Generating Subfields

Mark van Hoeij

Florida State University
Tallahassee, FL 32306

Jürgen Klüners

Mathematisches Institut
der Universität Paderborn
Universität Paderborn
Warburger Str. 100
33098 Paderborn, Germany

Andrew Novocin

David R. Cheriton School of Computer Science
University of Waterloo, Ontario, Canada N2L 3G1

Abstract

Given a field extension K/k of degree n we are interested in finding the subfields of K containing k. There can be more than polynomially many subfields. We introduce the notion of generating subfields, a set of up to n subfields whose intersections give the rest. We provide an efficient algorithm which uses linear algebra in k or lattice reduction along with factorization in any extension of K. Implementations show that previously difficult cases can now be handled.

Key words: Symbolic Computation, Subfields, Lattice Reduction

⋆ This research was partly supported by NSF Grant number 0728853 and 1017880 and by the Deutsche Forschungsgemeinschaft

Email addresses: `hoeij@math.fsu.edu` (Mark van Hoeij), `klueners@math.uni-paderborn.de` (Jürgen Klüners), `andy@novocin.com` (Andrew Novocin).

1. Introduction

Let K/k be a finite separable field extension of degree n and α a primitive element of K over k with minimal polynomial $f \in k[x]$. We explore the problem of computing subfields of K which contain k. We prove that all such subfields (there might be more than polynomially many) can be expressed as the intersections of at most n particular subfields which we will call the 'generating subfields'. We give an efficient algorithm to compute these generating subfields. Subfields are used in finding the Galois group of polynomials and can be used when simplifying expressions involving algebraic numbers (see section 3.4 for an example).

Previous methods progress by solving combinatorial problems on the roots of f, such as [4; 5; 8; 13]. Similar to our algorithm [11] starts by factoring f over K and then tries to find all subfield polynomials (see Definition 5) by a combinatorial approach. Such approaches can be very efficient, but in the worst cases they face a combinatorial explosion. While [14] proceeds by factoring resolvent polynomials of degree bounded by $\binom{n}{\lfloor n/2 \rfloor}$. By introducing the concept of generating subfields we restrict our search to a small number of target subfields. This new fundamental object allows for polynomial time algorithms.

We can find the generating subfields whenever we have a factorization algorithm for f over K or any $\tilde{K}/K$ and the ability to compute a kernel in k. For $k = \mathbb{Q}$ this implies a polynomial-time algorithm as factoring over $\mathbb{Q}(\alpha)$ and linear algebra over $k = \mathbb{Q}$ are polynomial time. When one desires all subfields we give such an algorithm which is additionally linear in the number of subfields.

For the number field case we are interested in a specialized and practical algorithm. Thus we replace exact factorization over $\mathbb{Q}(\alpha)$ by a p-adic factorization and the exact kernel computation by approximate linear algebra using the famous LLL algorithm for lattice reduction [15]. We take advantage of some recent practical lattice reduction results [19] and tight theoretical bounds to create an implementation which is practical on previously difficult examples.

ROADMAP: The concept of the principal and generating subfields are introduced in Section 2.1. In Section 2.2 we explain how to compute all subfields in a running time which is linearly dependent on the number of subfields. For the number field case we will use the LLL algorithm and this case is handled in detail in Section 3. Finally we compare our approach with the state of the art in Section 4.

NOTATIONS: For a polynomial g we let $\| g \|$ be the ℓ_2 norm on the coefficient vector of g. For a vector $\mathbf{v}$ we let $\mathbf{v}[i]$ be the i^{th} entry. Unless otherwise noted $\| \cdot \|$ will represent the ℓ_2 norm.

2. A general algorithm

2.1. Generating subfields

In this section we introduce the concept of a generating set of subfields and prove some important properties. Let $\tilde{K}$ be a field containing K. We remark that we can choose $\tilde{K} = K$, but in some case it might be better to choose a larger $\tilde{K}$ from an algorithmic point of view. E.g. in the number field case we choose a p-adic completion (see Section 3). Let $f = f_1 \cdots f_r$ be the factorization of f over $\tilde{K}$ where the $f_i \in \tilde{K}[x]$ are irreducible and $f_1 = x - \alpha$. We define the fields $\tilde{K}_i := \tilde{K}[x]/(f_i)$ for $1 \le i \le r$. We denote elements of K as $g(\alpha)$ where $g \in k[x]$ is a polynomial of degree $< n$, and define for $1 \le i \le r$ the embedding

$$\phi_i : K \to \tilde{K}_i, \qquad g(\alpha) \mapsto g(x) \bmod f_i.$$

Note that ϕ_1 is just the identity map $\mathrm{id} : K \to \tilde{K}$. We define for $1 \le i \le r$:

$$L_i := \mathrm{Ker}(\phi_i - \mathrm{id}) = \{g(\alpha) \in K \mid g(x) \equiv g(\alpha) \bmod f_i\}.$$

The L_i are closed under multiplication, and hence fields, since $\phi_i(ab) = \phi_i(a)\phi_i(b) = ab$ for all $a, b \in L_i$.

Theorem 1. *If L is a subfield of K/k then L is the intersection of L_i, $i \in I$ for some $I \subseteq \{1, \ldots, r\}$.*

Proof. Let f_L be the minimal polynomial of α over L. Then f_L divides f since $k \subseteq L$, and $f_L = \prod_{i \in I} f_i$ for some $I \subseteq \{1, \ldots, r\}$ because $L \subseteq \tilde{K}$. We will prove

$$L = \{g(\alpha) \in K \mid g(x) \equiv g(\alpha) \bmod f_L\} = \bigcap_{i \in I} L_i.$$

If $g(\alpha) \in L$ then $h(x) := g(x) - g(\alpha) \in L[x]$ is divisible by $x - \alpha$ in $K[x]$. The set of polynomials in $L[x]$ divisible by $x - \alpha$ is the principal ideal (f_L) by definition of f_L. Then $h(x) \equiv 0 \bmod f_L$ and hence $g(x) \equiv g(\alpha) \bmod f_L$. Conversely, $g(x) \bmod f_L$ is in $L[x]$ (mod f_L) because division by f_L can only introduce coefficients in L. So if $g(x) \equiv g(\alpha) \bmod f_L$ then $g(\alpha) \in K \cap L[x] = L$.

By separability and the Chinese remainder theorem, one has $g(x) \equiv g(\alpha) \bmod f_L$ if and only if $g(x) \equiv g(\alpha) \bmod f_i$ (i.e. $g(\alpha) \in L_i$) for every $i \in I$. □

Lemma 2. The set $S := \{L_1, \ldots, L_r\}$ is independent of the choice of $\tilde{K}$.

Proof. Let $f = g_1 \cdots g_s \in K[x]$ be the factorization of f into irreducible factors over K. Suppose that f_i divides g_l. Let L resp. L_i be the subfield corresponding to g_l resp. f_i. Assume $g(\alpha) \in L$, in other words $g(x) \equiv g(\alpha) \bmod g_l$. Then $g(x) \equiv g(\alpha) \bmod f_i$ because f_i divides g_l. Hence $g(\alpha) \in L_i$.

Conversely, assume that $g(\alpha) \in L_i$. Now $h(x) := g(x) - g(\alpha)$ is divisible by f_i, but since $h(x) \in L_i[x] \subseteq K[x]$ it must also be divisible by g_l since g_l is irreducible in $K[x]$ and divisible by f_i. So $g(x) \equiv g(\alpha) \bmod g_l$ in other words $g(\alpha) \in L$. It follows that $L = L_i$. □

Definition 3. We call the fields $L_1, \ldots, L_r$ the *principal subfields* of K/k. A set S of subfields of K/k is called a *generating set* of K/k if every subfield of K/k can be written as $\bigcap T$ for some $T \subseteq S$. Here $\bigcap T$ denotes the intersection of all $L \in T$, and $\bigcap \emptyset$ refers to K. A subfield L of K/k is called a *generating subfield* if it satisfies the following equivalent conditions

(1) The intersection of all fields L' with $L \subsetneq L' \subseteq K$ is not equal to L.
(2) There is precisely one field $L \subsetneq \tilde{L} \subseteq K$ for which there is no field between L and $\tilde{L}$ (and not equal to L or $\tilde{L}$).

The field $\tilde{L}$ in condition 2. is called *the field right above* L. It is clear that $\tilde{L}$ is the intersection in condition 1., so the two conditions are equivalent.

The field K is a principal subfield but not a generating subfield. A maximal subfield of K/k is a generating subfield as well. Theorem 1 says that the principal subfields form a generating set. By condition 1., a generating subfield can not be obtained by intersecting larger subfields, and must therefore be an element of every generating set. In particular, a generating subfield is also a principal subfield.

If S is a generating set, and we remove every $L \in S$ for which $\bigcap\{L' \in S | L \subsetneq L'\}$ equals L, then what remains is a generating set that contains only generating subfields. It follows that

Proposition 4. *S is a generating set if and only if every generating subfield is in S.*

Here we just want to illustrate the requirements for finding a generating set of subfields in polynomial time. Suppose that K/k is a finite separable field extension and that one has polynomial time algorithms for factoring over K and linear algebra over k (for example when $k = \mathbb{Q}$). Then applying Theorem 1 with $\tilde{K} = K$ yields a generating set S with $r \le n$ elements in polynomial time. We may want to minimize r by removing all elements of S that are not generating subfields, then $r \le n - 1$.

Note that the computation of the principal subfields L_i is trivial when we know a factorization of f over K. In this case we get a k-basis of L_i by a simple kernel computation. In the number field case, the factorization of f over K is the bottleneck. Therefore for some fields k we prefer to take a larger field $\tilde{K} \supsetneq K$ where the factorization is faster. In Section 3 this is done for $k = \mathbb{Q}$, but this can be generalized to an arbitrary global field. Then we let $\tilde{K}$ be some completion of K. This reduces the cost of the factorization, however, one now has to work with approximations for the factors f_i of f, which means that we get approximate (if $\tilde{K}$ is the field of p-adic numbers then this means modulo a prime power) linear equations. Solving approximate equations involves LLL in the number field case and [2; 7] in the function field case.

2.2. All subfields

We would like to compute all subfields of K/k by intersecting elements of a generating set $S = \{L_1, \ldots, L_r\}$. Figure 1 is an algorithm with complexity proportional to the number of subfields of K/k. Unfortunately there exist families of examples where this number is more than polynomial in n. Note that we represent our subfields $k \le L_i \le K$ as k-vector subspaces of K. This allows the intersection $L_1 \cap L_2$ to be found with linear algebra as the intersection of two subspaces of a vector space. To each subfield L of K/k we associate a tuple $e = (e_1, \ldots, e_r) \in \{0,1\}^r$, where $e_i = 1$ if and only if $L \subseteq L_i$.

Input: A generating set $S = \{L_1, \ldots, L_r\}$ for K/k.
Output: All subfields of K/k.
1. Let $e := (e_1, \ldots, e_r)$ be the associated tuple of K.
2. ListSubfields := $[K]$.
3. Call `NextSubfields`$(S, K, e, 0)$.
4. Return ListSubfields.

Fig. 1. `AllSubfields` Algorithm

The algorithm of figure 2 returns no output but appends elements to ListSubfields, which is used as a global variable. The input consists of a generating set, a subfield L, its associated tuple $e = (e_1, \ldots, e_r)$, and the smallest integer $0 \leq s \leq r$ for which $L = \bigcap\{L_i \mid 1 \leq i \leq s,\ e_i = 1\}$.

Input: S, L, e, s.
For all i with $e_i = 0$ and $s < i \leq r$ **do**
1. Let $M := L \cap L_i$.
2. Let $\tilde{e}$ be the associated tuple of M.
3. **If** $\tilde{e}_j \leq e_j$ for all $1 \leq j < i$ **then**
4. append M to ListSubfields and call `NextSubfields`$(S, M, \tilde{e}, i)$.

Fig. 2. `NextSubfields` Algorithm

Definition 5. Let L be a subfield of K/k. Then the minimal polynomial f_L of α over L is called the **subfield polynomial** of L.

Remark 6. Let $g \in K[x]$ be a monic polynomial. Then the following are equivalent:
(1) $g = f_L$ for some subfield L of K/k.
(2) $f_1 \mid g \mid f$ and $[\mathbb{Q}(\alpha) : \mathbb{Q}(\text{coefficients}(g))] = \text{degree}(g)$.
(3) $f_1 \mid g \mid f$ and the $\mathbb{Q}$–vector space $\{h(x) \in \mathbb{Q}[x] \mid \deg(h) < \deg(f), h \mod g = h \mod f_1\}$ has dimension $\deg(f)/\deg(g)$.

Remark 7. For each subfield L, we can compute the subfield polynomial f_L with linear algebra. Testing if $L \subseteq M$ then reduces to testing if f_L is divisible by f_M. For many fields K this test can be implemented efficiently by choosing a non-archimedian valuation v of K with residue field $\mathbf{F}$ such that the $f \mod v$ (the image of f in $\mathbf{F}[x]$) is defined and separable. Then f_L is divisible by f_M in $K[x]$ if and only if the same is true mod v, since both are factors of a polynomial f whose discriminant does not vanish mod v.

Subfields that are isomorphic but not identical are considered to be different in this paper. For instance, let $f = x^6 + 3$. Now $\mathbb{Q}[x]/(f)$ is isomorphic to the splitting field of $x^3 - 3$. This polynomial has 3 roots, $\sqrt[3]{3}, \omega\sqrt[3]{3}, \omega^2\sqrt[3]{3}$ where ω is a third root of unity. Each of these roots gives a subfield of degree 3 that is isomorphic to $\mathbb{Q}[x]/(x^3 - 3)$. We consider these as 3 distinct subfields.

Let m be the number of subfields of K/k. Since S is a generating set, all subfields occur as intersections of $L_1, \ldots, L_r$. The condition in Step (3) in Algorithm NextSubfields holds if and only if M has not already been computed before. So each subfield will be placed in ListSubfields precisely once, and the total number of calls to Algorithm NextSubfields equals m. For each call, the number of i's with $e_i = 0$ and $s < i \leq r$ is bounded by r, so the total number of intersections calculated in Step (1) is $\leq rm$. Step (2) involves testing which L_j contain M. Bounding the number of j's by r, the number of subset tests is $\leq r^2 m$. One can implement Remark 7 to keep the cost of each test low.

Theorem 8. *Given a generating set for K/k with r elements, Algorithm AllSubfields returns all subfields by computing at most rm intersections and at most $r^2 m$ subset tests, where m is the number of subfields of K/k.*

2.3. Quadratic subfields

As a theoretical application, to illustrate the framework of generating subfields, we note that all quadratic subfields can be computed in polynomial time when we already know the generating subfields. Note that during our discussion we encounter a field extension with Galois group C_2^s, which is the simplest example of a field extension which has more than polynomially many subfields.

Let $Q(K/k)$ denote the subfield generated over k by $\{a \in K | a^2 \in k\}$, and let C_2 denote the cyclic group of order 2. If $K = Q(K/k)$, in other words the Galois group of f is C_2^s for some s, then $n = 2^s$ and f splits over K into linear factors $f_1 \cdots f_n$ where $f_1 = x - \alpha$. Furthermore, there are precisely $n-1$ generating subfields $L_2, \ldots, L_n$ and n principal subfields $L_1, \ldots, L_n$ where $L_1 = K$.

Conversely, suppose there are n principal subfields. Every principal subfield corresponds to at least one factor of f over K, and hence to precisely one factor since f has degree n. So f must split into linear factors, and each L_i corresponds to precisely one linear factor f_i. Then the minimal polynomial of α over L_i is $f_1 f_i$ when $i \in \{2, \ldots, n\}$. The degree of $f_1 f_i$ is 2, so there are $n-1$ subfields of degree index 2, which implies that the Galois group is C_2^s for some s.

Theorem 9. *If factoring over K and linear algebra over k can be done in polynomial time then all quadratic subfields of K/k can be computed in polynomial time.*

Note that a subfield of degree index 2 of K/k corresponds to an autmorphism of K/k of order 2 which can be easily computed. Therefore the knowledge of all principal subfields of $Q(K/k)$ is equivalent to the knowledge of all automorphisms of the Galois group. Hence, the quadratic subfields of $Q(K/k)$ can be computed easily in polynomial time. So it suffices to prove that the following algorithm computes $Q(K/k)$ in polynomial time.

Input: A separable field extension K/k where $K = k(\alpha)$.
Output: $Q(K/k)$.

1. Let $n := [K : k]$. If n is odd then return k.
2. Compute the set S of generating subfields.
3. If K/k has $n-1$ distinct subfields of degree index 2 then return K.
4. Choose a generating subfield $L_i \in S$ with degree index > 2, and let $\tilde{L}_i$ be the field right above L_i, so $L_i \subsetneq \tilde{L}_i := \bigcap\{L_j \in S \,|\, L_i \subsetneq L_j\}$.
5. If $[\tilde{L}_i : L_i] = 2$ then return $Q(\tilde{L}_i/k)$, otherwise return $Q(L_i/k)$.

Fig. 3. Algorithm Q

In the first call to Algorithm Q, we can compute a generating set in Step (2) in polynomial time using Theorem 1 with $\tilde{K} := K$. For the recursive calls we use:

Remark 10. If S is a generating set for K/k and if L is a subfield of K/k, then $\{L \bigcap L' | L' \in S\}$ is a generating set of L/k.

For Step (3) see the remarks before Theorem 9. If we reach Step (4) then $K \neq Q(K/k)$. The field L_i in Step (4) exists by Lemma 11 below. Let $\tilde{L}_i$ be the field right above L_i. If $[\tilde{L}_i : L_i] = 2$ then $\tilde{L}_i \neq K$ so the algorithm terminates.

Let $a \in Q(K/k)$. We may assume that $a^2 \in k$. Now $\tilde{L}_i$ is contained in any subfield L' of K/k that properly contains L_i. So if $a \notin L_i$ then $L_i(a)$ contains $\tilde{L}_i$ and hence equals $\tilde{L}_i$ since $[L_i : L_i(a)] = 2$. Then $a \in \tilde{L}_i$. We conclude $Q(K/k) \subseteq \tilde{L}_i$. If $[\tilde{L}_i : L_i] \neq 2$ then the assumption $a \notin L_i$ leads to a contradiction since $L_i(a)$ can not contain $\tilde{L}_i$ in this case. So $Q(K/k) \subseteq L_i$ in this case, which proves that Step (5) is correct.

Lemma 11. If K/k does not have $n-1$ distinct subfields of degree index 2 then there exists a generating subfield of degree index > 2.

Proof. Assume that every generating (and hence every maximal) subfield has degree index 2. So the subfields of degree index 2 form a generating set. Let G be the automorphism group of K/k. If K/L_i and K/L_j are Galois extensions, then so is $K/(L_i \cap L_j)$ since $L_i \cap L_j$ is the fixed field of the group generated by the Galois groups of K/L_i and K/L_j. If $[K : L_i] = 2$ then K/L_i is Galois. Let k' be the intersection of all subfields L_i of degree index 2. Then K/k' is Galois. However, k' must equal k, otherwise the set of subfields of degree index 2 can not be a generating set. It follows that K/k is Galois.

If n is not a power of 2, then there exists a maximal subfield of odd degree index. If $n = 2^s$ with $s > 1$ then the Galois group must have an element of order 4 (G can not be C_2^s since the number of subfields of degree index 2 is not $n-1$). This element of order 4 corresponds to a linear factor f_i of f in $K[x]$. Let L_i be its corresponding principal subfield. Then L_i is contained in m maximal subfields where m is either 1 or 3. Let $\check{f}_i$ be the minimal polynomial of α over L_i. If $m = 3$ then every irreducible factor of $\check{f}_i/(x-\alpha)$ corresponds to a subfield of degree index 2. This is a contradiction since f_i divides $\check{f}_i/(x-\alpha)$. $\square$

3. The number field case

3.1. Introduction

In this section we describe an algorithm for producing a generating set when $K = \mathbb{Q}(\alpha)$. Factoring f over K, though polynomial time, is slow, thus we prefer to use an approximation of a p-adic factorization and LLL. We show that when the algorithm terminates [1], it returns the correct output.

For a prime number p, let $\mathbb{Q}_p$ denote the field of p-adic numbers, $\mathbb{Z}_p$ the ring of p-adic integers, and $\mathbf{F}_p = \mathbb{Z}/(p)$. We choose a prime number p with these three properties: p does not divide the leading coefficient of $f \in \mathbb{Z}[x]$, the image $\overline{f}$ of f in $\mathbf{F}_p[x]$ is separable, and has at least one linear factor which we denote $\overline{f}_1$ (asymptotically, the probability that a randomly chosen prime p has these properties is $\geq 1/n$, where equality holds when K/k is Galois).

By factoring $\overline{f}$ in $\mathbf{F}_p[x]$ and applying Hensel lifting, we obtain a factorization of $f = f_1 \cdots f_r$ over $\mathbb{Q}_p$ where f_1 has degree 1. By mapping $\alpha \in K$ to the root α_1 of f_1 in $\mathbb{Q}_p$ we obtain an embedding $K \to \mathbb{Q}_p$, and so we can view K as a subfield of $\tilde{K} := \mathbb{Q}_p$.

[1] a bound for the running time can be obtained in a similar way as in [3]

The advantage of taking $\mathbb{Q}_p$ (instead of K) for $\tilde{K}$ is that it saves time on factoring f over $\tilde{K}$. Since p does not divide the denominators of the coefficients of f, the factors $f_1, \ldots, f_r$ of f over $\mathbb{Q}_p$ lie in $\mathbb{Z}_p[x]$. We can not compute these factors with infinite accuracy, but only to some finite accuracy a, meaning that $f_1, \ldots, f_r$ are only known modulo p^a.

For each of the factors, f_i, we will need to find the principal subfield L_i which was defined in Section 2.1 as the kernel of $\phi_i - \mathrm{id}$. To do this we will make use of a knapsack-style lattice in the style of [19]. To get the best performance we would like to design a lattice such that boundably short vectors correspond to elements in L_i.

A natural approach would be to use $1, \alpha, \ldots, \alpha^{n-1}$ as a basis, and search for linear combinations whose images under $\phi_i - \mathrm{id}$ are 0 (mod p^a). However, we will use a different basis. Denote $\mathbb{Z}[\alpha]_{<n} := \mathbb{Z}\cdot\alpha^0 + \cdots + \mathbb{Z}\cdot\alpha^{n-1}$ (note: if f is monic then this is simply $\mathbb{Z}[\alpha]$ but we do not assume that f is monic). Then the basis $\frac{1}{f'(\alpha)}, \ldots, \frac{\alpha^{n-1}}{f'(\alpha)}$ of $\frac{1}{f'(\alpha)} \cdot \mathbb{Z}[\alpha]_{<n}$ allows us to prove more practical bounds (this phenomena has also been observed in other contexts [6]). Using this basis of K we prove the existence of a $\mathbb{Q}$-basis of L_i which has a bounded representation. We delay the proof of this theorem until section 3.5.

Theorem 12. *Let L_i, the target principal subfield, have degree m_i over $\mathbb{Q}$. For $\beta \in \frac{1}{f'(\alpha)} \cdot \mathbb{Z}[\alpha]_{<n}$ with $\beta = \sum b_i \frac{\alpha^i}{f'(\alpha)}$ we associate the vector $\mathbf{v}_\beta := (b_0, \ldots, b_{n-1})$. Then there exists m_i linearly independent algebraic numbers $\beta_1, \ldots \beta_{m_i} \in L_i \cap \frac{1}{f'(\alpha)} \cdot \mathbb{Z}[\alpha]_{<n}$ each with $\|\mathbf{v}_{\beta_k}\| \leq n^2\|f\|_2$.*

3.2. The computation of a principal subfield

Now we can continue the description of the computation of the principal subfield L_i corresponding to the factor f_i of degree d_i. As mentioned before we will represent our elements in the basis $\frac{1}{f'(\alpha)}, \ldots, \frac{\alpha^{n-1}}{f'(\alpha)}$. Each of these basis elements will be represented as the column of an identity matrix to which we attach entries for the image of that basis element under $\phi_i - \mathrm{id}$. Since these images are only known modulo p^a we must also adjoin columns which allow for this modular reduction. Suppose the degree of f_i is d_i, then our lattice is spanned by the columns of the following $(n + d_i) \times (n + d_i)$ integer matrix:

$$B_i := \begin{pmatrix} 1 & & & & \\ & \ddots & & & \\ & & 1 & & \\ c_{0,0} & \cdots & c_{0,n-1} & p^a & \\ \vdots & \ddots & \vdots & & \ddots \\ c_{d_i-1,0} & \cdots & c_{d_i-1,n-1} & & & p^a \end{pmatrix} \qquad (1)$$

where $c_{k,j}$ is the k^{th} coefficient of $\frac{x^j}{f'(x)} \mathrm{mod} f_i - \frac{x^j}{f'(x)} \mathrm{mod} f_1$ reduced modulo p^a. To interpret a vector $\mathbf{v}$ in the column space of this matrix we take the first n entries $b_0, \ldots, b_{n-1}$ and then compute $(\sum b_j \alpha^j)/f'(\alpha)$. A vector corresponding to an element in L_i will have its final d_i entries be 0 modulo p^a. Thus Theorem 12 shows us that the lattice generated by columns of B_i contains a dimension m_i sublattice which has a small basis. This allows us to use the new sub-lattice reduction techniques of [19] on B_i. Thus, rather than standard LLL, we use `LLL_with_removals` which performs lattice reduction but removes any vectors in the final position whose G-S norm is above a given bound. The following lemma is derived from [15] and justifies these removals.

Lemma 13. Given a basis $\mathbf{b}_1, \ldots, \mathbf{b}_d$ of a lattice Λ, and let $\mathbf{b}_1^*, \ldots, \mathbf{b}_d^*$ be the output of Gram-Schmidt orthogonalization. If $\| \mathbf{b}_d^* \| > B$ then any vector in L with norm $\leq B$ is a $\mathbb{Z}$-linear combination of $\mathbf{b}_1, \ldots, \mathbf{b}_{d-1}$.

This technique is common and is used in [10; 19]. As the removal condition requires Gram-Schmidt norms we can state that LLL reduced bases tend to be numerically stable for Gram-Schmidt computations so a floating point Gram-Schmidt computation could be used for efficiency (see [20]). Also FLINT 1.6 [9] has an LLL with removals routine which takes a bound and returns the dimension of the appropriate sub-lattice.

In this way using `LLL_with_removals` with the bound from Theorem 12 will allow us to reduce the dimension. In Figure 4 we give a practical algorithm which will create a basis of a subfield of K which is highly likely to be L_i. We will use $D := \mathrm{diag}\{1, \ldots, 1, C, \ldots, C\}$ as a matrix for scaling the last d_i rows of B_i by a scalar C. Since the vectors guaranteed by Theorem 12 come from L_i we know that the final d_i entries must be 0. Thus multiplication on the left by D and removals will eventually ensure that vectors with zero entries are found by LLL.

Input: f_i
Output: h_k which probably generate L_i

1. Create lattice B_i from equation ((1))
2. $A :=$ `LLL_with_removals`$(B_i, n^2 \parallel f \parallel)$
3. $m := \dim(A)$
4. while $\exists l > n, j$ such that $A[l, j] \neq 0$:
5. $\quad A := D \cdot A$
6. $\quad A :=$ `LLL_with_removals`$(A, n^2 \parallel f \parallel)$
7. $\quad m := \dim(A)$
8. if $m \nmid n$ increase precision repeat `principal`
9. for $1 \leq k \leq m$:
10. $\quad h_k := \sum_{j=1}^{n} \frac{A[j,k]x^{j-1}}{f'(x)}$

Fig. 4. `principal` algorithm

Using LLL on the matrix entire B_i will suffice for this paper. However, in practice the d_i final rows of B_i can also be reduced one at a time. In this way one could potentially arrive at a solution without needing all rows of B_i. Such an approach is seen in [19] and could be adapted to this situation.

The algorithm in figure 4 will produce m p-adic polynomials h_k, which are likely to correspond to algebraic numbers which generate L_i as a $\mathbb{Q}$-vector space. It is possible that m is not m_i but some other divisor of n. In particular, if the p-adic precision is not high enough then there could be entries in the lattice basis which are 0 modulo p^a but not exactly 0. In that case one of the h_k would not be from L_i. Even so the

Preliminary version – 3 June 2012

$\mathbb{Q}$-vector space generated by the h_k must at least contain L_i. The reason is that at least m_i linearly independent algebraic numbers from L_i remain within the lattice after `LLL_with_removals` thanks to the bound of Theorem 12 and Lemma 13.

Theorem 12 can also be used to make a guess for a starting precision of p^a. Since any reduced basis has Gram-Schmidt norms within a factor 2^{n+d_i} of the successive minima and the determinant of B_i is $p^{a \cdot d_i}$ then we should ensure that $p^{a \cdot d_i}$ is at least $(2^{n+d_i} n^2 \|f\|)^n$.

3.3. Confirming a principal subfield

In this section we will assume that we have elements which are likely to generate a principal subfield (in other words, the output of the algorithm in Figure 4). At this point it seems reasonable to discuss the possible paths forward. This must include a discussion of the types of output that a user might want. We recommend outputting the subfield polynomial represented in the $\alpha^i/f'(\alpha)$ basis. This has the advantage of certifying that the elements we have indeed generate the target L_i. In addition it gives us a representation of L_i which can be stored on a relatively small number of bits.

It may also seem reasonable to ask for a primitive element of L_i perhaps given as the root of some minimal polynomial with coefficients in $\mathbb{Z}$. The coefficients of the subfield polynomial are a good source of potential primitive elements which will have small minimal polynomials. After all, the coefficients of the subfield polynomial must generate L_i. It might also be likely that such a minimal polynomial could be much larger than our suggested representation of the subfield polynomial. For these reasons we will deal primarily with finding the subfield polynomial, we do this in section 3.3.2.

Before that we treat the option of resuming the algorithm using the block methods of [12; 13]. This makes some sense as the combinatorial explosion in that method might already have been bypassed. This approach is discussed in section 3.3.1. The output of that algorithm is a primitive element of the L_i.

Then in section 3.4 we will give an illustrative example of the algorithm in action so as to clarify the procedure. Finally in section 3.5 we prove the main technical theorem which allowed us to provably bound the output of Figure 4.

3.3.1. Using block systems to confirm the subfield

In this section we show the connection with what we have computed so far and the block systems approach of [12; 13]. We can use any of the non integral algebraic numbers output by figure 4 to generate block systems if we would like to avoid doing more LLL reduction. We try to combine the advantages of both methods. The big problem of the method presented in [12; 13] is that we have to consider exponentially many possibilities of potential block systems in the worst case. On the other hand this method is very efficient as soon as we have found the right block system. After the computation done in Figure 4 we get elements $h_1, \ldots, h_m$ and we are almost certain that these elements generate our principal subfield. More precisely we expect that they form a vector space basis of our prinicipal subfield L_i. In order to be sure we need a proof for this statement. Furthermore we would like to find a nice presentation of our subfield. Knowing the elements $h_1, \ldots, h_m$ it is easy to write down the corresponding block system. Having the actual block system in our hand we can apply the methods described in [12; 13] without having the combinatorial explosion.

Before we explain this approach let us give a criterion which gives a check if a given subfield L is equal to the principal subfield L_i.

Lemma 14. Let $L = \mathbb{Q}(\beta)$ be a subfield of K. Let $\beta = g(\alpha)$, where $g(x) \in \mathbb{Q}[x]$ is a polynomial of degree smaller than n. As before denote by $f = f_1 \cdots f_r \in \mathbb{Q}_p[x]$ the factorization of f into irreducible factors over $\mathbb{Q}_p$. Define $T := \{1 \leq i \leq r \mid g(x) \equiv g(\alpha) \bmod f_i\}$. Then the subfield polynomial $f_L = \prod_{i \in T} f_i$.

The proof of this lemma follows easily from the discussion before Theorem 1. Now L is a subfield of L_i if and only if $i \in T$. From the computation in Figure 4 we know that L_i has at most degree m. This means that our field $L = L_i$, when we know that $L \leq L_i$ and the degree of L is m.

One approach could be to compute the minimal polynomials of the elements h_i hoping to quickly find a primitive element (of degree m). Then we test if $L \leq L_i$ by using Lemma 14. We remark that the test in Lemma 14 can be done modulo p^k for a small k (in most cases $k = 1$). We can increase k until we get that the degree of L times the product of the degrees of the f_i with $i \in T$ equals n. In general the elements h_i are non-integral elements and their minimal polynomials are not nice at all. If we look at our computation it is not necessary to compute the minimal polynomials. In order to identify the right set T we can use the identity:

$$T = \{1 \leq i \leq r \mid \forall 1 \leq j \leq m : g_j(x) \equiv g_j(\alpha) \bmod f_i\},$$

where $h_j = g_j(\alpha)$.

Now we explain how to compute the corresponding (potential) block system which can be used by the method described in [13]. For this we use the notation of this paper. Let $\alpha_1, \ldots, \alpha_n$ be the roots of f in some unramified p-adic extension of $\mathbb{Q}_p$. Let $\beta = h(\alpha)$ be a primitive element of the subfield L of degree m, where $h \in \mathbb{Q}[x]$ is a polynomial of degree less than n. Furthermore we denote by $\beta_1, \ldots, \beta_m$ the roots of g in the same p-adic extension. Then the corresponding block system is given by Lemma 3.21 in [13] via

$$\Delta_i := \{\alpha_j \mid h(\alpha_j) = \beta_i, 1 \leq j \leq n\}.$$

Now enter the subfield algorithm in [13] using this potential block system. If this algorithm succeeds in computing a subfield L, then test if $L = L_i$ using Lemma 14. Note (see equation (12) in [13]) that this algorithm computes the element $\delta_1 = \prod_{\alpha \in \Delta_1} \alpha$ as a first guess of a primitive element of our subfield. If this element fails to generate our subfield then elements of the form $\prod_{\alpha \in \Delta_1} (\alpha + k)$ for some $k \in \mathbb{Z}$ are chosen. Note that δ_1 is, up to the sign, the absolute coefficient of the subfield polynomial f_L. It is easy to adapt the algorithm described in [13] to use other coefficients of f_L. In the case that $\sum_{\alpha \in \Delta_1} \alpha$ is a primitive element, this usually gives generators of small size.

3.3.2. Finding a small representation of the subfield polynomial using LLL

We give an algorithm which will construct the subfield polynomial g, of L_i or return failure, in which case more p-adic precision is needed. We choose the subfield polynomial as it will provide a proof that we have a principal subfield and can be stored in a relatively compact way thanks to our new basis. Of course other representations and proofs are possible.

From here on our algorithmic objective will be to output the minimal polynomial $g \in L_i[x]$ of α over L_i. This g is the subfield polynomial of L_i and its coefficients generate L_i. We know m elements h_k modulo p^a, we know that $m|n$ and that $(\phi_i - \mathrm{id})(h_k) \equiv 0$ modulo p^a for each k. Recall that the h_k were from columns of a lattice basis A. First we will create a p-adic candidate subfield polynomial which we then subject to 3 certification checks.

Candidate g: Create an index set $T := \{j | \phi_j(h_k) \equiv \mathrm{id}(h_k) \mod p^a \forall h_k\}$, that is find the p-adic factors of f which also agree with f_1 on the elements corresponding to the basis from A. T will contain at least 1 and i. Now let $g_{\text{cand}} := \prod_{j \in T} f_j \mod p^a$. This is done in steps 1–5 of Figure 5

Input: $h_1 \dots h_m, f_1, \dots f_r \in \mathbb{Q}_p[x]$, precision a
Output: g subfield poly, or `fail`

1. $T := \{\}$
2. for each $1 \leq j \leq r$:
3.   if $(h_k \mod f_j = h_k \mod f_1) \mod p^a \forall k$ then:
4. $T := T \cup j$
5. $g_{\text{cand}} := \mathrm{lc}(f) \cdot \prod_{j \in T} f_j \mod p^a$
   where $\mathrm{lc}(f)$ is the leading coefficient of f
6. Create lattice M using (2)
7. $M :=$LLL(M)
8. $g_{\text{temp}} = 0$
9. for each coefficient g_k of x^k in g_{cand}:
10.   create M_{g_k} lattice using (3)
11.   **Check 1** find $\mathbf{v}$ in LLL(M_{g_k})
    with $\mathbf{v}[n+1] = 0$ and $\mathbf{v}[n+2] = 1$
12. $g_{\text{temp}} := g_{\text{temp}} + \sum_{j=1}^{n} \frac{\mathbf{v}[j]\alpha^{j-1}}{f'(\alpha)} x^k$
13. $g_{\text{cand}} := g_{\text{temp}} \in \mathbb{Q}(\alpha)[x]$
14. **Check 2** ensure $g_{\text{cand}} | f$ exactly
15. **Check 3** ensure $(h_k \mod g_{\text{cand}} = h_k \mod f_1)\ \forall k$
16. return $g := g_{\text{cand}}$

Fig. 5. `final_check` algorithm

Check 1: Let $\Lambda(A) \subseteq \frac{\mathbb{Z}[\alpha]_{<n}}{f'(\alpha)}$ be the lattice generated by the algebraic numbers corresponding with columns of A. We now attempt to find an exact representation of g_{cand} by converting each coefficient into an algebraic number in $\Lambda(A) \cap \frac{\mathbb{Z}[\alpha]_{<n}}{f'(\alpha)}$. We'll do this by attempting to find linear combinations of h_k which exactly equal each coefficient of g_{cand}.

Note that this g_{cand} is a polynomial with p-adic coefficients, these coefficients can be quickly Hensel lifted using the fact that $f = g \cdot (f/g) \bmod p^a$ if more precision is needed. Now we want to express these coefficients in the basis $\frac{\mathbb{Z}[\alpha]_{<n}}{f'(\alpha)} \cap \Lambda(A)$. To do this we will use a lattice basis similar to A with a slight adjustment. Rather than finding algebraic numbers whose images under $\phi_i - \mathrm{id}$ are zero, we'll find combinations of the h_k whose p-adic valuations match a coefficient of g_{cand}.

Lets call $\mathbf{v}_{h_k}$ the coefficient vector of h_k, and the corresponding p-adic valuation $c_j := h_k(\alpha_1)$ (that is, h_k modulo f_1). Also we pick a large scalar constant C (to ensure that LLL works on reducing the size of the p-adic row). We let the columns of the new matrix be $(\mathbf{v}_{h_j}, C \cdot c_j)^T$, and the column $(0, \dots, 0, C \cdot p^a)$.

$$M := \begin{pmatrix} \mathbf{v}_{h_1}^T & \dots & \mathbf{v}_{h_m}^T & \mathbf{0} \\ C \cdot c_1 & \dots & C \cdot c_m & C \cdot p^a \end{pmatrix} \tag{2}$$

A vector in the column space of this matrix is a representation of a combination of the elements from h_k along with a p-adic valuation of that element. Now for each coefficient we'll use this matrix to find a combination which matches that coefficient. In practice we LLL-reduce M before adjoining data from the coefficients of g_{cand}, but here we present an augmented M without altering the columns first (for clarity).

For each coefficient g_k of g_{cand} augment each column of M with a zero, then adjoin a new column $(0, \dots, 0, C \cdot g_k, 1)^T$. This is what the coefficient matching matrix looks like:

$$M := \begin{pmatrix} \mathbf{v}_{h_1}^T & \dots & \mathbf{v}_{h_m}^T & \mathbf{0} & \mathbf{0} \\ C \cdot c_1 & \dots & C \cdot c_m & C \cdot p^a & C \cdot g_k \\ 0 & \dots & 0 & 0 & 1 \end{pmatrix} \tag{3}$$

Run LLL on this matrix (provided C is large enough) then find the vector which has its final two entries as 0,1, the first n entries are an expression of g_k in $\frac{\mathbb{Z}[\alpha]_{<n}}{f'(\alpha)}$. If this works for every coefficient of g_{cand} then the check has passed.

Check 2: Ensure that $g_{\text{cand}} | f$ in $\mathbb{Q}(\alpha)[x]$.

Check 3: Ensure that $h_k \mod g_{\text{cand}} = h_k \mod f_1$ for each h_k.

Theorem 15. *If all checks pass then the $\mathbb{Q}$-linear combination of the elements corresponding to the lattice basis A generate L_i the target principal subfield, and g_{cand} is the subfield polynomial of L_i.*

Proof. By construction of g_{cand} and A we know that the span over $\mathbb{Q}$ of the elements corresponding to A, the h_k, contains L_i. Let's call this span V, so $L_i \subseteq V$. Since g_{cand} divides f and f_i divides g_{cand} then $h \mod g_{\text{cand}} = h \mod f_1$ implies $h \mod f_i = h \mod f_1$. By check 1 this implies that $V \subseteq L_i$ thus the span over $\mathbb{Q}$ of the elements from the lattice is L_i.

Now $x - \alpha, f_i | g_{\text{cand}} \mod p^a$ and $g_{\text{cand}} | f$ exactly then $f_i | g_{\text{cand}}$ and $(x - \alpha) | g_{\text{cand}}$ exactly. Now by Remark 6 we know g_{cand} is the subfield polynomial of L_i. □

If any check fails increase the p-adic precision via Hensel lifting and try again.

3.4. An illustrative example

Here we provide an example from a potential application of the algorithm. Suppose that one is searching for solutions to the system of equations

$$a^2 - 2ab + b^2 - 8 = 0$$
$$a^2b^2 - (a^2 + 2a + 5)b + a^3 - 3a + 3 = 0.$$

Using MAPLE's [16] `solve` command the output is:

$$\alpha = \texttt{RootOf}(x^8 - 20x^6 + 16x^5 + 98x^4 + 32x^3 - 12x^2 - 208x - 191)$$
$$a = \alpha$$
$$b = -34\alpha^7 + 61\alpha^6 + 742\alpha^5 - 1757\alpha^4 - 3378\alpha^3 + 6013\alpha^2 + 6368\alpha + 7175.$$

By writing α in terms of generators of proper subfields of $\mathbb{Q}(\alpha)$ we can greatly simplify the expression to:

$$a = \sqrt{3} + \sqrt[4]{2} - \sqrt{2}$$
$$b = \sqrt{3} + \sqrt[4]{2} + \sqrt{2}.$$

This illustrates that computing subfields is an important step toward simplifying algebraic expressions.

Here we will give the various stages of the algorithm's output using the minpoly of α, $f = x^8 - 20x^6 + 16x^5 + 98x^4 + 32x^3 - 12x^2 - 208x - 191$, as our input.

The first step is to find a prime such that f is squarefree modulo p and has at least one linear factor. The first acceptable prime is 23 and the factorization of f mod 23 is:

$$f \equiv (x+3)(x-4)(x+10)(x-6)(x^2+x+1)(x^2-4x-1).$$

Now we must Hensel lift this factorization so that short vectors are not likely to be because of a small modulus. To decide a target p-adic precision we refer to Theorem 12 which asserts that we are looking for vectors of norm $\leq n^2 \parallel f \parallel_2 = 64 \cdot 302$. One could begin the algorithm with p^a just above this bound and resume hensel lifting in the case of failure; or one could begin well above the bound to minimize the chances of early failure. In this particular case a modulus of p^{25} is always sufficient for solving the problem.

We will let the first linear local factor of f be labeled f_1 (in this case the factor whose image is $x+3$ modulo 23). Recall that f_1 is defined to be $x - \alpha$ so that the principal subfield $L_1 := \{g(\alpha) \in \mathbb{Q}(\alpha) | g(x) \equiv g(\alpha) \mod f_1\}$ is simply $\mathbb{Q}(\alpha)$. Thus L_2 is potentially the first non-trivial principal subfield, where f_2 is the next factor (in this case the p-adic factor whose image mod 23 is equivalent to $x-4$).

Now we must construct the lattice from equation (1) whose columns correspond with a basis of $\frac{1}{f'(\alpha)}\mathbb{Z}_{\leq n}[\alpha]$. Specifically column i will be e_i (the standard basis vector) augmented with $\left(\frac{x^{i-1}}{f'(x)} \mod f_2\right) - \left(\frac{x^{i-1}}{f'(x)} \mod f_1\right)$, thus any element in L_2 will have 0 as the final entry. In the implementation we compute $\frac{1}{f'(x)} \mod f$ as an integer polynomial with a single denominator at the beginning of the procedure and use its image modulo the local factors in the various stages when it is needed.

For illustration we will show the lattice with low p-adic precision, so that the reader can easily confirm the construction. In this case $\frac{1}{f'(\alpha)} \mod \langle x+3, 23\rangle \equiv 3$ and $\frac{1}{f'(\alpha)} \mod \langle x-4, 23\rangle \equiv 22$. So $(\frac{x^0}{f'(x)} \mod f_2) - (\frac{x^0}{f'(x)} \mod f_1)$ is 19. Repeat the process for the other powers of x, $\left(\frac{x^{i-1}}{f'(x)} \mod f_2\right) - \left(\frac{x^{i-1}}{f'(x)} \mod f_1\right)$, to get the lattice from equation (1) with 23-adic precision 1 (i.e. $\mod 23$):

$$B_2^T := \begin{pmatrix} 1 & & & & & & & & 19 \\ & 1 & & & & & & & 5 \\ & & 1 & & & & & & 3 \\ & & & 1 & & & & & 17 \\ & & & & 1 & & & & 7 \\ & & & & & 1 & & & 4 \\ & & & & & & 1 & & 19 \\ & & & & & & & 1 & 21 \\ & & & & & & & & 23 \end{pmatrix}.$$

Now this matrix does not have large enough p-adic entries to get valuable information out of an LLL run. However LLL on B_2 with higher p-adic precision will yield 4 vectors of small norm and 5 vectors of large norm. When the precision is at least 23^{25} then the G-S norms of the 5 final vectors will be large enough (more than $64 \cdot 302$) to prove, via Lemma 13, that the span of 4 short vectors must contain the basis of L_2 guaranteed by Theorem 12. In this case the four short vectors are the transpose of:

$$\begin{pmatrix} 7 & 6 & 2 & -20 & -3 & 2 & 0 & 0 & 0 \\ -18 & 12 & 1 & 5 & 8 & 10 & -1 & -1 & 0 \\ 5 & -15 & -18 & 11 & -1 & 9 & 0 & -1 & 0 \\ -15 & -35 & 3 & -23 & 9 & -7 & -1 & 1 & 0 \end{pmatrix}.$$

These four vectors represent a potential basis of $L_2 \cap \frac{1}{f'(\alpha)}\mathbb{Z}_{\leq n}[\alpha]$. The fact that 4 divides 8 is a simple first check that we have a potential subfield. Next the fact that each of the last entries is 0 is a check that we might be looking at vectors inside of L_2 (the worst case is that two or four of these vectors happen to have last entry with an image of $0 \mod 23^{25}$ but this would not be exactly 0 at infinite precision). From here, there are several paths we could take, namely: compute the subfield polynomial of L_2, that is the minimal polynomial of α over L_2 (proving that we really have L_2) or compute a primitive element of L_2 and prove that what we have is actually L_2 in some other way. See the discussion in section 3.3. Here we will compute the subfield polynomial.

The subfield polynomial must be the product of some subset of the p-adic factors of f. We wish to find all factors which make up the subfield polynomial for L_2. We do this by checking which other p-adic factors of f agree with f_1 on the four given elements. For example to check the first vector one computes $\frac{(7+6x+2x^2-20x^3-3x^4+2x^5)}{f'(x)}$ modulo $\langle f_i, 23^{25}\rangle$ for all i. Then any f_i which give the same output as f_1 will be considered to agree on the first element. In this case, none of the other factors agree with f_1 and f_2 on all 4 vectors (although f_5 agrees on two of the four elements).

So we now assume that the subfield polynomial is $f_1 \cdot f_2$ until we can prove otherwise. Since we have approximations of those factors to precision 23^{25} we compute the candidate for the subfield poly $g = f_1 \cdot f_2 \mod 23^{25}$, this is g_{cand} in figure 5. Now we have p-adic numbers g_i for each coefficient in $g_{\text{cand}} = g_0 + g_1 x + g_2 x^2$. The lattice from equation (3) will help us find a representation of the g_i in terms of α. This works by attempting to find a linear combination of the short vectors which has the same p-adic image as one of the coefficients of g_{cand}. So if we let v_i be the p-adic images of the four short vectors in our example we could use a lattice like the transpose of this one for finding g_0:

$$\begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 23^{25} & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & g_0 & 1 \\ 7 & 6 & 2 & -20 & -3 & 2 & 0 & 0 & v_1 & 0 \\ -18 & 12 & 1 & 5 & 8 & 10 & -1 & -1 & v_2 & 0 \\ 5 & -15 & -18 & 11 & -1 & 9 & 0 & -1 & v_3 & 0 \\ -15 & -35 & 3 & -23 & 9 & -7 & -1 & 1 & v_4 & 0 \end{pmatrix}.$$

In this case we also suggest scaling the column containing p^a, v_j, and g_i by some large constant (in the implementation we used 2^{40}), so that LLL is more likely to find a vector which ends with 0 and 1. This particular lattice yields such a vector, $(0, -24, -368, -136, 32, 424, -16, -40, 0, 1)$. We interpret this to say that

$$g_0 = \frac{(-24\alpha - 368\alpha^2 - 136\alpha^3 + 32\alpha^4 + 424\alpha^5 - 16\alpha^6 - 40\alpha^7)}{f'(\alpha)}.$$

By constructing the same lattice for g_1 and g_2 we can get a representation of g_{cand} in $L_2[x]$ which uses the $\alpha^i/f'(\alpha)$ basis. That representation could be encoded in the transpose of the following:

$$\begin{pmatrix} 0 & -24 & -368 & -136 & 32 & 424 & -16 & -40 \\ 1552 & 1824 & 208 & -192 & -816 & -32 & 80 & 0 \\ 208 & 24 & -96 & -392 & -80 & 120 & 0 & -8 \end{pmatrix}.$$

Note that the final row is actually the coefficients of $f'(\alpha)$ so this is a monic polynomial. In general the coefficients of g, the subfield polynomial, will have much smaller minimal polynomials than the elements from the short vectors. If one needs to compute a primitive element of L_2 then we suggest taking coefficients of g and testing if they are primitive elements. For instance g_1 and g_0 have minimal polynomials of degree 4, so either will generate L_2 because g has degree 2 and $[\mathbb{Q}(\alpha) : \mathbb{Q}] = 8$. In this case the minimal polynomial of g_1, corresponding to the second row above, is $x^4 - 40x^2 + 16$. If this fails then try small combinations of the coefficients.

3.5. Bounds for the coefficients

The only aim of this section is to prove Theorem 12. The techniques described in this section are not used in the algorithm.

In order to get our desired bounds it is useful to introduce the notation of a codifferent, see [17, Chapter 4.2] for more details.

Lemma 16. Let $f \in \mathbb{Z}[x]$ be primitive and irreducible, with degree n. Let α be a root of f. Let $\mathcal{O}_K$ be the ring of integers in $K = \mathbb{Q}(\alpha)$ and let $\mathcal{O}_K^*$ be the *co-different* which is defined as:

$$\mathcal{O}_K^* = \{a \in K | \forall_{b \in O_K} \operatorname{Tr}(ab) \in \mathbb{Z}\}.$$

Then

$$\mathcal{O}_K^* \subseteq \frac{1}{f'(\alpha)} \mathbb{Z}[\alpha]_{<n}. \tag{4}$$

Proof. Let $a \in \mathcal{O}_K^*$, so $\operatorname{Tr}(ab) \in \mathbb{Z}$ for any $b \in \mathcal{O}_K$. The content of a polynomial $g = c_0 x^0 + \cdots + c_d x^d \in K[x]$ is defined as the fractional ideal $c(g) = \mathcal{O}_K c_0 + \cdots + \mathcal{O}_K c_d$. Let $g_1 = x - \alpha$ and $g_2 = f/g_1$. Gauss' lemma says $c(g_1)c(g_2) = c(g_1 g_2)$. Then $c(g_1)c(g_2) = c(f) = \mathcal{O}_K$, ($f$ is primitive) and since g_1 has a coefficient equal to 1 it follows that $c(g_2) \subseteq \mathcal{O}_K$, in other words $g_2 \in O_K[x]$. Now $ag_2 \in a \cdot \mathcal{O}_K[x]_{<n}$ and by definition of $\mathcal{O}_K^*$ we see that $\operatorname{Tr}(ag_2) \in \mathbb{Z}[x]_{<n}$. So

$$\operatorname{Tr}(a \frac{f(x)}{x - \alpha}) = \sum a^{(i)} \frac{f(x)}{x - \alpha^{(i)}} \in \mathbb{Z}[x]_{<n}$$

where $a^{(i)}$ and $\alpha^{(i)}$ denote the conjugates of a and α. Evaluating the right-hand side at $x = \alpha = \alpha^{(1)}$ gives $af'(\alpha) \in \mathbb{Z}[\alpha]_{<n}$ and hence $a \in 1/f'(\alpha) \cdot \mathbb{Z}[\alpha]_{<n}$. $\square$

Now suppose that we have a $\beta \in \mathcal{O}_K^*$, then we can write

$$f'(\alpha)\beta = \sum_{i=0}^{n-1} b_i \alpha^i \text{ with } b_i \in \mathbb{Z}. \tag{5}$$

In our applications β is an element of a principal subfield and we would like to bound the size of b_i. In the following we need the complex embeddings and some norms of algebraic numbers.

Definition 17. Let $K = \mathbb{Q}(\alpha)$ be a number field of degree n and f be the minimal polynomial of α. Then we denote by $\phi_1, \ldots, \phi_n : K \to \mathbb{C}, \alpha \mapsto \alpha_i$ the n complex embeddings, where $\alpha_1, \ldots, \alpha_n$ are the complex roots of f. We assume that $\alpha_1, \ldots, \alpha_{r_1}$ are real and the complex roots are ordered such that $\alpha_{r_1+i} = \bar{\alpha}_{r_1+r_2+i}$ for $1 \leq i \leq r_2$.

For $\beta \in K$ we define the norms

$$\|\beta\|_1 := \sum_{i=1}^{n} |\phi_i(\beta)| \text{ and } \|\beta\|_2 := \sqrt{\sum_{i=1}^{n} |\phi_i(\beta)|^2}.$$

Note the well known estimates:

$$\|\beta\|_2 \leq \|\beta\|_1 \leq \sqrt{n}\|\beta\|_2.$$

We are able to give the promised bounds.

Lemma 18. Let β be given as in (5) with coefficient vector $b := (b_0, \ldots, b_{n-1})$. Then we have $\|b\|_2 \leq n\|\beta\|_1\|f\|_2 \leq n^{1.5}\|\beta\|_2\|f\|_2$.

Proof. Let $h(x) := \sum_{i=0}^{n-1} b_i x^i$. Let $\alpha_i := \phi_i(\alpha)$ and $\beta_i := \phi_i(\beta)$, then we get: $h(\alpha_i) = \beta_i f'(\alpha_i)$ for $1 \leq i \leq n$. Using Lagrange interpolation we get:

$$h(x) = \sum_{i=1}^{n} \beta_i f'(\alpha_i) \frac{f(x)/(x-\alpha_i)}{f'(\alpha_i)} = \sum_{i=1}^{n} \beta_i \frac{f(x)}{x-\alpha_i}.$$

Now:

$$\|b\|_2 = \|h\|_2 = \sum_{i=1}^{n} |\beta_i| \|f/(x-\alpha_i)\|_2$$
$$\leq \max_i \|f/(x-\alpha_i)\|_2 \sum_{i=1}^{n} |\beta_i| \leq n\|f\|_2\|\beta\|_1,$$

$\|f/(x-\alpha_i)\|_2 \leq n\|f\|_2$ is proved in [18, cor4.7]. The second estimate follows then trivially from $\|\cdot\|_1 \leq \sqrt{n}\|\cdot\|_2$. □

Now our goal is the following. Let L be a principal subfield of degree m which we would like to compute. We want to find a $\mathbb{Q}$-basis of L represented in our $\frac{1}{f'(\alpha)}\mathbb{Z}[\alpha]_{<n}$–basis. Note that $\mathcal{O}_L^* \subseteq \mathcal{O}_K^* \subseteq \frac{1}{f'(\alpha)} \cdot \mathbb{Z}[\alpha]_{<n}$. In order to apply Lemma 18 we need to bound $\|\beta_i\|_2$ for m linearly independent elements $\beta_1, \ldots, \beta_m \in L$. We will use the following theorem.

Theorem 19 (Banaszczyk)**.** *Let $\Lambda \subset \mathbb{R}^m$ be a lattice and denote by $\Lambda^* := \{y \in \mathbb{R}^m \mid \forall x \in \Lambda : \langle x, y\rangle \in \mathbb{Z}\}$ the dual lattice. Furthermore denote by λ_i, λ_i^* the i-th successive minima of Λ, Λ^*, respectively. Then $\lambda_i \lambda_{m+1-i}^* \leq m$ for $1 \leq i \leq m$.*

The proof can be found in [1, Theorem 2.1]. In our application $\lambda_1 = \sqrt{m}$, so we get the upper bound $\lambda_m^* \leq \sqrt{m}$. There are canonical ways to map number fields to lattices, but we have the slight problem that the bilinear form $L \times L \to \mathbb{Q}, (x, y) \mapsto \mathrm{Tr}(xy)$ is not positive definite, if L has non-real embeddings. We assume the same order of the complex embeddings of L as in Definition 17, so we have $m = r_1 + 2r_2$. Defining $\gamma_i = \phi_i(\gamma)$ and $\delta_i = \phi_i(\delta)$ we get:

$$\mathrm{Tr}(\gamma\delta) = \sum_{i=1}^{m} \gamma_i \delta_i.$$

The corresponding scalar product looks like:

$$\langle \gamma, \delta \rangle := \sum_{i=1}^{m} \gamma_i \bar{\delta}_i.$$

For totally real number fields L those two notions coincide. The dual lattice equals $\mathcal{O}_L^*$ and we can apply Theorem 19 directly to get the desired bounds. First we introduce the canonical real lattice $\Lambda := \Psi(\mathcal{O}_L) \subseteq \mathbb{R}^m$ associated to $\langle \gamma, \delta \rangle$ via

$$\Psi : L \to \mathbb{R}^m, \tag{6}$$

$$\beta \mapsto (\beta_1, \ldots, \beta_{r_1}, \sqrt{2}\Re(\beta_{r_1+1}), \ldots, \sqrt{2}\Re(\beta_{r_1+r_2}),$$
$$\sqrt{2}\Im(\beta_{r_1+1}), \ldots, \sqrt{2}\Im(\beta_{r_1+r_2})).$$

Note that now the standard scalar product of $\mathbb{R}^m$ coincides with the (complex) scalar product defined above. This is the reason for the weight $\sqrt{2}$ in the above definition. Denote by $\langle\cdot,\cdot\rangle_1$ the standard scalar product of $\mathbb{R}^m$. Furthermore denote by

$$\langle x, y\rangle_2 := \sum_{i=1}^{r_1+r_2} x_i y_i - \sum_{i=r_1+r_2+1}^{m} x_i y_i.$$

Then we have

$$\langle\gamma,\delta\rangle = \langle\Psi(\gamma),\Psi(\delta)\rangle_1 \text{ and } \mathrm{Tr}(\gamma\delta) = \langle\Psi(\gamma),\Psi(\delta)\rangle_2.$$

Now we are able to compare our two dual objects, the dual lattice Λ^* of Λ corresponding to $\langle\cdot,\cdot\rangle_1$ and the codifferent.

Lemma 20. Using the above notations. Then $\theta : \mathbb{R}^m \to \mathbb{R}^m$,

$$(x_1,\ldots,x_m) \mapsto (x_1,\ldots,x_{r_1+r_2}, -x_{r_1+r_2+1},\ldots,-x_m)$$

induces an isomorphism $\Lambda^* \to \Psi(\mathcal{O}_L^*)$ of $\mathbb{Z}$–modules.

Proof. θ is linear and has the property

$$\langle x, y\rangle_1 = \langle x, \theta(y)\rangle_2 \text{ for all } x, y \in \mathbb{R}^m.$$

We need to show that $\theta(\Lambda^*) = \Psi(\mathcal{O}_L)$. Note that θ^2 is the identity and therefore this is equivalent to $\theta(\Psi(\mathcal{O}_L)) = \Lambda^*$. Denote by $\omega_1,\ldots,\omega_m$ a $\mathbb{Z}$–basis of $\mathcal{O}_L$. Then $\Lambda = \mathbb{Z}\Psi(\omega_1) + \ldots + \mathbb{Z}\Psi(\omega_m)$. Choose $\gamma \in \mathcal{O}_L^*$ arbitrarily. Then $\mathrm{Tr}(\omega_i\gamma) \in \mathbb{Z}$ for $1 \le i \le m$ and therefore

$$\langle\Psi(\omega_i), \theta(\Psi(\gamma))\rangle_1 = \langle\Psi(\omega_i), \Psi(\gamma))\rangle_2 = \mathrm{Tr}(\omega_i\gamma) \in \mathbb{Z}.$$

Therefore $\theta(\Psi(\gamma)) \in \Lambda^*$ and we have shown $\theta(\Psi(\mathcal{O}_L^*)) \subseteq \Lambda^*$. Denote by $\tau_1,\ldots,\tau_m \in \mathcal{O}_L^*$ the dual basis of $\omega_1,\ldots,\omega_m$. Because of duality (e.g. see [17, Proof of Prop. 4.14]) we know that $\mathrm{disc}(\tau_1,\ldots,\tau_m) = \mathrm{disc}(\omega_1,\ldots,\omega_m)^{-1} = d_L^{-1}$. Furthermore $\theta(\Psi(\tau_i))$ $(1 \le i \le m)$ are linearly independent elements of Λ^* and the discriminant of the $\mathbb{Z}$–module generated by those elements is $|d_L^{-1}|$ since the corresponding determinants differ by a power of -1 because we have to consider the twists between our two bilinear forms. Therefore we know a subset $\theta(\Psi(\mathcal{O}_L^*)) \subseteq \Lambda^*$ which has the correct lattice discriminant. Therefore we get equality. □

Now we are able to get our bound by applying Lemma 20 and Theorem 19.

Lemma 21. Let L be a number field of degree m. Then $\mathcal{O}_L^*$ contains m $\mathbb{Q}$–linearly independent elements $\gamma_1,\ldots,\gamma_m$ such that $\|\gamma_i\|_2 \le \sqrt{m}$ for $1 \le i \le m$.

Proof. As before let $\Lambda := \Psi(\mathcal{O}_L)$, where Ψ is defined in (6). Now we claim that the first successive mimimum λ_1 equals $\sqrt{m}$ by taking the element $\Psi(1)$. Let $\gamma \in \mathcal{O}_L$. Then

$$1 \le |\mathrm{Norm}(\gamma)| = \left(\prod_{i=1}^{m} |\gamma_i|^2\right)^{1/2} \le \left(\frac{\sum_{i=1}^{m}|\gamma_i|^2}{m}\right)^{m/2}$$

$$= \left(\frac{\langle\Psi(\gamma),\Psi(\gamma)\rangle_1}{m}\right)^{m/2},$$

where the inequality is the one between geometric and arithmetic means. Now we get that $\langle\Psi(\gamma),\Psi(\gamma)\rangle_1 \geq m$ which finishes the proof that $\lambda_1 = \sqrt{m}$.

Applying Theorem 19 we find m linearly independent elements $y_1,\ldots,y_m \in \Lambda^*$ with euclidean length bounded by $m/\sqrt{m} = \sqrt{m}$. By using Lemma 20 we find elements $\theta(y_i) \in \Psi(\mathcal{O}_L^*)$ which have the same euclidean length. By choosing $\gamma_i := \Psi^{-1}(\theta(y_i))$ for $1 \leq i \leq m$ we finish our proof. $\square$

Now we are able to prove our theorem. Note that the field L takes the role of the principal subfield L_i in the statement.

Proof. [of Theorem 12] Using Lemma 21 we find m_i linearly independent elements β_j in $\mathcal{O}_L^*$ with 2-norm bounded by $\sqrt{m_i}$. When we interpret those elements in K, we get n/m_i copies of the complex embeddings, which gives that the 2-norm as elements of K is bounded by $\sqrt{n}$. Now apply Lemma 18. $\square$

4. CPU time in Magma

The aim of this section is to compare our algorithm with the state of the art, to be precise, to compare it with [12]. We use the computer algebra system Magma for this comparison, as it has an implementation of both [12] and the algorithm in this paper.

The algorithm from [12] uses a combinatorial approach in order to find block systems corresponding to subfields. Each combination is processed very quickly, however, it is possible that exponentially many combinations need to be tried. The number of combinations depends on the cycle structures of the elements of the Galois group, i.e., it depends on the number of factors of f mod prime numbers, and the degrees of these factors.

To keep the table below simple, we do not list these cycle structures, but list only r, n and l_f. Here r is the minimal number of monic irreducible factors of f in $\mathbf{F}_p[x]$ (taken over all p that do not divide the leading coefficient or the discriminant of f). The degree of f is denoted by n, while l_f denotes $\log_{10}(||f||_2)$.

Note that these polynomials are not chosen at random. Most polynomials have large Galois groups, and consequently, have small r. For small r, [12] is much faster than the algorithm in this paper (however, if r is large, then [12] becomes exponentially slow, and our algorithm will be faster). To get a good comparison, we constructed examples with a variety of values for r and n. To ensure that r is not very small, these examples are chosen to have Galois groups whose orders are not much greater than n. The table lists the CPU time (in seconds) for [12] and for the algorithm in this paper. Those marked "N/F" are computations that did not (yet) finish at the time of writing, and have been running for days, while "N/A" means not attempted.

r	n	l_f	[12]	HKN	r	n	l_f	[12]	HKN
6	36	5.4	1.3	30.7	13	75	32.4	228796.3	2911.2
7	75	10.6	75.2	453.2	14	81	13.1	16281.8	1849.3
8	48	4.4	15.5	101.2	15	81	14.0	31478.1	1644.1
8	56	11.2	3161.2	192.2	16	32	5.5	N/F	170.8
10	50	9.3	212.6	95.3	16	64	8.3	N/F	609.1
11	60	11.1	262.9	458.0	16	80	16.6	N/A	4195.9
12	60	15.0	5746.7	746.2	16	96	10.0	N/A	5494.3
12	64	10.5	277.4	417.1	18	90	21.5	N/A	4053.2
12	72	4.3	9575.4	785.9	20	100	28.2	N/A	43376.6

Observe that [12] was faster on all examples with $r < 8$, as well as some with $r \leq 12$, while our algorithm (the columns HKN) was faster on all examples with $r > 12$, as well as some with $r \geq 8$. We list two of the examples here: $r = 12$, $n = 72$, $f = x^{72} + 4x^{68} + 62x^{64} + 346x^{60} + 2077x^{56} + 3367x^{52} + 6647x^{48} + 6536x^{44} + 6634x^{40} + 5773x^{36} + 6634x^{32} + 6536x^{28} + 6647x^{24} + 3367x^{20} + 2077x^{16} + 346x^{12} + 62x^{8} + 4x^{4} + 1$.
$r = 16$, $n = 32$, $f = x^{32} - 161x^{28} + 25345x^{24} - 92414x^{20} + 305854x^{16} - 92414x^{12} + 25345x^{8} - 161x^{4} + 1$.
The remaining f's can be copied from `http://www.math.fsu.edu/~hoeij/subfields/` where some other details (Magma commands, memory usage, processor) are given as well.

References

[1] W. Banaszczyk. New bounds in some transference theorems in the geometry of number. *Math. Ann*, 296:625–635, 1993.

[2] Bernhard Beckermann and George Labahn. A uniform approach for the fast computation of matrix-type pade approximants. *SIAM J. Matrix Anal. Appl.*, 15:804–823, July 1994.

[3] Karim Belabas, Mark van Hoeij, Jürgen Klüners, and Allan Steel. Factoring polynomials over global fields. *J. Théor. Nombres Bordeaux*, 21:15–39, 2009.

[4] D. Casperson and J. McKay. Symmetric functions, m-sets, and Galois groups. *Math. Comput.*, 63:749–757, 1994.

[5] Henri Cohen and Francisco Diaz y Diaz. A polynomial reduction algorithm. *Séminaire de Théorie des Nombres de Bordeaux 2*, 3:351–360, 1991.

[6] Xavier Dahan and Éric Schost. Sharp estimates for triangular sets. In *Proceedings of the 2004 international symposium on Symbolic and algebraic computation*, ISSAC '04, pages 103–110, New York, NY, USA, 2004. ACM.

[7] H. Derksen. An algorithm to compute generalized Padé-Hermite forms. Preprint, Catholic University Nijmegen, 1994.

[8] J Dixon. Computing subfields in algebraic number fields. *J. Austral. Math. Soc. Series A*, 49:434–448, 1990.

[9] W. Hart. Flint. open-source C-library `http://www.flintlib.org`.
[10] Mark van Hoeij. Factoring polynomials and the knapsack problem. *J. Number Theory*, 95:167–189, 2002.
[11] A. Hulpke. Block systems of a Galois group. *Exp. Math.*, 4(1):1–9, 1995.
[12] J. Klüners. *Über die Berechnung von Automorphismen und Teilkörpern algebraischer Zahlkörper*. Dissertation, Technische Universität Berlin, 1997.
[13] J. Klüners. On computing subfields - a detailed description of the algorithm. *Journal de Théorie des Nombres de Bordeaux*, 10:243–271, 1998.
[14] D. Lazard and A. Valibouze. Computing subfields: Reverse of the primitive element problem. In A. Galligo F. Eyssete, editor, *MEGA-92, Computational algebraic geometry*, volume 109, pages 163–176. Birkhäuser, Boston, 1993.
[15] A. K. Lenstra, H. W. Lenstra Jr., and L. Lovász. Factoring polynomials with rational coefficients. *Math. Ann.*, 261:515–534, 1982.
[16] Michael B. Monagan, Keith O. Geddes, K. Michael Heal, George Labahn, Stefan M. Vorkoetter, James McCarron, and Paul DeMarco. *Maple 10 Programming Guide*. Maplesoft, Waterloo ON, Canada, 2005.
[17] Władisław Narkiewicz. *Elementary and Analytic Theory of Algebraic Numbers*. Springer, 2004.
[18] M. van Hoeij and V. Pal. Isomorphisms of algebraic number fields. arXiv:1012.0096v2, 2010.
[19] Mark van Hoeij and Andrew Novocin. Gradual sub-lattice reduction and a new complexity for factoring polynomials. In *LATIN*, pages 539–553, 2010.
[20] G. Villard. Certification of the QR factor R, and of lattice basis reducedness. In *Proceedings of the 2007 International Symposium on Symbolic and Algebraic Computation (ISSAC'07)*, pages 361–368, 2007.