\mathbb{Q}$  is prohibitively large and also, since usually users expect the output to be given with respect to their input.

In this article, we will focus mainly on the cyclotomic fields. While most of the techniques apply equally to more general fields, we will discuss mainly the techniques necessary in the context of character theory.

## 2 Sparse Representation

Our sparse representation is based on Magma's sparse representation for multivariate polynomials: Let  $R := \mathbb{Q}[x_1, \dots, x_r]$  be a polynomial ring of rank  $r$  over  $\mathbb{Q}$  and  $I < R$  be a zero dimensional maximal ideal. Then  $K := R/I$  is a finite extension of  $\mathbb{Q}$  thus a number field. Elements of  $K$  can be represented as *reduced* polynomials in  $x_1, \dots, x_r$ , where the notion of *reduced* depends on the term order of  $R$ .

In what follows, the ideal  $I$  will always be generated by cyclotomic polynomials with pairwise co-prime conductors. The term order is the lexicographical order so that the generating polynomials already form a Groebner-basis for  $I$ .

Let us fix the setting: We want to perform computations in  $K := \mathbb{Q}(\zeta_n)$  where  $\zeta_n$  is a primitive  $n$ -th root of unity. Later we will specify  $\zeta_n := \exp(2\pi i/n)$  so that  $\zeta_n$  is uniquely determined as a complex number. In order to have a meaningful sparse representation, we assume that  $n$  is not a prime power:

$$n = \prod_{i=1}^r p_i^{n_i}$$

with  $r > 1$  and the  $p_i$  pairwise different. For the implementation we assume that  $\phi(p_i^{n_i}) = (p_i - 1)p_i^{n_i - 1} < 2^{30}$  ( $1 \leq i \leq r$ ).

Writing  $f_n$  for the  $n$ -th cyclotomic polynomial, we set

$$I := \langle f_{p_1^{n_1}}(x_1), \dots, f_{p_r^{n_r}}(x_r) \rangle$$

and immediately see

$$\mathbb{Q}(\zeta_n) = \mathbb{Q}(\zeta_{p_1^{n_1}}, \dots, \zeta_{p_r^{n_r}}) \cong R/I. \quad (1)$$

Elements of  $K$  are represented as *sparse* polynomials, that is as a (sorted) list of pairs  $(c_e, e)$  with  $e = (e_1, \dots, e_r)$  the exponent vector and  $c_e \in \mathbb{Q}$  the coefficient, thus  $(c_e, e)$  represents  $c_e \prod \zeta_{p_i}^{e_i}$ . Exponents  $e_i \geq \deg f_{p_i^{n_i}}$  can be reduced

using the cyclotomic polynomials (or explicit formulae). Since the polynomials are univariate in different variables, the reduction of  $e_i$  does not change any of the other exponents, the term is replaced by a sum of terms where then  $i$ th exponents are bounded by  $\phi(p_i^{n_i})$ . In general however, since reduction is a fairly expensive operation, we allow the exponents to grow to approximately  $2\phi(p_i^{n_i})$  before we reduce - unless a unique representation is required for some operation.

### 3 Arithmetic

*Basic Arithmetic.* The basic arithmetic operations like addition and multiplication, are done using representatives in  $R$ , thus they reduce to addition and multiplication of multivariate polynomials, possibly followed by a reduction. It is important to note that none of these operations require a unique representation, any representative of  $f + I$  in  $R$  can be used. In what follows however, we require the terms in  $f$  to be sorted. The complexity of the operations is straightforward to estimate: Let  $\alpha$  and  $\beta \in K$  be represented by  $f + I$  and  $g + I$  respectively, denoted by  $\alpha \cong f + I$  and  $\beta \cong g + I$ . We write  $\#f$  for the number of terms in  $f$ . By abuse of notation, we also write  $\#\alpha$  to denote the number of terms in the current representative for  $\alpha$  when it is clear what the representative looks like.

**Theorem 1.** *For  $\alpha$  and  $\beta \in K$ ,  $\alpha \cong f + I$ ,  $\beta \cong g + I$  we have*

1.  $\alpha + \beta = \gamma \cong h + I$  and  $h = f + g$  can be computed in  $O((\#f + \#g))$  operations.
2.  $\alpha\beta = \gamma \cong h + I$  and  $h = fg$  can be computed in  $O((\#f\#g)\log(\#f\#g))$  operations.

*Proof.* Using naive algorithms for the operation on the multivariate polynomials, the only part of the statement that need any explanation is the log factor in the complexity estimate for the multiplication. It comes from the sorting that is required to find identical exponent vectors. If we would use a hash-based implementation for multivariate polynomials, the log factor would essentially disappear.

It is important to realize that the complexity of operations depends mainly on the algebraic numbers involved and is essentially independent of the degree of the field  $K$  or even of the polynomials  $f_{p_i^{n_i}}$  used to represent  $K$ . In fact, the defining polynomials are only used to reduce a representation, thus as long as no “overflow” occurs during a multiplication, the defining polynomials are never used. Thus for *small* numbers, the sparse representation is much better than the classical dense one.

On the other hand, since the hidden constants are not negligible, as the numbers involved get denser the sparse representation performs worse and worse, thus for *dense* numbers the classical representation is better. Additionally, even if we assume that  $\#f, \#g \leq \deg K$  and therefore get for the complexity of the multiplication:

$$O(\#f\#g\log(\#f\#g)) = O((\deg K)^2 \log \deg K),$$

which is far worse than the  $O(\deg K \log \deg K)$  that the asymptotically fast methods in [10] that are available for the classical representation would incur.

*Minimal polynomials.* For arbitrary dense elements, the computation of minimal polynomials is a hopeless task, as their degree will be the degree of the field. However, for elements that lie in small degree subfields, it is reasonable to compute minimal polynomials using linear algebra. In a later paragraph we will indicate a different approach as well.

Let  $B := \{\prod_{i=1}^r x_i^{e_i} \mid 0 \leq e_i < \deg f_{p_i^{n_i}}\}$  be the canonical basis for  $K/\mathbb{Q}$ . By mapping

$$\Psi : K \ni \alpha = \sum_{b \in B} \alpha_b b \mapsto (\alpha_b)_{b \in B} \in \mathbb{Q}^B$$

we obtain a  $\mathbb{Q}$ -vectorspace isomorphism that we will use to compute minimal polynomials. We write  $(A|B)$  for the concatenation of two matrices  $A$  and  $B$  with the same number of rows.

**Algorithm 1 (Minimal Polynomial)** *Let  $0 \neq \alpha \in K$  be arbitrary and  $\Psi : K \rightarrow \mathbb{Q}^B$  as above.*

1. Set  $M := \Psi(1)$ ,  $\beta = 1$  and  $i := 0$ .
2. While  $i \leq d$  do
3. Repeat  $i := i + 1$ ,  $\beta := \beta\alpha$  and  $M := (M|\Psi(\beta))$  until  $i$  divides  $d$ .
4. Try to find a non-trivial element  $e = (e_0, \dots, e_i)^t$  in the nullspace of  $M$ , ie.  $Me = 0$ . If there is such an element, set  $f := 1/e_i \sum_{j=0}^i e_j x^j$  and return  $f$ , if not, go back to step 2

*Proof.* Since minimal polynomials of elements always define subfields, the degree of  $f$  must be a divisor of  $d$ . By construction, the above algorithm finds the relation between powers of  $\alpha$  of minimal degree, it is clear that  $e_i$  is non-zero in the last step. The fact that  $\Psi$  is an isomorphism of  $\mathbb{Q}$ -vectorspaces and the basis property of  $B$  guarantee that  $f$  is indeed the minimal polynomial of  $\alpha$ .

To estimate the the complexity of the above algorithm, we note that  $\deg f$  many multiplications are used,  $O(\deg f)$  many rank computations and 1 nullspace. If we use sparse matrices to represent  $M$  and assume that none of the multiplications requires a reduction, then the complexity is independent of the degree of  $K$ , it depends on the number of non-zero coefficients of  $\alpha$  (and it's powers) and the degree of  $f$ . By Theorem 1 we see that, under the assumption that  $\alpha$  is (very) sparse (to be more precise we assume that no reductions are necessary after the multiplications, or equivalently, that for all products of elements occurring, the products of the basis elements with non-zero coefficients are in  $B$ ), the dimension of the algebra problem is bounded by  $(\deg f)\#\alpha$ . The total complexity for the  $\deg f$  multiplications becomes  $O((\#\alpha)^{\deg f} \log(\#\alpha))$ . It must be stressed that this a very crude estimate only. In practise overflow is very likely to occur making it very hard to give a better, realistic estimate.

*Automorphisms.* Since  $\text{Gal}(K/\mathbb{Q}) \cong (\mathbb{Z}/n\mathbb{Z})^\times \cong \prod_{i=1}^r (\mathbb{Z}/p_i^{n_i}\mathbb{Z})^\times$ , automorphisms can be parametrized by integer vectors  $a = (a_1, \dots, a_r) \in \mathbb{Z}^r$  such that  $a_i$  is co-prime to  $p_i$ . An application of  $a$  to a basis element  $B \ni b = \prod_{i=1}^r x_i^{e_i}$  can thus be computed as  $a(b) = b^a = \prod_{i=1}^r x_i^{a_i e_i}$  - followed by a reduction modulo

I. The cost of an application of any automorphism  $a$  to an arbitrary  $\alpha \in K$  is therefore  $O(\#\alpha)$  plus the cost  $r$  for a reduction. An elementary argument shows that  $r = O(\#\alpha \max_{i=1}^r \phi(p_i^{n_i}))$ .

*Inversion.* Inversion is a more complicated operation. We will give two algorithms that can be used to compute inverses: the first is based on the minimal polynomial, while the second uses the automorphism group.

The first method is straightforward: given  $0 \neq \alpha \in K$ , using Algorithm 1, compute a polynomial  $f = \sum_{i=0}^l a_i x^i$  of minimal degree such that  $f(\alpha) = 0$ . Now,  $\beta := -1/a_0 \sum_{i=1}^l a_i \alpha^{i-1}$  is the inverse. By reusing the matrix  $M$  built in Algorithm 1, we can compute  $\beta$  without any additional operations in  $K$ , thus obtaining the inverse with the same complexity as the minimal polynomial.

The second method is based on identities  $1/\alpha = \prod_{g \in G, g \neq 1} \alpha^g / N(\alpha)$  and  $N(\alpha) = \prod_{g \in G} \alpha^g$  and the fact that products (and sums) over all group elements of an Abelian group can be evaluated efficiently.

**Algorithm 2 (Inversion)** *Let  $0 \neq \alpha \in K$  be arbitrary and  $G = \text{Gal}(K/\mathbb{Q}) = \prod_{i=1}^l \langle g_i \rangle$  decomposed into a direct product with  $\#\langle g_i \rangle = c_i$ .*

1. Set  $i := 1$ ,  $\beta := \alpha$  and  $\gamma := 1$ .
2. While  $i \leq l$  do
3.   Compute  $\beta := \prod_{i=1}^{c_i-1} \alpha^{g_i^i}$ ,  $\gamma := \gamma\beta$  and  $\alpha := \beta\alpha$ .
4. Return  $\gamma/\beta$

*Proof.* After each iteration, we have  $\gamma = \prod_{1 \neq g \in G_i} \alpha^g$  and  $\beta := \prod_{g \in G_i} \alpha^g$  where  $G_i := \langle g_1, \dots, g_i \rangle$ . Thus at the end of the algorithm,  $\beta \in \mathbb{Q}$  so that the division is easily done.

The complexity of this operation is easily estimated: we need  $\sum_{i=1}^l c_i - 1$  automorphism applications and  $\sum_{i=1}^l c_i$  multiplications in  $K$ .

For further optimization, note that we can essentially omit the Step 3 for all  $i$  where  $\beta = \beta^{g_i}$  and thus reduce the complexity for elements in certain subfields.

We also note that similar techniques can be used to compute the norm and trace of elements and easy modifications allow one to compute a set of all algebraic conjugates of an element, ie. a full Galois-orbit. This full orbit can then be used to compute the degree of an element and its minimal polynomial.

## 4 Embeddings

From the point of view of character theory, an important property of the sparse representation is that it is trivial to decide if an element already lies in a smaller cyclotomic field and, if so, to compute the new representation. If we set  $\zeta_n := \exp(2\pi i/n)$  then for all  $ml = n$  we have  $\zeta_n^l = \zeta_m$ , in particular,  $\zeta_{p^{l+1}}^p = \zeta_{p^l}$ . Therefore for a monomial  $\alpha = \prod_{i=1}^r \zeta_{p_i}^{e_i} \in \mathbb{Q}(\zeta_n)$  the smallest  $m|n$  such that  $\alpha \in \mathbb{Q}(\zeta_m)$  is obtained as  $m = \prod_{i=1}^r p_i^{l_i}$  for  $l_i = n_i - v_{p_i}(e_i)$  and the new representation is  $\alpha = \prod_{i=1}^r \zeta_{p_i}^{f_i}$  for  $f_i := e_i/p_i^{v_{p_i}(e_i)}$ .

In order to decide if  $\alpha \in \mathbb{Q}(\zeta_m)$  for some given  $m|n$  we only need to check if all the exponents are divisible by the correct prime powers. Similarly, to represent  $\alpha$  in a larger field, the exponents have to be scaled by some powers of  $p_i$ .

A related task that is needed frequently is to find a fixed primitive root of unity in a given field. However, while for  $p$  and  $q$  coprime,  $\zeta_p \zeta_q$  is a primitive  $pq$ -th root of unity, in general,  $\zeta_p \zeta_q \neq \zeta_{pq}$ . We make use of the following algorithm:

**Algorithm 3** *Let  $K := \mathbb{Q}(\zeta_n)$  be given in sparse representation and let  $m|n$  be arbitrary.*

1. Set  $a_i = n/p_i^{n_i}$  for  $i \leq i \leq r$
2. Compute (using the extended GCD)  $b_i$  s.th.  $a_i b_i \equiv 1 \pmod{p_i^{n_i}}$
3. Set  $\zeta_n := \prod_{i=1}^r \zeta_{p_i}^{b_i}$
4. Return  $\zeta_n^{n/m}$

*Proof.* We must have  $\zeta_n^{a_i} = \zeta_{p_i}^{n_i}$  in view of the complex embeddings, expanding  $\zeta_n$ , we get

$$\zeta_n^{a_i} = \prod_{j=1}^r \zeta_{p_j}^{a_i b_j} = \prod_{j=1}^r (\zeta_n^{a_j})^{a_i b_j}.$$

Now  $a_i b_j$  is divisible by  $p_j^{n_j}$  for all  $j \neq i$  since  $a_i$  already is, thus  $\zeta_n^{a_j a_i b_j} = 1$ . For  $j = i$  our construction gives  $\zeta_n^{a_i a_j b_j} = (\zeta_n^{a_i})^{a_j b_j} = \zeta_{p_i}^{a_j b_j} = \zeta_{p_i}^{n_i}$  as desired.

This algorithm can obviously also be used to compute sparse representations from dense ones, the reverse, sparse from dense being trivial.

## 5 Prime splitting

We consider only the unramified primes here, so let  $p$  be a prime, co-prime to  $n$ . The prime will split in the maximal order  $\mathbb{Z}_K = \mathbb{Z}[\zeta_n]$  into a product of prime ideals:

$$p\mathbb{Z}_K = \prod_{i=1}^l P_i$$

where  $f_i := \deg P_i := \deg(\mathbb{Z}_K/P_i : \mathbb{GF}(p))$ , the degree of the residue class field of  $P_i$ , is constant for all  $i$ . Class field theory easily gives  $f = f_i = \text{ord}(p(n\mathbb{Z}))$  in the group  $(\mathbb{Z}/n\mathbb{Z})^\times$ , so we can assume the degree to be known. The standard way of computing the splitting behaviour of an unramified prime is based on a theorem of Kummer:

**Theorem 2.** *Let  $g$  be the minimal polynomial for a primitive element  $\alpha$  of  $K/\mathbb{Q}$ . If  $p$  is a prime co-prime to the discriminant of  $g$ , then  $p\mathbb{Z}_K = \prod_{i=1}^l P_i$  and  $P_i = \langle p, g_i(\alpha) \rangle$  where  $g_i$  is a lift from the factorisation  $g = \prod_{i=1}^l g_i \pmod{p\mathbb{Z}}$ .*

Of course, since in our case the degree  $(K : \mathbb{Q})$  is large and the defining polynomial is not “known”, we cannot directly use this theorem, nevertheless, it is the foundation of our method:

**Algorithm 4 (Prime Splitting)** Let  $K := \mathbb{Q}(\zeta_n)$  be given in sparse representation and let  $p$  a prime co-prime to  $n$ .

1. Compute  $f = \text{ord}(p)$  in  $(\mathbb{Z}/n\mathbb{Z})^\times$
2. Let  $C := \text{GF}(p, f)$ ,  $z \in C^\times$  be an element of order  $n$  and  $z_n$  be a primitive  $n$ th root of unity given by Algorithm 3.
3. Compute  $\bar{g} := \prod_{i=1}^f (x - z^{p^i}) \in \text{GF}(p)[x]$  and a lift  $g \in \mathbb{Z}[x]$  of  $\bar{g}$ .
4. Let  $I := \{g\}$ ,  $U := \langle p \rangle < (\mathbb{Z}/n\mathbb{Z})^\times$  and  $S := (\mathbb{Z}/n\mathbb{Z})^\times / U$  a set of coset representatives.
5. Set  $I := \{\langle p, g(z_n^s) \rangle \mid s \in S\}$ .

*Proof.* The validity of the algorithm follows directly from Kummer's theorem 2 and the fact that the Galois group of a number field operates transitively on the prime ideals lying over a fixed prime number.

The complexity of the algorithm can roughly be estimated to depend on  $f$  and  $\#S = n/f$ .

While this very simple method is certainly extremely efficient for cyclotomic fields and unramified primes, it does not easily generalize to arbitrary number fields, not even to normal or Abelian ones.

## 6 Comparison

In the last few sections we demonstrated how the sparse representation can be used to implement efficient algorithms for some tasks relating to representation theory. In particular for the computation of characters of finite groups there are other representations for cyclotomic numbers known in the literature. In [2] a basis for the ring of integers is suggested such that certain subsets form bases for cyclotomic subfields, and in [3] a different approach gives bases such that bases for all Abelian subfields can easily be obtained.

In general, it is difficult to compare the different methods as their implementation follows completely different strategies. However, some observations can be made.

Firstly, [2] focuses essentially on a representation that allows for easy recognition of numbers that are in cyclotomic subfields. A close examination of our Section 4 shows that our field basis has essentially the same properties as Bosma's basis, namely that subfields correspond to subsets. However, since our implementation is based on sparse multivariate polynomials, while Bosma's is based on dense elements, it is clear that for sparse elements our representation will be much more efficient. Moreover, elements in cyclotomic subfields automatically get almost optimal arithmetic, even without changing their representation to reflect the smaller fields. Thus the problem of finding a good strategy to decide when to find minimal fields of definition is far less important and can usually be deferred right to the end. Also, since Bosma uses a "generic" basis, multiplication of elements has to be done either by using the structure constants or by changing the representation. Both possibilities require either slow algorithms or

the storage of a large amount of data,  $n^2$  elements for structure constants or  $2n$  elements for a base change, both of which is infeasible for huge values of  $n$ .

For the second method [3], we can actually directly compare the algorithms since they are implemented in the gap-system [5]. The gap implementation also uses a sparse representation. Based on architectural differences, the implementation is limited to  $n < 2^{28}$ , while the Magma version can handle larger  $n$  if the prime powers are bounded by  $2^{30}$ . In practice, none of those limitations matter, as general operations become very slow regardless. The algorithms used for multiplication and addition are essentially the same as ours, the main difference is that the polynomial reduction is replaced by an explicit formula re-writing “wrong” powers of the primitive element in terms of the basis, thus the complexity should be the same. For the computation of inverses, they use the same idea as we do in Algorithm 2, but instead of using the structure of the automorphism group, they use a list of all automorphisms, and thus obtain a far worse runtime. Since Breuer’s motivation is purely group theoretical, they do not give any algorithms for minimal polynomials, norm and trace computations or prime splitting. On the other hand, their representation allows in principle to find minimal fields of definition easily, while ours only finds minimal cyclotomic fields. However, in practice, no one seems to use non-cyclotomic fields.

## 7 Generalisation

Several of the algorithms presented here for sparse cyclotomic fields apply easily to arbitrary sparse represented number fields. The algorithms that do not carry over easily are the ones that rely on the knowledge of the (Abelian) automorphism group (Algorithm 2) and the primitive element (Algorithm 4). While it is clear that inverses can be computed using minimal polynomials, obviously, if the automorphism group is known, a variation of Algorithm 2 does apply.

So it remains to give an alternative to Algorithm 4. In [8], it is shown that the polynomial factorisation in Kummer’s theorem can be replaced by a primary decomposition, however, in practice, this is not fast enough. Alternatively, also in [8], an algorithm is given where a randomly chosen primitive element is used, the difficulties being that the verification of primitivity is rather expensive and the construction of the minimal polynomial is complicated. Here we suggest a different method:

**Algorithm 5** *Let  $K := \mathbb{Q}(\alpha_1, \dots, \alpha_r)$  be given and assume that the intermediate fields  $\mathbb{Q}(\alpha_i)$  have disjoint normal closures. Let  $f_i$  be the minimal polynomial of  $\alpha_i$ . Furthermore, let  $p$  be a rational prime that is co-prime to the discriminants of the  $f_i$  for all  $1 \leq i \leq r$ .*

1. Set  $I := \{\}$ .
2. Compute  $l_{i,j} \in \text{GF}(p)[x]$  such that  $f_i = \prod_{j=1}^{r_i} l_{i,j}$ .
3. Set  $L_i := \{l_{i,j} \mid 1 \leq j \leq r_i\}$ .
4. For all  $l = (l_1, \dots, l_r) \in L_1 \times \dots \times L_r$  do
5.   Compute  $d := \text{lcm}\{\deg l_i \mid 1 \leq i \leq r\}$  and set  $k := \text{GF}(p, d)$ .

6. Compute  $R_i := \{x \in k \mid l_i(x) = 0\}$  ( $1 \leq i \leq r$ ) and set  $R := \{xi \in R_1 \times \cdots \times R_r\}$ .
7. While  $R \neq \emptyset$  do
8. Fix some  $x \in R$  and set  $R := R \setminus \{(x_i^{p^j})_{1 \leq i \leq r} \mid 1 \leq j \leq d\}$ .
9. Set  $\psi : K \rightarrow k : \alpha_i \mapsto x_i$  and  $\phi : k \rightarrow \text{GF}(p)^d$  and compute a basis  $b$  for the nullspace of  $\phi \circ \psi$  restricted to  $\mathbb{Z}[B]$  as a map between modules.
10. Finally, set  $I := I \cup \langle b \rangle$ .

*Proof.* The condition on  $p$  guarantees that  $\mathbb{Z}[B]$  is  $p$ -maximal as an order in  $K$  (implying that prime ideals in  $\mathbb{Z}[B]$  “are” primes in the maximal order as well) and that  $f_i$  is square-free over  $\text{GF}(p)$ . Since  $\psi$  is obviously a (surjective) ring homomorphism from  $\mathbb{Z}[B]$  onto  $k$ , its kernel is an ideal. Since  $k$  is a field, the ideal has to be prime. Using the fact that  $\phi$  is an isomorphism of  $\text{GF}(p)$ -vectorspaces, it is now obvious that the kernel of  $\phi \circ \psi$  generates the prime ideal that has  $\psi$  as a residue class field map.

The algorithm can be optimized in various ways, for example, since  $p\mathbb{Z}[B]$  is obviously contained in the kernel of  $\psi$ , we can compute the kernel as a nullspace over  $\text{GF}(p)$  and supplement it afterwards.

A major difference between Algorithms 4 and 5 is the way the prime ideals are represented: in Algorithm 4 the ideals are given in a very compact form using only 2 generators while Algorithm 5 computes only a  $\mathbb{Z}$ -basis for the ideals. While this does not appear to be a major problem it limits the applicability quite severely: the second generator (or a close relative) is crucial for many algorithms, for example it is used to compute valuations at this prime. One way to overcome this is to randomly choose elements of the ideal and test if they are suitable as 2nd generators. While this method is usually successful, it still has two problems. First, as pointed out by Belabas [1], for small prime numbers that are highly split, the probabilities for randomly choosing a suitable element are quite small, and secondly, the test for suitability involves norm computations which easily dominate the running time.

If a primitive element  $\beta$  for  $K/\mathbb{Q}$  as a polynomial in the  $\alpha_i$  is known and if the prime  $p$  is co-prime to the discriminant of  $\mathbb{Z}[\beta]$ , then Algorithm 5 can easily be adapted to compute 2-element representations as well.

It should also be noted that the complexity of the computation of the complete prime-splitting depends on the degree of  $K/\mathbb{Q}$ , so that even the more optimized algorithms like Algorithm 5 cannot be applied to really large fields.

## 8 Examples

We want to illustrate the power of the sparse method and demonstrate that the very rough complexity analysis of the previous sections does reflect the behaviour of the algorithms properly.

We start with simple arithmetic. We will work in the fields  $\mathbb{Q}(\zeta_n)$  for  $n = 7^l - 1$ , using random elements with a growing number  $n_a$  of non-zero coefficients in the range  $[0, n_b]$ . Table 1 compares times for elements with small coefficients

$n_b = 10$  in moderately large fields ( $n = 7^4 - 1, \phi(n) = 640$ ) for basic operations ( $+, \times$ ) in sparse and dense representation. It is easy to see that the time for the dense representation is independent on the sparsity.

In the next table, Table 2, we compare the times for basic operations in a family of fields,  $n_a = 7^l - 1$  for  $l = 2, \dots, 8$ . Again, the times support our rough complexity analysis, as they show that the times depends only on the number of non-zero coefficients and is independent of the field degree, the dependence on the degree in the table for small  $l$  is due to the fact that the elements are relatively dense in those examples,  $\phi(7^2 - 1) = 16$ ,  $\phi(7^3 - 1) = 108$  thus the multiplication is dominated by the reduction - which, for dense elements, is mainly dependent on the representation of the field.

In the last table, Table 3, we give timings for the computation of minimal polynomials of small degree and of inverses using Algorithm 2. We first choose a “random” subgroup  $U$  of the automorphism group  $G \cong (\mathbb{Z}/(7^4 - 1)\mathbb{Z})^\times$  of  $K = \mathbb{Q}(\zeta_{7^4 - 1})$  of small index  $l = 4, 8, 16, 20$ , then compute a basis for the field fixed by  $U$  and finally choose small linear combinations of those basis elements. We also give the average number of non-zero coefficients with respect to  $B$ . It should be noted that the times for minimal polynomials are obtained using an optimized implementation in the c-language, while the inverses were computed using a (crude) magma language implementation. So while the times should not be compared directly, it can be noted that the times for minimal polynomial depend strongly on the degree of the polynomial, while the inverse depends on the structure of  $G$ . So for elements in small degree subfields, the minimal polynomial method is better suited for inverses than Algorithm 2 which depends on the sparsity and the structure of the automorphism group.

Finally, we present an example showing the overall impact of the sparse representation in computational class field theory. Starting with the field  $k := \mathbb{Q}(\sqrt{10})$ , we compute  $R$  the 5-part of the ray class group modulo  $5^2 \cdot 11 \cdot 31$  which is isomorphic to  $C_5^4$ . Since the defining modulus is invariant under the  $\mathbb{Q}$ -automorphisms of  $k$ ,  $R$ , as an  $\text{GF}(5)$ -module has an induced action of  $\sqrt{10} \mapsto -\sqrt{10}$ . Under this action  $R$  has a unique invariant subspace isomorphic to  $C_5$ , thus  $R$  has a Galois-stable quotient that is isomorphic to  $C_5^3$ . Using Magma, we can compute a sparse representation for the corresponding field  $K$  in 2.6 seconds, ie. 3 polynomials of degree 5, that can be printed using 400 characters. In this representation it takes Magma further 2 minutes to compute explicitly 3 generating  $k$ -automorphisms and an extension of the  $\mathbb{Q}$ -automorphism of  $k$  to  $K$ . This computations uses the sparse representation for both  $K$  and the Kummer extension  $K(\zeta_5)/k(\zeta_5)$ . Each of the automorphisms can be written down using less than 1000 characters, that is, the total number of digits in the coefficients of the images of the generators is reasonable small (for a field of degree 250 over  $\mathbb{Q}$ ). On the other hand, it takes Magma 450 seconds to compute the minimal polynomial of a sum of the three generators. The resulting polynomial, which can be used to define  $K$  in the traditional way, needs a total of about 50,000 characters to print, each of the generating automorphisms takes more than 200,000 characters to write down, making them totally useless for any further applications.

**Table 1.**  $l = 4$ , degree  $(\mathbb{Q}(\zeta_{7^4-1}) : \mathbb{Q}) = 640$ ,  $n_b = 10$ . The first two rows are timings in sparse representation, the last two are for the dense model. Times are for 1000 operations each.

$n_a =$	4	8	12	16	20	30	40	50	100	200	300
+	0.000	0.010	0.010	0.010	0.010	0.010	0.010	0.020	0.020	0.030	0.040
×	0.020	0.050	0.120	0.180	0.360	0.570	1.020	1.410	3.510	8.620	16.930
+	0.010	0.020	0.010	0.020	0.010	0.010	0.020	0.010	0.020	0.010	0.010
×	0.090	23.910	24.140	24.240	24.310	24.160	24.340	24.330	24.310	24.320	24.270

**Table 2.**  $l = 2, \dots, 8$ ,  $n_a = 4, 8, 12, 20, 30, 40$ ,  $n_b = 10$ . Times are for 1000 operations each

$l   n_a =$	4	8	12	20	30	40
2 +	0.020	0.030	0.030	0.030	0.030	0.030
2 ×	0.030	0.050	0.060	0.070	0.090	0.090
3 +	0.030	0.020	0.030	0.040	0.040	0.060
3 ×	0.040	0.070	0.120	0.210	0.330	0.440
4 +	0.020	0.030	0.030	0.040	0.050	0.060
4 ×	0.040	0.080	0.150	0.370	0.720	1.130
5 +	0.040	0.020	0.030	0.040	0.050	0.060
5 ×	0.030	0.080	0.150	0.400	0.870	1.460
6 +	0.020	0.030	0.040	0.050	0.050	0.060
6 ×	0.040	0.080	0.150	0.410	0.940	1.710
7 +	0.050	0.030	0.030	0.040	0.050	0.060
7 ×	0.040	0.080	0.150	0.460	0.970	1.760
8 +	0.030	0.030	0.030	0.050	0.050	0.060
8 ×	0.040	0.080	0.160	0.410	0.970	1.870

**Table 3.** Minimal polynomials and inverses of elements of degree  $l = 4, 8, 16, 20$  in  $\mathbb{Q}(\zeta_n)$  for  $n = 7^4 - 1$  of elements with “small” coefficients. The times are for 100 random elements of the same subfield each.

$l =$	4	8	16	20
$\#a$	13	9	20	60
$f$	0.240	0.380	2.120	11.800
$()^{-1}$	10.660	10.460	12.190	22.340

## References

1. Belabas, K.: Topics in computational algebraic number theory. *J. Theor. Nombres Bordeaux* 16, 19-63 (2004).
2. Bosma, W.: Canonical bases for cyclotomic fields. *Appl. Algebra Eng. Commun. Comput.* 1, No.2, 125-134 (1990).
3. Breuer, T.: Integral bases for subfields of cyclotomic fields. *Appl. Algebra Eng. Commun. Comput.* 8, No.4, 279-289 (1997).
4. Fieker, C.: Computing class fields via the Artin map. *Math. Comput.* 70, 1293-1303 (2001).
5. GAP4: <http://www-gap.mcs.st-and.ac.uk/>
6. Bosma, W., Cannon, J. and Playoust, C.: The Magma Algebra System I: The User Language. *J. of Symb. Comput.* 24, No 3, 235-265 (1997).
7. Magma: <http://magma.maths.usyd.edu.au/>
8. Pohl, S.: Primidealzerlegung in Komposita von Zahlkörpern. Diplom Thesis, TU-Berlin, 2002.
9. Unger, W.: Computing the character table of a finite group. Submitted to *J. of Symb. Comput.* (2005).
10. Gathen, J. von zur., Gerhard, J.: Modern computer algebra. Cambridge University Press (1999).