# FAST DECOMPOSITION OF POLYNOMIALS WITH KNOWN GALOIS GROUP

ANDREAS ENGE AND FRANÇOIS MORAIN

Abstract. Let $f(X)$ be a separable polynomial with coefficients in a field $K$, generating a field extension $M/K$. If this extension is Galois with a solvable automorphism group, then the equation $f(X) = 0$ can be solved by radicals. The first step of the solution consists of splitting the extension $M/K$ into intermediate fields. Such computations are classical, and we explain how fast polynomial arithmetic can be used to speed up the process. Moreover, we extend the algorithms to a more general case of extensions that are no longer Galois. Numerical examples are provided, including results obtained with our implementation for Hilbert class fields of imaginary quadratic fields.

## 1. Introduction

Let $M_0/K_0$ be a finite separable field extension. Our aim is to describe the extension by a tower of intermediate fields. Clearly, it is sufficient to show how to obtain one intermediate field $L_0$ and to describe the extensions $L_0/K_0$ and $M_0/L_0$, since this process can be iterated. If $M_0/K_0$ is defined by some irreducible polynomial $f \in K_0[X]$, then the decomposition of the field extension into two relative extensions naturally leads to a factorisation of $f$.

In the case that $M_0/K_0$ is Galois with solvable automorphism group, such a factorisation of $f$ is an important step towards a solution of the equation by radicals. This can also be seen as a particular case of the subfield computations of algorithms presented in [2, 14, 15].

The algorithms needed to perform these computations are explained in [11, 12] and applied to the practical solution of equations arising in the theory of class fields. Our work extends this approach to polynomials $f$ which are not Galois themselves, but whose Galois groups split nicely into one part corresponding to the simple extension described by $f$ and another one describing the Galois closure. We also explain how to carry out the computations directly for $M_0/K_0$ without working in the Galois closure, which results in a significant speed-up. The necessary theory is developed in Section 2.

Relying on the Hecke representation of algebraic numbers reviewed in Section 3, we show that fast polynomial arithmetic can be applied to compute such a representation and thus to solve the problem of determining intermediate fields, cf. Section 4. We present and analyse the complete algorithm in Section 5. All these ideas are explained on numerical examples and again the case of class fields is detailed. This has important consequences in the implementation of the primality proving algorithm ECPP, see Sections 6 and 7.

## 2. Galois theory

Let $M$ be the Galois closure of $M_0$ of Galois group $\hat{G} = \mathrm{Gal}(M/K_0)$. Let $C$ be the subgroup of $\hat{G}$ fixing $M_0$ elementwise, that is, $M_0 = \mathrm{Fix}(C)$. We assume that $C$ possesses a normal complement $G$ in $\hat{G}$, so that in fact $\hat{G} = G \rtimes C$. If $K$ denotes the fixed field of $G$, then $M/K$ is Galois of group $G$, and $K/K_0$ is Galois of group $C|_K$. Let $H$ be a normal subgroup of $G$ and $L = \mathrm{Fix}(H)$, so that $M/L$ is Galois of group $H$ and $L/K$ is Galois of group $G/H$. Denote by $L_0$ the fixed field of $\langle H, C\rangle$. Then $K_0 \subseteq L_0 \subseteq M_0$.

¿From now on, we assume that $C$ normalises $H$, that is, $chc^{-1} \in H$ for any $c \in C$, $h \in H$. Then, any element $ch$ of $\langle H, C\rangle$ with $c \in C$ and $h \in H$ can be written as $h'c$ with $h' = chc^{-1} \in H$, and $\langle H, C\rangle$ is in fact the semidirect product $H \rtimes C$. Consequently, $L/L_0$ is Galois of group $C|_L$.

In particular, $[M_0 : L_0] = [M : L]$ and $[L_0 : K_0] = [L : K]$. We are thus in a situation where the tower of extensions $M_0/L_0/K_0$, although not Galois itself, bears strong similarities with the Galois tower $M/L/K$. The situation is summarised in Figure 1.

---

*Date*: December 13, 2002.
The second author is on leave from the French Department of Defense, Délégation Générale pour l'Armement.

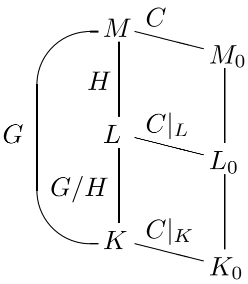


FIGURE 1. Field extensions and Galois groups

We now consider some irreducible polynomial $f \in K_0[X]$ which has a root $x \in M_0$. As $M = K \cdot M_0 = K \cdot K_0(x) = K(x)$ and $[M : K] = [M_0 : K_0]$, the polynomial $f$ remains irreducible over $K$. Denoting the image of a field element under a Galois automorphism by adding a superscript, we thus obtain

$$f(X) = \prod_{g \in G} (X - x^g).$$

Conversely, let $M/K$ be defined by an irreducible polynomial $f \in K[X]$ which has a root $x \in M_0$. Then $f = \prod_{g \in G}(X - x^g)$, and in fact $f \in K_0[X]$, as it is stable under $C$: For $c \in C$, we have

$$\begin{aligned} f^c(X) &= \prod_{g \in G} (X - x^{cg}) = \prod_{g \in G} (X - x^{(cgc^{-1})c}) \\ &= \prod_{g' \in G} (X - x^{g'c}) \text{ as } C \text{ normalises } G \\ &= \prod_{g \in G} (X - x^g) \text{ since } x \in M_0 \text{ is stable under } C \\ &= f \end{aligned}$$

Taking into account that $[M : K] = [M_0 : K_0]$, we find that $M_0 = K_0(x)$.

Hence, the polynomials generating $M_0/K_0$ are exactly the polynomials generating $M/K$ which additionally have a root in $M_0$.

To construct $M_0/L_0$, let us define for $\overline{g} \in G/H$ and $g \in G$ a representative of $\overline{g}$,

$$U_{\overline{g}}(X) = \prod_{h \in H} (X - x^{hg}).$$

The polynomial $U_{\overline{g}}$ does not depend on the chosen representative $g$ and is invariant under the action of $H$. Thus, $U_{\overline{g}} \in L[X]$. Precisely, $f$ factors as

$$f = \prod_{\overline{g} \in G/H} U_{\overline{g}},$$

any of the $U_{\overline{g}}$ generates $M/L$, and $G$ resp. $G/H$ act by permutation on the $U_{\overline{g}}$:

$$U_{\overline{g}}^{g'} = U_{\overline{g'g}} \text{ for } g' \in G.$$

Let us compute the action of $C$ on the $U_{\overline{g}}$. For $c \in C$,

$$U_{\overline{g}}^c = \prod_{h \in H} (X - x^{chg}) = \prod_{h \in H} \left( X - x^{(chc^{-1})(cgc^{-1})c} \right) = U_{\overline{cgc^{-1}}} \tag{1}$$

as $C$ normalises $H$ and $G$ and $x$ is stable under $C$.

So $C$ also acts by permutation on the $U_{\overline{g}}$, and more precisely by conjugation of the index. In particular, $U_{\overline{1}} = (X - x) \prod_{h \in H \setminus \{1\}} (X - x^h)$ is stable under $C$. It is thus a polynomial in $L_0[X]$ of degree $|H| = [M_0 : L_0]$ with root $x$ such that $M_0 = L_0(x)$. Hence, $U_{\overline{1}}$ is irreducible and generates $M_0/L_0$. Write

$$U_{\overline{g}} = \sum_{k=0}^{|H|} (-1)^{|H|-k} \vartheta_{\overline{g}}^{(k)} X^k \text{ with } \vartheta_{\overline{g}}^{(k)} \in L, \vartheta_{\overline{1}}^{(k)} \in L_0.$$

To construct $L_0/K_0$, we define polynomials $V_k$ whose roots are the Galois images of $\vartheta_{\overline{1}}^{(k)}$ under $G/H$, that is,

$$V_k = \prod_{\overline{g} \in G/H} \left( Y - \vartheta_{\overline{g}}^{(k)} \right).$$

Notice that the actions of $G$ and $C$ on the $\vartheta_{\overline{g}}^{(k)}$ are the same as on the $U_{\overline{g}}$. So as $G$ and $C$ permute the $\vartheta_{\overline{g}}^{(k)}$, the polynomials $V_k$ lie in $K_0[Y]$. Furthermore, the $\vartheta_{\overline{1}}^{(k)}$ are elements of $L_0$. Thus, if some $V_{k_0}$ is irreducible, then it generates $L_0/K_0$. We assume henceforth that this is the case; otherwise, one has to apply a translation to the roots of $f$.

We have thus split $M_0 = K_0[X]/(f)$ into a tower of two field extensions $L_0 = K_0[Y]/(V_{k_0})$ and $M_0 = L_0[X]/(U_{\overline{1}})$.

Notice that if $M_0/K_0$ is Galois itself, then $C = 1$, the fields with index 0 coincide with the indexless fields, and we recover the approach of [11].

## 3. Hecke representation

The concluding statement of the previous section is overly optimistic in the sense that $L_0/K_0$ is well described by some polynomial $V_{k_0} \in K_0[Y]$, but the construction of the relative extension $M_0/L_0$ as $L_0[X]/(U_{\overline{1}})$ is not yet completely effective. Indeed, while we show by theoretical reasoning that the coefficients of $U_{\overline{1}}$ lie in $L_0$, we have described them only in terms of the absolute extension $M_0/K_0$, namely as elementary symmetric functions in some of the roots of a polynomial defining the large extension. It remains to explicitly write down the coefficients as elements of $K_0[Y]/(V_{k_0})$.

In [11], it was suggested to use a power basis representation for algebraic numbers. When dealing with algebraic integers, this is already not satisfying from a theoretical point of view since integral power bases do not always exist. In practice, this leads to tedious computations based on the integrality defect. As already observed in [12], the following *Hecke representation* provides an elegant solution, which is furthermore well suited for computations. This representation is based on the following lemma [13].

**Lemma 1.** *Let $L/K$ be a separable field extension of degree $n$ with primitive element $\alpha = \alpha_0$, and denote by $(\alpha_i)_{0 \leq i < n}$ the conjugates of $\alpha$ and by $f = \prod_{i=0}^{n-1} (Y - \alpha_i)$ its minimal polynomial. Then any element $\vartheta \in L$ may be written as*

$$\vartheta = \frac{g_\vartheta(\alpha)}{f'(\alpha)} \text{ for some } g_\vartheta \in K[Y].$$

*Moreover, if $\mathcal{O}_K$ is an integrally closed subring of $K$ and $\alpha$ and $\vartheta$ are integral over $\mathcal{O}_K$, then $f$, $g_\vartheta \in \mathcal{O}_K[Y]$.*

*Proof.* Let $(\vartheta_i)_{0 \leq i < n}$ be the conjugates of $\vartheta$. Put

$$g_\vartheta(Y) = \sum_{i=0}^{n-1} \vartheta_i \frac{f(Y)}{Y - \alpha_i}. \tag{2}$$

By Galois theory, we have $g_\vartheta \in K[Y]$. Furthermore, $g_\vartheta(\alpha) = \vartheta f'(\alpha)$. If $\alpha$ and $\vartheta$ are integral over $\mathcal{O}_K$, then so are the coefficients of $f$ and $g_\vartheta$, whence $f$, $g_\vartheta \in \mathcal{O}_K[Y]$ since $\mathcal{O}_K$ is integrally closed. □

Formula (2) is the key to writing the coefficients $\vartheta_{\overline{1}}^{(k)}$ of $U_{\overline{1}}$ in terms of the generating element $\vartheta_{\overline{1}}^{(k_0)}$ of $L_0/K_0$. It is now straightforward to formulate an algorithm computing the field extension decomposition. However, we postpone this task to Section 5, as the analysis of the algorithm requires to have a closer look at the underlying polynomial arithmetic.

## 4. Fast polynomial arithmetic

The key ingredient of the algorithm derived from the theory of the previous sections, and to be described in detail in Section 5, is the computation of the Hecke representation of an algebraic number. The problem is very closely related to that of multiple polynomial interpolation over the same set of abscissae, but with varying ordinates. In this section, we thus recall fast algorithms on polynomials and improve them with respect to our specific problem. The reference is [8, Chapter 10], and we use the same notations. We first treat the basic case corresponding to a Galois extension in Section 4.1. Then we show in Section 4.2 how even in the non-Galois case one may restrict to polynomial arithmetic in the base field and thus gain performance.

4.1. **Basics — the Galois case.** Let $(u_0, u_1, \ldots, u_{n-1})$ be $n$ elements of a field $K$. The subproduct tree associated to them is a tree built from subproducts of the $u_i$'s obtained by a divide and conquer approach as shown in Figure 2.

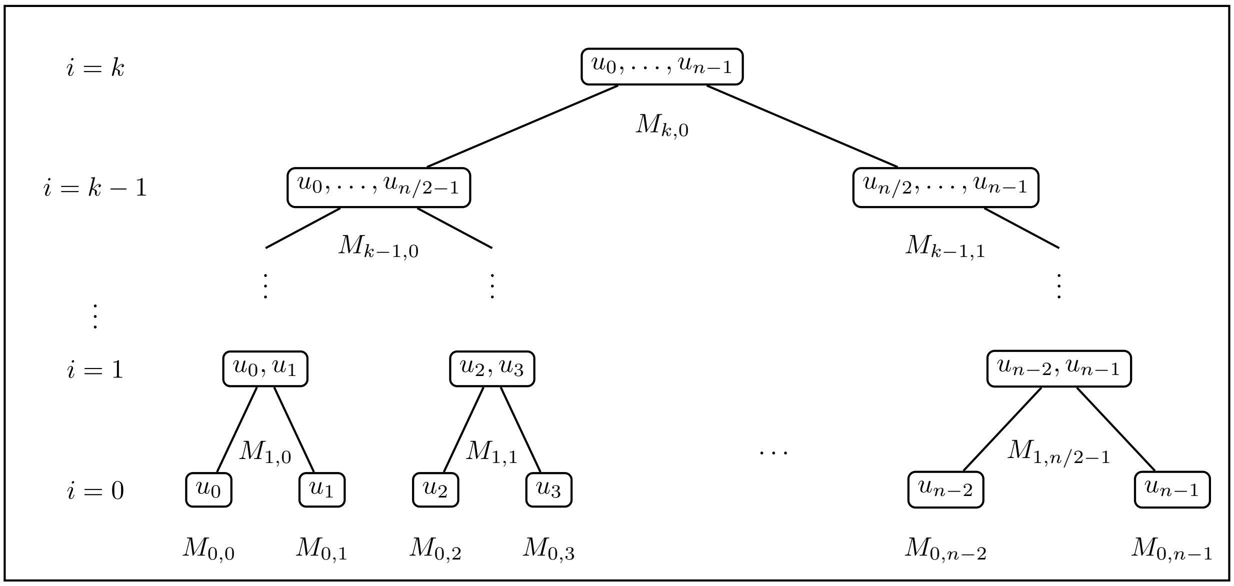


FIGURE 2. Subproduct tree

To begin with, assume first that $n = 2^k$. We define for $0 \leq i \leq k = \log_2 n$ and $0 \leq j < 2^{k-i}$:

$$M_{i,j} = \prod_{\ell=0}^{2^i-1} (X - u_{j \cdot 2^i + \ell}).$$

When $n$ is not a power of 2, with $2^{k-1} < n \leq 2^k$, we build the tree with $2^k$ leaves, replacing the missing $M_{i,j}$'s by 1. We have to adapt the procedure of creation of intermediate nodes in this case, remembering that we are proceeding bottom to top.

**procedure** BUILDSUBPRODUCTTREE

1. [Level 0 :]
   a. **for** $j = 0..n-1$ **do** $M_{0,j} := X - u_j$ ;
   b. **for** $j = n..2^k - 1$ **do** $M_{0,j} := 1$ ;
2. [Levels $1 \leq i \leq k$ :] **for** $j = 0..2^{k-i} - 1$ **do**
   **if** $M_{i-1,2j+1} = 1$ {empty right subtree} **then**
      **if** $M_{i-1,2j} = 1$ {empty left subtree} **then** $M_{i,j} = 1$ ; **else** $M_{i,j} := M_{i-1,2j}$ ;
   **else** {ordinary case} $M_{i,j} := M_{i-1,2j} M_{i-1,2j+1}$.

If the underlying polynomial arithmetic uses Karatsuba multiplication and counting only multiplications in $K$, it is shown in [3] that the strategy of adding leaves with content 1 rightmost in the tree is optimal. The cost of building the tree is then bounded above by $3^k$ multiplications in $K$, or otherwise said, it is in $O(M_K(n))$, where $M_K(n)$ denotes the cost of multiplying two polynomials over $K$ with $n$ coefficients. If the polynomial multiplication uses the FFT, then the cost of creating the tree is no more than $M_K(n)$, cf. [8].

The tree $M$ is conveniently stored in an array `T` of size $2^{k+1} - 1$ in such a way that `T[0]` contains $M_{k,0}$, `T[1]` contains $M_{k-1,0}$, `T[2]` contains $M_{k-1,1}$, ..., `T[2^(k-i)-1+j]` contains $M_{i,j}$. Thus, if the node $M_{i,j}$ is stored at `T[s]`, then the node $M_{i-1,2j}$ (resp. $M_{i-1,2j+1}$) is stored at `T[2*s+1]` (resp. `T[2*s+2]`).

We take the opportunity to rewrite [8, Algorithm 10.9], which is the heart of the computation of the Hecke representation, in a version more suited to implementation.

**procedure** LCFLM$(u, c, k, T, s)$

INPUT: $u_0, \ldots, u_{n-1}, c_0, \ldots, c_{n-1}$ with $2^{k-1} < n \le 2^k$; $T$ containing the subproduct tree; $s = 2^{k-i} - 1 + j$ s.t. $0 \le i < k$, $0 \le j < 2^{k-i}$.

OUTPUT: $\sum_{\ell=0}^{2^i-1} c_{j\cdot 2^i+\ell} \frac{M_{i,j}}{X-u_{j\cdot 2^i+\ell}} \in K[X]$.

1. **if** $i = 0$, i.e., $s \ge 2^k - 1$, **then return** $c_j = c_{s-2^k+1}$.
2. **if** $T[2s+2] = 1$ **then return** LCFLM$(u, c, k, T, 2s+1)$ ;
3. $r_0 :=$ LCFLM$(u, c, k, T, 2s+1)$;
4. $r_1 :=$ LCFLM$(u, c, k, T, 2s+2)$;
5. **return** $r_0 T[2s+2] + r_1 T[s+1]$.

and the launching procedure is then simply

**procedure** LinearCombinationForLinearModuli$(u, c, n, T)$
1. **return** LCFLM$(u, c, \lceil \log_2 n \rceil, T, 0)$.

The improvement to the procedure LCFLM, in case that $n$ is not a power of 2, stems from the fact that if we encounter a right $M$-subtree which is 1, then this branch is pruned.

The cost analysis of LinearCombinationForLinearModuli is very closely related to that of BuildSubproductTree, which is easily seen by rewriting BuildSubproductTree in a recursive style, cf. [3]. Let $M_K(m, n)$ denote the cost of multiplying two polynomials over $K$ with $m$ resp. $n$ coefficients. Denote by $B(n)$ resp. $L(n)$ the cost of BuildSubproductTree resp. LinearCombinationForLinearModuli. For $L(n)$, we suppose that the subproduct tree has already been computed. Then for $2^{k-1} < n \le 2^k$ we have

$$B(1) = 0 \text{ and } B(n) = B(2^{k-1}) + B(n - 2^{k-1}) + M_K(2^{k-1}, n - 2^{k-1}),$$

while

$$L(1) = 0 \text{ and } L(n) = L(2^{k-1}) + L(n - 2^{k-1}) + 2M_K(2^{k-1}, n - 2^{k-1}).$$

It follows that $L(n) = 2B(n)$. In particular, the analysis of [3] carries over and shows that in the Karatsuba model, our strategy of filling the subproduct tree with rightmost 1's remains optimal also for LinearCombinationForLinearModuli.

4.2. **Tricks — the non-Galois case.** In order to determine Hecke representations in the non-Galois setting of Section 2, the procedures BuildSubproductTree and LinearCombinationForLinearModuli can be applied verbatim. Then all computations take place in $K$, and the final result will be in $K_0$. It is desirable, however, to work directly in $K_0$, where in general the basic arithmetic is less costly. To this goal, notice that with the notations of Sections 2 and 4.1, while the values $u_i$ and $c_i$ are elements of $L$, they nevertheless form complete orbits under $C|_L = \mathrm{Gal}(L/L_0)$. This is precisely what causes the result to be defined over $K_0$ instead of $K$. We may then modify our procedures by computing first the results for the orbits of $C$, which will be elements of $K_0[X]$, and filling the leaves of the trees with these results. Otherwise said (and not taking the complications into account coming from the fact that not all integers are powers of 2), we stop our recursion at a higher level.

To do so, we need control of the orbits. Precisely, we require a function that takes as input the $u_0, \ldots, u_{n-1}$ and $c_0, \ldots, c_{n-1}$ and the action of $C$ on them. It should return the number $n'$ of joint orbits of the $u_i$'s and $c_i$'s, their cardinalities $n_j$ and the orbits $(U_j, (c_{j,0}, \ldots, c_{j,n_j-1}))$ with $U_j = (u_{j,0}, \ldots, u_{j,n_j-1})$ for $0 \le j < n'$ Notice that the orbits will in general not all have the same size (in particular, one of them is trivial, but not all of them are, unless we are in the Galois case $M = M_0$). In the setting of Section 2, the procedure can be realised symbolically making use of the group structure of $\mathrm{Gal}(M/K_0) = G \rtimes C$ and of the action of the Galois group on the $u_i$ and $c_i$. Another approach is possible, for instance, when $C$ is generated by the complex conjugation, in which case the orbits can be obtained by simple inspection of the $u_i$ and $c_i$, see Section 6.

The two procedures now generalise as follows. The function OrbitalBuildSubproductTree is the same as BuildSubproductTree, replacing $n$ by $n'$ and $k$ by $k'$ s.t. $2^{k'-1} < n' \le 2^{k'}$ and Step 1a by
1a'. **for** $j = 0..n' - 1$ **do** $M_{0,j} := \prod_{r=0}^{n_j-1}(X - u_{j,r}) \in K_0[X]$.

In LCFLM, we need to replace the input $c_j$ corresponding to an element $u_j$ by the interpolation polynomial $C_j$ corresponding to an orbit $U_j$, letting

$$C_j = \sum_{r=0}^{n_j-1} c_{j,r} \frac{M_{0,j}}{X - u_{j,r}} \in K_0[X].$$

The $C_j$'s may be obtained by applying the original BuildSubproductTree and LinearCombinationForLinearModuli; in many cases with small $C$, however, they are trivial to write down, cf. Section 6.

The remainder of the function OrbitalLinearCombinationForLinearModuli is straightforward.

## 5. Algorithm

5.1. **Description.** We use the notations of Section 2. Let $m = |H|$ and $n = |G/H|$. We assume that a root $x \in M_0$ of $f$ is identified, that $H = \{h_0 = 1, h_1, \ldots, h_{m-1}\}$ and a system of representatives $\{g_0 = 1, g_1, \ldots, g_{n-1}\}$ of $G/H$ are given, and that the action of $G$ on $x$ is explicitly known. This allows to create an ordered list of conjugates $(x_0 = x, x_1, \ldots, x_{mn-1}) = \left(x^{h_0 g_0}, x^{h_1 g_0}, \ldots, x^{h_{m-1} g_0}, x^{h_0 g_1}, \ldots, x^{h_{m-1} g_{n-1}}\right)$, i.e., $x_{i+mj} = x^{h_i g_j}$. In the non-Galois case, where $C$ is non-trivial, we furthermore need an enumeration of the elements of $C$ and must be able to determine their action on the $x_i$.

**procedure** Decomposition($f(X)$, $\mathcal{R}$, $m$, $n$, $C$)
INPUT: $\mathcal{R} = (x_0, \ldots, x_{mn-1})$ a list of the roots of $f$ ordered as explained above.
OUTPUT: a defining polynomial $V_{k_0}(Y)$ of $L_0/K_0$ and a factor $U_0(X, Y)$ of $f$ defining $M_0/L_0$.

1. **for** $i = 0..n-1$ **do** compute

$$U_i(X) = U_{\overline{g}_i}(X) = \prod_{j=0}^{m-1} (X - x_{mi+j}) = \sum_{k=0}^{m} (-1)^{m-k} \vartheta_i^{(k)} X^k$$

2. **do** compute

$$V_k(Y) = \prod_{i=0}^{n-1} \left(Y - \vartheta_i^{(k)}\right) \in K_0[Y]$$

for $k = 0..m-1$ **until** one of them, $V_{k_0}(Y)$, is irreducible over $K$. If none of them is irreducible, declare failure.

3. **for** $k = 0..m-1$ **do** compute the polynomials $g_k(Y) = g_{\vartheta_0^{(k)}}(Y)$ defining the Hecke representations of the coefficients of $U_0 = U_{\overline{1}}$ according to (2).

4. **return** $V_{k_0}(Y)$ and

$$U_0(X, Y) = X^m + \frac{1}{V'_{k_0}(Y)} \sum_{k=0}^{m-1} (-1)^{m-k} g_k(Y) X^k.$$

5.2. **Analysis. Step 1:** The computation of the $U_i(X)$'s costs $n$ applications of BuildSubproductTree for polynomials of degree $m$. Notice that if we can compute the action of $C$ on the coefficients of the $U_i$'s, then we may use (1) to compute only some of the $U_i$'s and deduce the others via the action of $C$. In the best case, this may reduce the complexity by a factor of about $|C|$, cf. the examples of Section 6.
**Step 2:** The computation of the $V_k(Y)$'s costs at most $m$ applications of BuildSubproductTree for polynomials of degree $n$ in the basic case. If $C$ is non-trivial, then we need to identify the orbits of the $\vartheta_i^{(k)}$ and then apply at most $m$ times OrbitalBuildSubproductTree on polynomials whose degrees are defined by the number of orbits.
As $V_k(Y)$ is a power of an irreducible polynomial over $K$ (the minimal polynomial of $\vartheta_0^{(k)}$) we may check for irreducibility by verifying that $\gcd(V_k, V_k') = 1$. In many cases, it is possible to perform a sufficient simpler test. Over the rationals, for instance, we may reduce modulo small prime numbers and check for irreducibility.

**Step 3:** Having kept the subproduct tree leading to $V_{k_0}$ in Step 2, we need $m-1$ calls to LinearCombinationForLinearModuli on a polynomial of degree $n$. (Notice that we already know $g_{k_0} = Y\, V'_{k_0}(Y)$.) For non-trivial $C$, one may use OrbitalLinearCombinationForLinearModuli on the orbits already identified in Step 2.

Denote by $B(n)$ the cost of BuildSubproductTree and by $L(n)$ the cost of LinearCombinationForLinearModuli. In the basic Galois case and computing all possible $V_k$, the total cost of the algorithm then becomes

$$nB(m) + mB(n) + (m-1)L(n) \approx nB(m) + 3mB(n),$$

where $B(n) = O(M_K(n))$ in the Karatsuba and $B(n) = O(\log n M_K(n))$ in the FFT case.

As a possible improvement to the algorithm, one may want to compute all the $V_k$'s in Step 2 and choose the best one according to some optimality criterion, for instance the heights of the resulting polynomials, the sizes of their roots, etc. If the process is to be iterated with $V_{k_0}$ in the role of $f$, it may pay off to carefully arrange the subproduct tree in Step 2, so that the subproduct trees of the $U_i$'s of the next iteration appear as branches and need not be recomputed; see the example in Section 6.2.

5.3. **An illustrative example: Kummer extensions.** To illustrate the algorithm of the previous section, we describe as a very simple example the case of a Kummer extension.

Let $K_0 = \mathbb{Q}$ and $f(X) = X^{15} - a$ for some $a \in \mathbb{Q}$, neither a third nor a fifth power of a rational number, and denote by $x$ the real root of $f$, $M_0 = \mathbb{Q}(x)$. Then $K = \mathbb{Q}(\zeta)$, where $\zeta$ is a primitive 15-th root of unity, and $M = \mathbb{Q}(\zeta, x)$. The Galois groups are given by $G = \mathrm{Gal}(\mathbb{Q}(\zeta, x)/\mathbb{Q}(\zeta)) = \langle \sigma \rangle$ with $\sigma(x) = \zeta x$, $\sigma(\zeta) = \zeta$, and $C = \{\tau_i : i \in (\mathbb{Z}/15\mathbb{Z})^\times\} \simeq \mathrm{Gal}(\mathbb{Q}(\zeta)/\mathbb{Q})$ with $\tau_i(x) = x$, $\tau_i(\zeta) = \zeta^i$. Then $C$ normalises $G$ and all of its subgroups, as can be seen from the relation $\tau_i \sigma \tau_i^{-1} = \sigma^i$.

We choose $H = \langle \sigma^3 \rangle = (1, \sigma^3, \sigma^6, \sigma^9, \sigma^{12})$, and a natural system of representatives of $G/H$ is given by $(1, \sigma, \sigma^2)$. The roots of $f$, in the ordering of Section 5, are

$$\mathcal{R} = \left(x, \zeta^3 x, \zeta^6 x, \zeta^9 x, \zeta^{12} x;\ \zeta x, \zeta^4 x, \zeta^7 x, \zeta^{10} x, \zeta^{13} x;\ \zeta^2 x, \zeta^5 x, \zeta^8 x, \zeta^{11} x, \zeta^{14} x\right).$$

We obtain

$$\begin{aligned} U_0 &= \prod_{j=0}^{4} \left(X - \zeta^{3i} x\right) = X^5 - x^5 \\ U_1 = U_{\overline{\sigma}} &= \prod_{j=0}^{4} \left(X - \zeta^{3i+1} x\right) = X^5 - \zeta^5 x^5 \end{aligned}$$

Instead of computing $U_2 = U_{\overline{\sigma^2}}$ directly, we may observe that $\sigma^2 = \tau_2 \sigma \tau_2^{-1}$, so that by (1),

$$U_2 = U_1^{\tau_2} = X^5 - \zeta^{10} x^5.$$

In this case, we have $V_1 = \cdots = V_4 = Y^5$, whereas $V_0$ is non-trivial. Notice that the constant coefficients of the $U_i$'s split into two orbits under the action of $C$, given by $\{x^5\}$ and $\{\zeta^5 x^5, \zeta^{10} x^5\}$. We may thus compute

$$V_0 = \left(Y - x^5\right) \left((Y - \zeta^5 x^5)(Y - \zeta^{10} x^5)\right) = \left(Y - x^5\right) \left(Y^2 + x^5 Y + x^{10}\right) = Y^3 - a;$$

notice that once the orbits are regrouped, there is no need any more to carry $\zeta$'s along.

Now $U_0$ has only one non-trivial coefficient, and it is exactly the one defining the intermediate field. Hence, there are no Hecke representations to compute, and we have decomposed $\mathbb{Q}(\sqrt[15]{a})/\mathbb{Q}$ as $\mathbb{Q}(\sqrt[15]{a})/\mathbb{Q}(\sqrt[3]{a})/\mathbb{Q}$ with defining polynomials

$$V_0(Y) = Y^3 - a,\ U_0(X, Y) = X^5 - Y.$$

## 6. Application: Class fields of imaginary quadratic fields

Class fields of imaginary quadratic fields are used in a crucial way in the primality proving algorithm ECPP, which motivated our research. During the algorithm, one needs to find a root of the polynomial defining the class field modulo a prime. A fast way of doing so is obtained by first decomposing the polynomial (independently of the prime) over a tower of algebraic field extensions, then reducing and solving a series of lower degree equations in the finite prime field. See [1, 11, 5] for more details and a description of the general approach. This large class of fields fits into the framework developed above and provides many numerical examples for our algorithm.

6.1. **Theoretical setting.** With the notations of the preceding sections, we put $K_0 = \mathbb{Q}$ and $K = \mathbb{Q}(\sqrt{-D})$, an imaginary quadratic field of discriminant $-D < 0$. The field $M$ is the Hilbert class field, $G \sim \mathrm{Cl}(-D)$ is the class group and its cardinality $h(-D)$ the class number of $K$. The group $C$ is generated by the complex conjugation; dealing with this action is perhaps the simplest possible case if we use a floating point representation of the roots.

The polynomials $f(X)$ we are interested in are minimal polynomials $H_D[u](X)$ of class invariants $u$, such as the $j$-invariant, or (generalised) Weber functions, cf. [4, 6]. We restrict the presentation to the cases where $H_D[u](X)$ is in fact defined over $\mathbb{Q}$, so that our theory applies.

Let $Q = [a, b, c]$ be a quadratic form of discriminant $-D$. We can associate to $Q$ a complex number $\tau_Q = (-b + \sqrt{-D})/(2a)$, so that the roots of $H_D[u]$ are in fact $u(Q) = u(\tau_Q)$, where $Q$ varies over a certain system of representatives of $\mathrm{Cl}(-D)$. The action of the Galois group on these roots is explicitly given as follows: For $Q \in \mathrm{Cl}(-D)$, the associated automorphism $\sigma_Q \in G$ acts by

$$\sigma_Q\left(u(Q')\right) = u(Q^{-1} \cdot Q').$$

Moreover, complex conjugation acts via

$$\overline{u(Q)} = u(Q^{-1})$$

which immediately tells us that $u(Q)$ is real if and only if $Q$ has order 2 ($Q$ is then called an *ambiguous* form).

6.2. **A complete numerical example.** Let us consider the case of $D = 95$. We choose the defining polynomial

$$H_{4\cdot 95}[f/\sqrt{2}](X) = X^8 - 2\,X^7 - 2\,X^6 + X^5 + 2\,X^4 - X^3 + X - 1,$$

whose roots in the right order for $m = 2$ and $n = 4$ are given by

| $i$ | $x_i$ | comment |
|---|---|---|
| 0 | $2.533$ | real |
| 1 | $-0.8287 - 0.6954i$ | |
| 2 | $0.8090 + 0.2097i$ | |
| 3 | $0.2107 - 0.6954i$ | |
| 4 | $-0.9146$ | real |
| 5 | $0.2107 + 0.6954i$ | $= \overline{x_3}$ |
| 6 | $0.8090 - 0.2097i$ | $= \overline{x_2}$ |
| 7 | $-0.8287 + 0.6954i$ | $= \overline{x_1}$ |

Using our theoretical knowledge, we are able to spot the complex conjugates without actually computating them.

We compute the polynomials $U_i$ as follows:

| $i$ | $U_i$ | | comment |
|---|---|---|---|
| 0 | $(X - x_0)(X - x_4) = X^2 - \vartheta_{0,1}X + \vartheta_{0,0}$ | $= X^2 - 1.618X - 2.317$ | real |
| 1 | $(X - x_1)(X - x_5) = X^2 - \vartheta_{1,1}X + \vartheta_{1,0}$ | $= X^2 + 0.6180X + 0.3090 - 0.7229i$ | |
| 2 | $(X - x_2)(X - x_6) = X^2 - \vartheta_{2,1}X + \vartheta_{2,0}$ | $= X^2 - 1.618X + 0.6985$ | real |
| 3 | $(X - x_3)(X - x_7) = X^2 - \vartheta_{3,1}X + \vartheta_{3,0}$ | | $= \overline{U_1(X)}$ |

Since we have the information on the conjugates, we do not need to compute $U_3$, which saves some time. From this, we can deduce the conjugacy properties of the $V$'s, namely that $\vartheta_{3,j} = \overline{\vartheta_{1,j}}$, and $\vartheta_{0,j}$ and $\vartheta_{2,j}$ are real.

To compute $V_0$, we can precompute the trees for the $U$'s of the next iteration, namely

$$U_0^*(X) = (X + 2.317)(X - 0.6985),\ U_1^*(X) = (X - (0.3090 - 0.7229i))\overline{(X - (0.3090 - 0.7229i))}$$

and deduce from this

$$V_0(Y) = U_0^*(Y)U_1^*(Y) = Y^4 + Y^3 - 2\,Y^2 + 2\,Y - 1.$$

For the sake of completeness, note that

$$V_1(Y) = Y^4 - 2\,Y^3 - Y^2 + 2\,Y + 1 = \left(Y^2 - Y - 1\right)^2.$$

We finally compute:

$$U_0(X,Y) = X^2 + \frac{1}{V_0'(Y)}\left((-2Y^3 + Y^2 - 3Y - 1)X - Y^3 + 4Y^2 - 6Y + 4\right).$$

The next and last iteration yields $V_0^*(Y_1) = Y_1^2 + Y_1 - 1$ and

$$U_0^*(Y,Y_1) = Y^2 + \frac{1}{V_0^*(Y_1)'}((Y_1 - 2)Y - Y_1 + 2),$$

so that we replaced the resolution of a degree 8 equation by the solution of $V_0^*(Y_1) = 0$, followed by that of $U_0^*(Y, Y_1) = 0$ and finally $U_0(X,Y) = 0$.

## 7. Implementation

We have implemented our algorithms in C. As we represent the roots by complex numbers with high precision, we rely on the packages `gmp`, `mpfr` and `mpc` [9, 10, 7] for the basic arithmetic. The fast polynomial arithmetic as described in Section 4 is implemented in the Karatsuba model. We linked the field decomposition code with our programs for computing class fields and ultimately with the second author's current implementation of ECPP. In the following tables, we provide some timings for class polynomials $H_D[f/\sqrt{2}]$. We give the precision of the computations (in bits), the time in seconds needed for computing complex approximations of the roots to this precision, and the total running time. All examples have been obtained on a Pentium III with 450 MHz.

| $D$ | $h(-D)$ | $\mathrm{Cl}(-D)$ | prec | $u$ | time |
|---|---|---|---|---|---|
| 2519 | 64 | $C(64)$ | 250 | 0.1 | 0.1 |
| 10295 | 128 | $C(32) \times C(4)$ | 508 | 0.3 | 0.8 |
| 39431 | 256 | $C(128) \times C(2)$ | 1095 | 1.6 | 5.4 |
| 132599 | 512 | $C(512)$ | 2320 | 12.4 | 19.7 |
| 328319 | 1024 | $C(1024)$ | 4571 | 86.5 | 138.4 |
| 1333631 | 2048 | $C(2048)$ | 10130 | 777.6 | 1256.8 |
| 4599839 | 4096 | $C(2048) \times C(2)$ | 21601 | 6586.9 | 9023.7 |
| 3671 | 81 | $C(81)$ | 309 | 0.1 | 0.2 |
| 29399 | 243 | $C(81) \times C(3)$ | 988 | 1.4 | 4.0 |
| 178559 | 729 | $C(729)$ | 3143 | 30.7 | 46.5 |
| 2102999 | 2187 | $C(2187)$ | 11859 | 1056.2 | 1645.3 |
| 361919 | $31 \cdot 29$ | $C(899)$ | 4327 | 62.1 | 82.2 |

Let us examine the last example in more detail. We use $m = 31$ and $n = 29$. We first compute the 29 polynomials $U$ of degree 31. There are 1 real polynomial and 14 pairs of complex conjugate ones. The real one takes 0.16 seconds and each pair of complex ones takes 0.32 seconds. Constructing the 31 polynomials $V$ of degree 29 takes 13 seconds; reconstructing the algebraic factor $U(X,Y)$ requires 30 interpolations, for a total of 7.15 seconds.

The time needed in the Hecke phase for $D = 132599$ of degree 512 illustrates the effect of fast multiplication. Reconstructing one polynomial $U(X,Y)$ of degree 256 takes 2.98 seconds, whereas one of degree 128 takes 0.99 seconds: this is exactly the factor 3 predicted theoretically in the Karatsuba model.

## 8. Conclusions

We have presented an algorithm for decomposing certain non-Galois field extensions, generalising previously known algorithms for Galois extensions. Our implementation is used in the second author's ECPP program. It enables us to make use of high degree class fields. For instance, smooth class numbers around 300 are employed without any problem, while the previous implementation did not go beyond a degree 64. Moreover, the program runs so fast for small and medium degrees that precomputing large tables of class fields is no longer beneficial. This has a positive impact on the size of primes that can be certified.

**Acknowledgements.** We thank G. Hanrot and P. Zimmermann for rapidly correcting the (minor) bugs we found in `mpfr`.

## References

[1] A. O. L. Atkin and F. Morain. Elliptic curves and primality proving. *Math. Comp.*, 61(203):29–68, July 1993.
[2] J. D. Dixon. Computing subfields in algebraic number fields. *J. Austral. Math. Soc. Ser. A*, 49:434–448, 1990.
[3] A. Enge, P. Gaudry, G. Hanrot, and P. Zimmermann. Reconstructing a poynomial from its roots. In preparation, 2002.
[4] A. Enge and F. Morain. Further investigations of the generalised weber functions. Preprint, July 2001.
[5] A. Enge and F. Morain. Comparing invariants for class fields of imaginary quadratic fields. In C. Fieker and D. R. Kohel, editors, *ANTS IV — Algorihmic Number Theory*, volume 2369 of *Lecture Notes in Comput. Sci.*, pages 252–266. Springer-Verlag, 2002.
[6] A. Enge and R. Schertz. Constructing elliptic curves from modular curves of positive genus. Preprint, 2001.
[7] A. Enge and P. Zimmermann. `mpc` — a library for multiprecision complex arithmetic with exact rounding. Version 0.4.1, available from http://www.loria.fr/˜zimmerma.
[8] J. von zur Gathen and J. Gerhard. *Modern Computer Algebra*. Cambridge University Press, 1999.
[9] T. Granlund et. al. `gmp` — gnu multiprecision library. Version 4.1.1, available from http://www.swox.com/gmp.
[10] G. Hanrot, V. Lefèvre, and P. Zimmermann et. al. `mpfr` — a library for multiple-precision floating-point computations with exact rounding. Version `cvs`, available from http://www.mpfr.org.
[11] G. Hanrot and F. Morain. Solvability by radicals from an algorithmic point of view. In B. Mourrain, editor, *Symbolic and algebraic computation*, pages 175–182. ACM, 2001. Proceedings ISSAC'2001, London, Ontario.
[12] G. Hanrot and F. Morain. Solvability by radicals from a practical algorithmic point of view. Submitted. Available from `http://www.lix.polytechnique.fr/Labo/Francois.Morain/`, November 2001.
[13] E. Hecke. *Vorlesungen über die Theorie der algebraischen Zahlen*. Chelsea Publishing Company, 2 edition, 1970.
[14] J. Klüners. On computing subfields. A detailed description of the algorithm. *J. Théor. Nombres Bordeaux*, 10:243–271, 1998.
[15] J. Klüners and M. Pohst. On computing subfields. *J. Symbolic Comput.*, 24:385–397, 1997.

École polytechnique (LIX, CNRS/UMR 7650) and INRIA Futurs, 91128 Palaiseau cedex, France
*E-mail address*: `{enge,morain}@lix.polytechnique.fr`