MATHEMATICS OF COMPUTATION
Volume 00, Number 0, Xxxx XXXX, Pages 000–000
S 0025-5718(XX)0000-0

AN EFFICIENT ALGORITHM FOR THE COMPUTATION OF GALOIS AUTOMORPHISMS

BILL ALLOMBERT

Abstract. We describe an algorithm for computing the Galois automorphisms of a Galois extension which generalizes the algorithm of [AcKl] to the non-Abelian case. This is much faster in practice than algorithms based on LLL or factorization.

1. Introduction

Let $T \in \mathbb{Z}[X]$ be a monic irreducible polynomial of degree n, α a root of T in a fixed algebraic closure $\overline{\mathbb{Q}}$ of $\mathbb{Q}$ and let $K = \mathbb{Q}(\alpha)$. The aim of this paper is to present an efficient algorithm which checks whether or not K is a Galois extension of $\mathbb{Q}$, and when that is the case, computes the image of α by each automorphism, expressed as a polynomial in α.

Algorithms for doing these tasks already exist in the literature and in computer algebra systems (for example [PARI] and [KANT V4]). They essentially fall into four categories:

- the use of LLL algorithms, either real or p-adic (see for example [Cohen, page 174]),
- explicit factorization of $T(X)$ in the number field K (see for example [Roblot, page 19]),
- the use of unramified p-adic extensions and block system elimination (see [Klüners1]),
- in the Abelian case, the lift of Frobenius automorphisms (see [AcKl]).

All these methods have advantages and disadvantages: methods based on the LLL algorithm are often quite fast but, due to the huge size of the solutions, do not always guarantee the full result in reasonable time. Explicit factorization can be implemented to be polynomial time, but is nonetheless usually rather slow, although it is of course failsafe. The Frobenius lifting method is by far the best practical method, but unfortunately applies only to the Abelian case.

In the present paper, we give an efficient algorithm based on a combination of techniques.

• A combinatorial approach based on the explicit knowledge of the group structure of groups of "small" cardinality (less than 100, say), (even though the Galois group is not known initially); in particular, we will explain how to find a nearly optimal strategy for minimizing the number of necessary Frobenius liftings to find some nontrivial element in the automorphism group.

• The use of previously known elements and known relations to considerably reduce the number of necessary Frobenius liftings to find new elements of the automorphism group.

Received by the editor on June 25, 2001.
1991 *Mathematics Subject Classification.* Primary 11Y40.

• The polynomial chinese remainder theorem which allows us to reconstruct all the necessary Frobenius liftings from a single one.

• Several classical implementation tricks such as the "$d-1$-test" explained below, which are possible thanks to use of the above techniques.

The above combinatorial techniques are mostly unnecessary in the abelian case, hence even though our algorithm can be considered as a modification of the method of [AcKl] which is still valid in the non-Abelian case, its essential ideas are fundamentally different. Even in the Abelian case, for reasonable-sized groups we use a more efficient strategy to find the automorphisms.

The main advantage of our algorithm is that it is much faster than the LLL or factorization methods, and of comparable speed with the Frobenius lifting methods of [AcKl] in the Abelian case. In [Klüners1], one can find a generalization of the Frobenius lifting method to the non-Abelian case. However, the combinatorial techniques mentioned above are not used, hence the number of necessary liftings is much larger than in our algorithm. This is confirmed by the comparison of our timings with the only available implementation of [Klüners1] in Kash given below.

Although the ideas presented in this paper are valid in the general case, we will only present a complete algorithm when the Galois group is supersolvable (see 6.1). The reason for this is that, among the 1048 abstract groups of order less than or equal to 100, only 73 are not supersolvable. Moreover a modification of the algorithm (as implemented in [PARI, function `nfgaloisconj`]) enables us to treat all but 22 groups, the smallest one being of order 36.

2. Representation of automorphisms

In this section, we assume that T is the irreducible polynomial of a primitive element α of a Galois extension K of $\mathbb{Q}$. If φ is a Galois automorphism, we can represent φ in two different ways which will both be useful: the first consists in expressing $\varphi(\alpha)$ as a polynomial in α, and the second in expressing the action of φ as a permutation of the roots of T. We give some algorithmic details.

2.1. Polynomial representation of φ. For algorithmic reasons, it is convenient to identify K with $\mathbb{Q}[X]/(T)$ and α with the class $\overline{X}$ of X modulo T. A first natural representation of a Galois automorphism φ is by the class modulo T of a polynomial S such that

$$\varphi(\overline{X}) = \overline{S} \in \mathbb{Q}[X]/(T) \ .$$

A polynomial S defines such an automorphism if and only if $\overline{S}$ is a root of T in K, in other words if and only if $T \mid T \circ S$. The polynomial S being known, we can compute the image of an element $\overline{P} \in \mathbb{Q}[X]/(T)$ by the formula

$$\varphi(\overline{P}) = \varphi(P(\overline{X})) = P(\varphi(\overline{X})) = \overline{P \circ S} \ .$$

2.2. Permutation representation. Let ℓ be a prime such that the factorization of T modulo ℓ is equal to a product of distinct linear factors, and let $(\alpha_i)_{i=1}^n$ be suitable ℓ-adic approximations of the roots of T. The element $P(\alpha) \in K$ can be given by the ℓ-adic conjugate vector representation $(\beta_i)_{i=1}^n = (P(\alpha_i))_{i=1}^n$.

Note that the only reason for which we use the ℓ-adic conjugate vector representation instead of the complex one is to avoid annoying accuracy problems which can be handled much more easily in the ℓ-adic case. Since the polynomial T is assumed to be Galois, the cost to find a suitable prime ℓ is small. In addition, the first stage of the algorithm will frequently return ℓ as a sub-product.

If φ is an automorphism defined by a polynomial S, there exists a permutation π of the roots such that $S(\alpha_i) = \alpha_{\pi(i)}$, and we have

$$\varphi((\beta_i)_{i=1}^n) = \varphi(P(\alpha)) = (P \circ S)(\alpha) = (P(S(\alpha_i)))_{i=1}^n = (P(\alpha_{\pi(i)}))_{i=1}^n = (\beta_{\pi(i)})_{i=1}^n$$

hence the permutation π determines uniquely the automorphism φ.

Note that it is trivial to compute the product of two automorphisms using the permutation representation, and it is more costly to do that using the polynomial representation.

2.3. **Testing permutations.** If π is a permutation of the roots, we would like to check whether π is the restriction of a Galois automorphism φ and in this case to compute the corresponding polynomial S.

To do this, we need an integer D such that the polynomial DS has integral coefficients and a bound B on these coefficients. A suitable value for D can be a common multiple of the denominators of the entries of the matrix giving an integral basis of K on the powers of α. For B we can either use theoretical bounds which depend only of the coefficients of T as in [AcKl], but practical computations show that it is faster to spend time to compute complex approximation for the roots of T and derive a more accurate bound using the following lemma. Let us denote by $\|V\|$ the L^∞-norm when V is a vector and by $\|M\|$ the functional L^∞-norm when M is a matrix, so that we have the inequality $\|MV\| \leq \|M\|\|V\|$, which is simply the supremum of the L^1-norms of the rows of M.

Lemma 2.1. *Let* $\mathrm{v}_{\mathbb{C}}(T)$ *be the Vandermonde matrix with complex entries associated to the complex roots* $(r_i)_{i=1}^n$ *of* T*, given by*

$$\mathrm{v}_{\mathbb{C}}(T) = (r_i^{j-1}) \in M_n(\mathbb{C}) \ .$$

Then

$$B = D \| \mathrm{v}_{\mathbb{C}}(T)^{-1} \| \sup(|r_i|)_{i=1}^n$$

is a suitable bound for the coefficient of DS*.*

Proof. Let C be the vector $(r_{\pi(i)})_{i=1}^n$ and let V be defined by $DS = \sum_{i=1}^n V_i X^{i-1}$ then $\mathrm{v}_{\mathbb{C}}(T)V = DC$. It follows that

$$\|V\| = \|D\,\mathrm{v}_{\mathbb{C}}(T)^{-1}C\| \leqslant D\|\,\mathrm{v}(T)^{-1}\|\|C\| \leqslant B \ ,$$

proving the lemma. ☐

Now let $\mathrm{v}(T)^{-1}$ be the inverse of the Vandermonde matrix with ℓ-adic entries associated to the roots of T. Note that this matrix can easily be computed explicitly (row i is given by the coefficients of the polynomial $T(X)/(T'(\alpha_i)(X-\alpha_i))$ in reverse order), and this is the only matrix that we will need in practice.

We will assume that the ℓ-adic roots $(\alpha_i)_{i=1}^n$of T have been computed with an ℓ-adic error less than $(2B)^{-1}$. Let C be the vector $(\alpha_{\pi(i)})_{i=1}^n$ and let $\tilde{V}$ be defined by

$$\tilde{V} = D\,\mathrm{v}(T)^{-1}C \ .$$

If the permutation π corresponds to an automorphism, the entries of $\tilde{V}$ must be close to integers which are less than B. Thus,

1. We test if the entries of $\tilde{V}$ are close to integers which are less than B. Since the entries are ℓ-adic, this upper bound gives a bound on the ℓ-adic accuracy.
2. If this condition is satisfied, let $(V_i)_{i=1}^n$ be the vector obtained from $\tilde{V}$ by rounding each entry to the ℓ-adically closest integer less than B, and let $S = \sum_{i=1}^n V_i X^{i-1}$. We then check whether $T \mid T \circ S$.

We perform the computations with a higher accuracy so as to eliminate many more permutations in Step 1, because this step is much faster than Step 2.

Remarks:

- To compute an integral basis is generally slow. Instead, we get a suitable value for D as follows : we partially factor the discriminant of T in the form $\mathrm{Disc}(T) = duf^2$, where d is squarefree and u is not a perfect square and has no small prime factors, then $D = uf$ is a multiple of the index, hence a multiple of the common denominator of the integral basis.
- As already mentioned, we could perform all the computations using complex numbers instead of ℓ-adic numbers but, apart for the time spent in computing ℓ, computations are usually slower, and it is difficult to control rounding errors. Note that we still have to compute the roots of T two times, since we must first compute the needed accuracy.
- We check whether the entries of $\tilde{V}$ are close to integers which are less than B by computing only a single entry $\tilde{V}_{i_0}$ and testing whether this is true, and only then computing the other entries. More importantly, we compute once and for all the n^2 products $(\lambda_i \alpha_j)_{1 \leqslant i,j \leqslant n}$ of the entries $(\lambda_i)_{i=1}^n$ of the i_0-th row of the matrix $D\,\mathrm{v}(T)^{-1}$ by the roots $(\alpha_j)_{j=1}^n$, so that we will need no multiplications when checking the i_0-th entry. In practice, the second entry seems to be a good choice and the first entry a poor one. Furthermore, by using an idea known as the $d-1$–*test* (see [ASZ]), the additions can be performed only in single precision.

3. Testing permutations directly

A simple method for computing the Galois automorphisms would be to test all the possible permutations of the roots using the method explained above. This would require $n!$ tests, hence would be limited to very small values of n. A more complex method consists in testing all the transitive subgroups of $\mathfrak{S}_n$(*not* up to conjugation). In the present section, we give an example of this method for the special case where we want to test whether a polynomial T of degree 12 has Galois group $G \cong \mathfrak{A}_4$ and more importantly to find the Galois automorphisms. We choose this example because it is the smallest non supersolvable group (see Subsection 6.1). The group G acts simply transitively on the roots. An element $g \in G$ acts as a permutation whose cycles have all the same length. Indeed, if this was not the case, if l is the length of the smallest cycle we would have $g^l \neq 1$ and g^l would have fixed points, and this contradicts the fact that the action is simply transitive.

The number of permutations which are products of cycles of equal length l is given by the formula

$$\frac{n!}{(n/l)!\,l^{(n/l)}}\ .$$

In the group $\mathfrak{A}_4$, apart from the identity there exist 3 elements of order 2 and 8 of order 3.

- For $l = 2$, we find 10395 possible permutations.
- For $l = 3$, we find 246400 possible permutations.
- More generally, n being fixed, the number of possible permutations is an increasing function of l.

We can thus find an element σ of order 2 by testing at most 10395 permutations. We must then find the two other automorphisms of order 2, which we will denote by τ and ν which satisfy the relations $\sigma\tau = \tau\sigma = \nu$. Replacing the roots by their index and changing the ordering of the roots, we may assume that the cycle decomposition of σ is $(1,2)(3,4)(5,6)(7,8)(9,10)(11,12)$. For $\tau(1)$, we have 10 possible values, i.e., roots number 3 to 12. Assume first that $\tau(1) = 3$. It follows that $\sigma\tau(1) = 4$, hence

$\tau\sigma(1) = 4$ so $\tau(2) = 4$, and of course also $\tau(3) = 1$ and $\tau(4) = 2$. Thus we know immediately 3 new values of τ. For $\tau(5)$, we only have 6 choices left, i.e., roots number 7 to 12. This will again give us 4 new values, for example if $\tau(5) = 7$ then $\tau(6) = 8$. There remains only 2 choices for $\tau(9)$, i.e., roots number 11 and 12. To summarize, we find τ (hence ν) by testing at most $10 \times 6 \times 2 = 120$ permutations. In fact, possibly after exchanging τ and ν we can always assume that $\tau(1)$ is odd, which only makes 60 permutations to test.

We still need the 8 elements of order 3. One of them, which we will denote by ρ, is such that $\rho\sigma = \tau\rho$ and $\rho\tau = \nu\rho$, and a similar reasoning allows us to find ρ with only 15 tests. Since ρ, σ and τ generate $\mathfrak{A}_4$, we can easily generate the remaining automorphisms. Thus, in the worst case we have tested 10470 permutations. It is important to note that, if done in a naïve way, we would instead need to test up to 12!=479001600 permutations.

For the case of a more general group, the above example shows that essentially all the tests are performed to find the first element. Thus, we must start by computing elements of minimal order. Afterwards, as the above example shows, the number of tests decreases considerably since the number of possible transitive subgroups decreases very rapidly. In the worst case, we need at least $\frac{n!}{(n/2)!2^{(n/2)}}$ tests. With a careful implementation, it is possible to perform the tests very quickly.

As the $\mathfrak{A}_4$ example shows, we need to know precisely the group structure and relations between elements to generate an efficient algorithm, hence it does not seem possible to generate automatically the sequence of permutations to test in the context of a general algorithm.

4. p-ADIC METHODS

4.1. The algorithm of Acciaro and Klüners.

In [AcKl], V. Acciaro and J. Klüners give an algorithm for finding Galois automorphisms in the case where the Galois group is Abelian. It can be summarized as follows. We choose a prime number p such that T has no square factor modulo p, and we want to compute the Frobenius automorphism φ corresponding to p. The following result is well known.

Proposition 4.1. *If p is a prime number such that T has no square factor modulo p, then*

$$\mathbb{Z}[X]/(p,T) \cong \mathbb{Z}_K/p\mathbb{Z}_K \ .$$

Their method is based on the following algorithms.

Algorithm 4.2. *Under the above hypotheses on T and p, let $S_0 \in \mathbb{Z}[X]$ such that $T \circ S_0 \equiv 0 \pmod{(p,T)}$. There exists a unique sequence of polynomials $(S_k)_{k\geqslant 1} \subset \mathbb{Z}[X]$ such that*

$$T \circ S_k \equiv 0 \pmod{(p^{2^k},T)} \text{ and } S_k \equiv S_0 \pmod{(p,T)}$$

which is obtained using the recurrence relation

$$S_{k+1} \equiv S_k - \frac{T \circ S_k}{T' \circ S_k} \pmod{(p^{2^{k+1}},T)} \ .$$

Algorithm 4.3. *Under the above hypotheses on T and p, let $S_0 \in \mathbb{Z}[X]$ such that $T \circ S_0 \equiv 0 \pmod{(p,T)}$. We find explicitly an automorphism σ, if it exists, such that $\sigma(x) \equiv S_0(x) \pmod{p\mathbb{Z}_K}$ for all $x \in \mathbb{Z}_K$.*

1. *Apply algorithm 4.2 until we have $p^{2^k} > 2B$.*
2. *Lift DS_k to $\tilde{S}$ by choosing coefficients in $\{1 - p^{2^k}/2, \dots, p^{2^k}/2\}$ and set $S = \tilde{S}/D$.*

3. *Check whether $T \mid T \circ S$. If this is the case, return the polynomial S. Otherwise, return "No solution".*

If $K/\mathbb{Q}$ is an Abelian extension, we know that there exists a Frobenius automorphism $\varphi = \left(\frac{p}{K/\mathbb{Q}}\right)$ such that $\varphi(x) \equiv x^p \pmod{p\mathbb{Z}_K}$ hence we can use Algorithm 4.3 with $S_0 = X^p$. If this algorithm fails, $K/\mathbb{Q}$ is not Abelian, and if it succeeds we have computed an automorphism and we start again with another value of p until the automorphisms that we have obtained generate n distinct automorphisms.

4.2. **Generalization to the non-Abelian case.** We now want to generalize the above method to the non-Abelian case. We assume that the extension $K/\mathbb{Q}$ is Galois with Galois group G, and we let p be as above. If $\overline{T} = \prod_{i=1}^{g} \overline{T}_i \pmod{p}$ is the factorization of T modulo p and $\mathfrak{P}_i = (p, T_i(\alpha))$, we know that $p\mathbb{Z}_K = \prod_{i=1}^{g} \mathfrak{P}_i$ is the splitting of p in $\mathbb{Z}_K$. Let $\mathbb{F}_{\mathfrak{P}_i} = \mathbb{Z}_K/\mathfrak{P}_i$ be the residue fields, $\varphi_i = \left(\frac{\mathfrak{P}_i}{K/\mathbb{Q}}\right)$ the corresponding Frobenius automorphisms, and $f = \dim_{\mathbb{F}_p} \mathbb{F}_{\mathfrak{P}_i}$ the residual degree of p.

Proposition 4.4. *The map*

$$\Gamma \left| \begin{array}{lcl} \mathrm{Gal}(K/\mathbb{Q}) & \longrightarrow & \mathrm{Aut}(\mathbb{Z}_K/p\mathbb{Z}_K) \cong \mathrm{Aut}(\prod_{i=1}^{g} \mathbb{F}_{\mathfrak{P}_i}) \\ \sigma & \longmapsto & (x + p\mathbb{Z}_K \mapsto \sigma(x) + p\mathbb{Z}_K) \end{array} \right.$$

is an injective group homomorphism.

Proof. Let $\sigma \in \mathrm{Ker}(\Gamma)$. In particular $\sigma(x) \equiv x \pmod{\mathfrak{P}_1}$ for all $x \in \mathbb{Z}_K$, so that σ belongs to the inertia group of $\mathfrak{P}_1$ which is the trivial group since $\mathfrak{P}_1$ is unramified in $K/\mathbb{Q}$. $\square$

The following proposition gives the structure of $\mathrm{Aut}(\prod_{i=1}^{g} \mathbb{F}_{\mathfrak{P}_i})$.

Proposition 4.5. *Let $\iota_{i,j}$ be an isomorphism from $\mathbb{F}_{\mathfrak{P}_i}$ to $\mathbb{F}_{\mathfrak{P}_j}$. The group*

$$\mathrm{Aut}(\prod_{i=1}^{g} \mathbb{F}_{\mathfrak{P}_i})$$

is isomorphic to the wreath product $(\mathbb{Z}/f\mathbb{Z}) \wr \mathfrak{S}_g = (\mathbb{Z}/f\mathbb{Z})^g \rtimes \mathfrak{S}_g$, of order $f^g g!$, where multiplication is given by

$$((a_i)_{i=1}^{g}; \sigma)((b_i)_{i=1}^{g}; \tau) = ((a_i + b_{\sigma(i)})_{i=1}^{g}; \sigma\tau) \ .$$

Explicitly, we have

$$\Delta \left| \begin{array}{lcl} (\mathbb{Z}/f\mathbb{Z}) \wr \mathfrak{S}_g & \longrightarrow & \mathrm{Aut}(\prod_{i=1}^{g} \mathbb{F}_{\mathfrak{P}_i}) \\ ((a_i)_{i=1}^{g}; \sigma) & \longmapsto & ((x_i)_{i=1}^{g} \mapsto (\iota_{\sigma(i),i}(x_{\sigma(i)})^{p^{a_i}})) \end{array} \right. .$$

Proof. Let $\rho \in \mathrm{Aut}(\mathbb{Z}_K/p\mathbb{Z}_K)$. The maximal ideals of $\mathbb{Z}_K/p\mathbb{Z}_K$ are exactly the $(\mathfrak{P}_i/p\mathbb{Z}_K)_{i=1}^{g}$. The image of a maximal ideal by an automorphism is again a maximal ideal, hence ρ permutes the $\mathfrak{P}_i$. Thus, there exists $\sigma \in \mathfrak{S}_g$ such that $\rho(\mathfrak{P}_i/p\mathbb{Z}_K) = \mathfrak{P}_{\sigma(i)}/p\mathbb{Z}_K$. Let $\mathfrak{Q}_i = \bigcap_{j \neq i} \mathfrak{P}_j$, so that $\mathfrak{Q}_i/p\mathbb{Z}_K \cong \mathbb{F}_{\mathfrak{P}_i}$ and $\rho(\mathfrak{Q}_i) = \mathfrak{Q}_{\sigma(i)}$. Thus, $\iota_{i,\sigma(i)}^{-1} \circ \rho$ is an automorphism of $\mathbb{F}_{\mathfrak{P}_i}$, and since $\mathrm{Gal}(\mathbb{F}_{\mathfrak{P}_i}/\mathbb{F}_p)$ is generated by the Frobenius automorphism $x \mapsto x^p$, there exists $a_i \in \mathbb{Z}/f\mathbb{Z}$ such that $\iota_{i,\sigma(i)}^{-1} \circ \rho = x \mapsto x^{p^{a_i}}$, from which we obtain $\rho((x_i)_{i=1}^{g}) = (\iota_{\sigma^{-1}(i),i}(x_{\sigma^{-1}(i)})^{p^{a_i}})$. $\square$

Thanks to Algorithm 4.3 we can check if an element belongs to $\mathrm{Im}(\Gamma)$ and find its inverse image by Γ if it does. It naturally leads to a second algorithm : given an integer v which is expected to be the order of some automorphism, apply Algorithm 4.3 to every element of order v of $\mathrm{Aut}(\mathbb{Z}_K/p\mathbb{Z}_K)$. However the number of elements to test becomes quickly large, and this method is restricted to polynomials of small degree. In fact using the idea of Section 3 it is possible to greatly reduce the number of tests, and we have thus designed an algorithm for the group $\mathfrak{S}_4$. This method needing knowledge of the group structure and of relations between elements, it does not lead to a general algorithm.

5. An algorithm using the two methods

5.1. **Lift of diagonal elements.** The idea of this section is to lift diagonals automorphisms.

Definition 5.1. *We will say that an automorphism $\sigma \in G$ is a **diagonal** automorphism (with respect to p) if $\Delta^{-1} \circ \Gamma(\sigma) \in (\mathbb{Z}/f\mathbb{Z})^g \times \{1\}$.*

Lemma 5.2. *The set of diagonal automorphisms is equal to the intersection of the decomposition groups of the prime ideals above p.*

Proof. The automorphism σ is diagonal if and only if for all $1 \leqslant i \leqslant g$ there exists a_i such that $\sigma(x) \equiv x^{p^{a_i}} \pmod{\mathfrak{P}_i}$, which means that σ belongs to $\langle\varphi_i\rangle$ which is exactly the decomposition group of $\mathfrak{P}_i$. □

Let b be an positive integer and $a \in \mathbb{Z}/b\mathbb{Z}$. In the sequel, let us denotes by $\underline{a}$ the only non-negative integer in the class a strictly less than b. The following proposition allows us to find the diagonal automorphisms.

Proposition 5.3. *There exists a diagonal automorphism $\sigma \neq id$ if and only if there exists a divisor d of f and different from f such that $\langle\varphi_1^d\rangle \trianglelefteq G$. In addition, if this condition is satisfied we have:*

(i). *There exists a map $\psi : \{1,\ldots,g\} \longrightarrow (\mathbb{Z}/\frac{f}{d}\mathbb{Z})^*$ such that*

$$\forall i \in \{1,\ldots,g\} \quad \sigma = \varphi_i^{d\,\underline{\psi(i)}} .$$

(ii). *The set $\mathrm{Im}(\psi)$ is a subgroup of $(\mathbb{Z}/\frac{f}{d}\mathbb{Z})^*$ and all the elements of $\mathrm{Im}(\psi)$ have the same number of preimages.*

Proof. If σ is a diagonal automorphism different from the identity then $\sigma \in \langle\varphi_i\rangle$ hence $\sigma = \varphi_i^{a_i}$. Since the φ_i have all the same order, there exists d and $\psi(i) \in (\mathbb{Z}/\frac{f}{d}\mathbb{Z})^*$ such that σ is of order f/d and $d\,\underline{\psi(i)} = a_i$. If $\tau \in G$ there exists i such that

$$\tau\varphi_1^d\tau^{-1} = \varphi_i^d = \varphi_1^{d\,\underline{\psi(1)/\psi(i)}} \in \langle\varphi_1^d\rangle \tag{1}$$

hence $\langle\varphi_1^d\rangle \trianglelefteq G$.
Conversely, if there exists $d \mid f, d \neq f$ such that $\langle\varphi_1^d\rangle \trianglelefteq G$, we set $\sigma = \varphi_1^d$ and we let τ be such that $\tau\varphi_1\tau^{-1} = \varphi_i$. Then there exists $\psi(i) \in (\mathbb{Z}/\frac{f}{d}\mathbb{Z})^*$ which does not depend on τ such that

$$\tau\varphi_1^d\tau^{-1} = \varphi_i^d = \varphi_1^{d\,\underline{\psi(i)^{-1}}}$$

hence $\varphi_1^d = \varphi_i^{d\,\underline{\psi(i)}}$ belongs to the decomposition group of $\mathfrak{P}_i$.
Finally, we must prove the stated properties of ψ, which factors as follows.

$$\begin{array}{ccccc} \{1,\ldots,g\} & \overset{\psi_1}{\longleftrightarrow} & G/\langle\varphi_1\rangle & \overset{\psi_2}{\twoheadrightarrow} & G/C_G(\langle\varphi_1^d\rangle) \\ i & \mapsto & \{\tau;\tau(\mathfrak{P}_1) = \mathfrak{P}_i\} & \mapsto & \{\tau;\tau\varphi_1^d\tau^{-1} = \varphi_i^d\} \end{array}$$

$$\begin{array}{ccccc} G/C_G(\langle\varphi_1^d\rangle) & \overset{\psi_3}{\hookrightarrow} & \mathrm{Aut}(\langle\varphi_1^d\rangle) & \overset{\psi_4}{\longleftrightarrow} & (\mathbb{Z}/\frac{f}{d}\mathbb{Z})^* \\ \{\tau;\tau\varphi_1^d\tau^{-1}=\varphi_i^d\} & \mapsto & \mathrm{Int}_\tau & \mapsto & \psi(i) \end{array},$$

where $C_G(\langle\varphi_1^d\rangle)$ is the centralizer of $\langle\varphi_1^d\rangle$ and Int_τ is the inner automorphism corresponding to τ. In addition, $\mathrm{Ker}(\tau \mapsto \mathrm{Int}_\tau) = C_G(\langle\varphi_1^d\rangle)$, showing that ψ_3 is a well-defined injective group homomorphism. The map ψ_1 corresponds to the bijection from $G/\operatorname{Stab}_{\mathfrak{P}_1}$ to $\mathrm{Orb}_{\mathfrak{P}_1}$, where $\operatorname{Stab}_{\mathfrak{P}_1}$ is the stabilizer of $\mathfrak{P}_1$ in the set of prime ideals above p and $\mathrm{Orb}_{\mathfrak{P}_1}$ is the orbit of $\mathfrak{P}_1$ in the same set. The map ψ_4 is trivially a group isomorphism. The map ψ_2 is the canonical projection from $G/\langle\varphi_1\rangle$ to $G/C_G(\langle\varphi_1^d\rangle)$, hence it is surjective and all the elements of $\mathrm{Im}(\psi_2)$ have the same number of preimages by ψ_2. Thus the same is true for ψ, and $\mathrm{Im}(\psi) = \mathrm{Im}(\psi_3)$ is a subgroup. □

Algorithm 5.4. *Let T and p as before, and let d be a divisor of f different from f. We find φ_1^d when $\langle\varphi_1^d\rangle \trianglelefteq G$.*

1. *For each subgroup $H \subseteq (\mathbb{Z}/\frac{f}{d}\mathbb{Z})^*$ of order h dividing the number g of prime ideals above p and for every surjective map ψ from $\{1,\dots,g\}$ to H such that $\underline{\psi}(1) = 1$ and such that every element of H has g/h preimages, do the following.*
   (a) *Use the Chinese remainder theorem to find S_0 such that*
   $$\forall i \in \{1,\dots,g\} \quad S_0 \equiv x^{p^{d\,\underline{\psi(i)}}} \pmod{\mathfrak{P}_i}\ .$$
   (b) *Apply Algorithm 4.3.*
   (c) *If it succeeds, return S and terminate.*
2. *Return "$\langle\varphi_1^d\rangle \ntrianglelefteq G$ or T not Galois" and terminate.*

For a given subgroup H of order h, there are $\frac{g!}{h(g/h)!^h}$ maps ψ to be tested. In practice, if the number of maps is large, the following algorithm is much faster.

Algorithm 5.5. *Let T and p as before, and let d be a divisor of f different from f. We find φ_1^d when $\langle\varphi_1^d\rangle \trianglelefteq G$.*

1. *Find N such that $p^N > 2B$*
2. *Apply Algorithm 4.2 to $S_0 \equiv x^{p^d} \pmod{(p,T)}$ until $2^k \geqslant N$.*
3. *Let $U_1 \equiv S_k \pmod{(p^N,T)}$ and compute $(U_k)_{k=2}^{f/d}$ given by the formula*
   $$U_{k+1} \equiv U_k \circ U_1 \pmod{(p^N,T)}$$
   .
4. *Lift the factorization $\overline{T} = \prod_{i=1}^g \overline{T}_i \pmod p$ to a factorization $T \equiv \prod_{i=1}^g T_i \pmod{p^N}$ so that $\mathbb{Z}_K/\mathfrak{P}_i^N \cong \mathbb{Z}[X]/(p^N,T_i)$.*
5. *Compute polynomials $(V_i)_{i=1}^n$ so that $V_i \equiv \delta_i^j \pmod{T_j,p^N}$ for all $(i,j) \in \{1,\dots,g\}^2$, usually as a subproduct of the previous factorization.*
6. *For each subgroup $H \subseteq (\mathbb{Z}/\frac{f}{d}\mathbb{Z})^*$ of order h dividing the number g of prime ideals above p and for every surjective map ψ from $\{1,\dots,g\}$ to H such that $\underline{\psi}(1) = 1$ and such that every element of H has g/h preimages, do the following.*
   (a) *Compute*
   $$S \equiv \sum_{i=1}^{g} V_i U_{\underline{\psi(i)}} \pmod{(p^N)}\ .$$
   (b) *if $T \mid T \circ S$ then return S and terminate.*
7. *Return "$\langle\varphi_1^d\rangle \ntrianglelefteq G$ or T not Galois" and terminate.*

Note that the idea for pruning the tests given in Subsection 2.3 still apply, by precomputing the products V_iU_j and then by trying only one coefficient of S first, with the $d-1$*–test*. Moreover, before testing if $T \mid T \circ S$, we can check whether $(S(\alpha_i))_{i=1}^n$ is a permutation of the approximates roots $(\alpha_i)_{i=1}^n$ of T, which is much faster.

5.2. **Example of a complete computation of the Galois automorphisms.** Before giving a general algorithm, we will show on an example how the above method combined with permutation tests allows us to finish the computation of the Galois automorphisms. Assume for example that we know that the Galois group G is isomorphic to the dihedral group of order n, and assume that we know a prime number p such that $2f = n$, where $f = f_p$ is the residual degree of the prime ideals above p. Algorithm 5.4 combined with a single application of Algorithm 4.3 allows us to find an element ρ of order f. We then need to find an element $\sigma \notin \langle\rho\rangle$. Our hypothesis on G implies that σ is of order two and that $\sigma\rho\sigma^{-1} = \rho^{-1}$. We reorder the roots so that the cycle decomposition of the permutation corresponding to ρ is

$$((\alpha_1, \dots, \alpha_f)(\alpha_{f+1}, \dots, \alpha_n)) \ .$$

Since G acts transitively, we can assume that $\sigma(\alpha_1) = \alpha_n$. Since $\sigma\rho(\alpha_1) = \rho^{-1}\sigma(\alpha_1)$, we must have $\sigma(\alpha_2) = \alpha_{n-1}$ and by induction we find $\sigma(\alpha_i) = \alpha_{n+1-i}$, which completely determines σ, and terminates the computation with very little extra work. This is the method that we will generalize in the next section.

5.3. **Computation of the other automorphisms using permutations.** We assume that Algorithm 5.4 has succeeded and that it has given us $\sigma = \varphi_1^d$ and the corresponding map ψ. The next step is to compute a polynomial whose splitting field is K^σ. We denote by s the order of σ

We use the following result which follows from [KlMa].

Proposition 5.6. *Let $\mathcal{A}$ be the set of roots of T in a fixed algebraic closure, and $\{O_i;\ i = 1 \text{ to } k\}$ be the set of orbits of $\mathcal{A}$ under the action of σ. Let Σ be a symmetric polynomial in s indeterminates and integral coefficients. We set for $1 \leqslant i \leqslant k$*

$$r_i = \Sigma(o_1, o_2, \dots, o_s) \text{ where } O_i = \{o_1, o_2, \dots, o_s\}$$

and

$$R = \prod_{i=1}^{k} (X - r_i) \ .$$

Assume that the polynomial R is squarefree. Then R is monic, irreducible and defines K^σ. For an automorphism $\tau \in G$, we have $\tau(r_i) = r_j$ if and only if $\tau(O_i) = O_j$.

There exists exactly one polynomial of degree at most $n-1$ such that for each $1 \leqslant i \leqslant k$ and each $\alpha \in O_i$ we have $M(\alpha) = r_j$. For this polynomial, $T \mid R \circ M$ holds and it defines the inclusion from K^σ to K.

To apply this result, we need to have approximations of the roots with sufficient accuracy and to find a symmetric polynomial Σ such that the resulting R is squarefree. A deterministic way to find Σ is given in [Klüners2, Lemma 32]. To reduce the needed accuracy on the roots, it is better to find a polynomial Σ of small degree. In practice, it may also be convenient to choose Σ so that R is squarefree modulo a specific prime or prime power. Note that Lemma 43 in [Klüners2] is false, since Lemma 42 applies only to irreducible polynomials.

Thus, we find a symmetric polynomial Σ such that the resulting polynomial R is squarefree and even squarefree modulo p and we recursively apply our algorithm to R to obtain the group of Galois automorphisms $\mathrm{Gal}(K^\sigma/\mathbb{Q}) = G/\langle\sigma\rangle$. We must now lift $\overline{\tau} \in G/\langle\sigma\rangle$ to an element $\tau \in G$. For this, we will test permutations as explained in Subsection 2.3. To reduce the number of permutations to be tested, we search for relations between σ and τ in the group $\mathfrak{S}_n$.

Proposition 5.7. *Let us denote by t the order of $\overline{\tau}$ and by $s = f/d$ the order of σ.*
There exist $(\overline{u}, \overline{v}) \in (\mathbb{Z}/s\mathbb{Z})^2$ such that

$$\tau\sigma\tau^{-1} = \sigma^u \tag{2}$$

$$\tau^t = \sigma^v \tag{3}$$

moreover,

(i). *We have $(u-1)v \equiv 0 \pmod{s}$,*
(ii). *The element $\overline{u}$ does not depend on the choice of the lift τ of $\overline{\tau}$,*
(iii). *If we denote $w = GCD(\sum_{i=1}^t u^i, s)$, then the class of v modulo w does not depends of the choice of the lift τ of $\overline{\tau}$,*
(iv). *We have $\overline{u} = \psi(1)/\psi(c)$, where c is such that $\tau(\mathfrak{P}_1) = \mathfrak{P}_c$.*

Proof. The subgroup $\langle\sigma\rangle$ being normal, $\tau\sigma\tau^{-1} \in \langle\sigma\rangle$ and there exists one and only one $\overline{u} \in \mathbb{Z}/s\mathbb{Z}$ such that (2) holds. The equality $\overline{\tau}^t = 1$ implies that $\tau^t \in \langle\sigma\rangle$. Thus there exists $\overline{v} \in \mathbb{Z}/s\mathbb{Z}$ such that (3) holds. Since

$$\sigma^{uv} = (\tau\sigma\tau^{-1})^v = \tau\sigma^v\tau^{-1} = \tau\tau^t\tau^{-1} = \sigma^v \quad ,$$

we have $\sigma^{(u-1)v} = 1$ hence $(u-1)v \equiv 0 \pmod{s}$. Let $\rho = \tau\sigma^k$ be another representative of $\overline{\tau}$. Then

$$\rho\sigma\rho^{-1} = \tau\sigma^k\sigma\sigma^{-k}\tau^{-1} = \tau\sigma\tau^{-1} = \sigma^u \quad ,$$

hence u is independent of k. On the other hand,

$$\rho^t = \prod_{i=1}^t \tau\sigma^k = \prod_{i=1}^t \tau^{-(i-1)}\tau^i\sigma^k = (\prod_{i=1}^t \tau^i\sigma^k\tau^{-i})\tau^t = (\prod_{i=1}^t \sigma^{ku^i})\tau^t = \sigma^{\sum_{i=1}^t u^i}\sigma^v$$

hence $\rho^t = \sigma^{v+w'k}$ with $w' = \sum_{i=1}^t u^i$, so that the class of v modulo $w = GCD(w', s)$ is independent of k. Finally, if c is such that $\tau(\mathfrak{P}_1) = \mathfrak{P}_c$, identity (1) gives $\tau\sigma\tau^{-1} = \sigma^{\psi(1)/\psi(c)}$ and we deduce $\overline{u} = \psi(1)/\psi(c)$. $\square$

We can compute u explicitly by using the following algorithm.

Algorithm 5.8. *With the notation of Proposition 5.6, if R is squarefree and is squarefree modulo p and $\overline{\tau}$ is given, we find an integer c such that $\tau(\mathfrak{P}_1) = \mathfrak{P}_c$.*

1. *Factor $\overline{R}$ modulo p as $\overline{R} = \prod_{i=1}^g \overline{R}_i \pmod{p}$.*
2. *By trying successive values of $m \in \{1, \dots, g\}$, find m such that $\overline{T}_1 \mid \overline{R}_m \circ M$, where M is as in Proposition 5.6.*
3. *Compute $\overline{R}_d = GCD(\overline{R}, \overline{R}_m \circ \overline{\tau})$*
4. *By trying successive values of $c \in \{1, \dots, g\}$, find c such that $\overline{T}_c \mid \overline{R}_d$.*
5. *Return c.*

Proof. Since σ belong to the decomposition group of $\mathfrak{P}_i$ in $K/\mathbb{Q}$ which is unramified, the prime ideal $\mathfrak{Q}_i = \mathfrak{P}_i \cap \mathbb{Z}_{K^\sigma}$ of K^σ is inert in K, and this is true for all i. We get $p\mathbb{Z}_{K^\sigma} = \prod_{i=1}^g \mathfrak{Q}_i$ and the identity $\tau(\mathfrak{P}_1) = \mathfrak{P}_c$ holds if and only if $\overline{\tau}(\mathfrak{Q}_1) = \mathfrak{Q}_c$. Step 2 finds m such that $\mathfrak{Q}_1 = (p, R_m)$, Step 3 finds R_d such that $\overline{\tau}(\mathfrak{Q}_1) = (p, R_d)$ and Step 4 finds c such that $\mathfrak{Q}_c = (p, R_d)$. $\square$

Proposition 5.9. *Let σ, $\overline{\tau}$, u and v defined as above. Let $o = n/st$ and $(C_i)_{i=1}^{o}$ be the orbits of $\overline{\tau}$ acting on the roots of R, We can specify the roots of T using three coordinates (see Example 5.10)*

$$[a, b, c] \in \mathbb{Z}/o\mathbb{Z} \times \mathbb{Z}/t\mathbb{Z} \times \mathbb{Z}/s\mathbb{Z}$$

where $[a, b, c]$ stands for the c-th element of the orbit O_i and r_i is the b-th element of C_a, where O_i and r_i are defined in Proposition 5.6. Then we have the following rules which are immediate consequences of (2).

$$\sigma([a,b,c]) = [a,b,c+1] \tag{4}$$

$$\tau([a,b,c]) = [a,b+1,d] \;\Rightarrow\; \tau([a,b,e]) = [a,b+1,u(e-c)+d] \tag{5}$$

Assume chosen the $\tau([a,b,0])$ for all pairs (a,b) such that $b \neq t-1$. Then equation (3) determines $\tau([a,t-1,0])$ and implication (5) determines the other values of τ.

Example 5.10. *Assume σ and τ correspond to the following permutations*

$$\sigma = (1\ 2\ 3\ 4\ 5)(6\ 7\ 8\ 9\ 10)\quad (11\ 12\ 13\ 14\ 15)$$
$$(16\ 17\ 18\ 19\ 20)(21\ 22\ 23\ 24\ 25)(26\ 27\ 28\ 29\ 30)$$

$$\overline{\tau} = (123)(456)$$

with $o = 2, t = 3$ and $s = 5$. The numbers represented by the three-index coordinates are, in lexicographic order:

$$\begin{array}{ccc} (1\ 2\ 3\ 4\ 5\ ) & (6\ 7\ 8\ 9\ 10\ ) & (11\ 12\ 13\ 14\ 15\ ) \\ (16\ 17\ 18\ 19\ 20) & (21\ 22\ 23\ 24\ 25) & (26\ 27\ 28\ 29\ 30) \end{array}$$

where, for example

$$[0,0,0] = 1;\ [0,0,1] = 2;\ [0,1,0] = 6;\ [1,0,0] = 16;\ [1,1,4] = 25\ .$$

Algorithm 5.11. *Let σ, $\overline{\tau}$ and u as above. This algorithm lifts $\overline{\tau}$ to an element τ of G.*

1. *Compute the cycle decomposition of σ as a permutation of the roots of T, and of $\overline{\tau}$ as a permutation of the roots of R.*
2. *Order the cycles and renumber the roots using 3 components as in Proposition 5.9.*
3. *Compute u and w thanks to Proposition 5.7 and Algorithm 5.8.*
4. *For each v such that $0 \leqslant v < w$ and for each map C from $\{0, \ldots, o-1\} \times \{0, \ldots, t-2\}$ to $\{0, \ldots, s-1\}$ do as follows.*
   (a) *Using Proposition 5.9, compute τ such that*

   $$\forall (a,b) \in \{0,\ldots,o-1\} \times \{0,\ldots,t-2\} \quad \tau([a,b,0]) = [a,b+1,C(a,b)]\ .$$

   (b) *Test if τ corresponds to an element of the Galois group of T.*
   (c) *If this is the case, return τ and terminate.*
5. *Return "G not Galois" and terminate.*

Note that for each u, the number of maps to test is at most equal to $s^{(t-1)(n/st)} = s^{n/s(1-1/t)}$, so it is important to keep t as small as possible. One way to do this is to lift some powers of $\overline{\tau}$ as in the following example: suppose $\overline{\tau}$ is of order 6. It is much faster to lift $\overline{\tau}^2$ and $\overline{\tau}^3$ and then to compute $\tau = (\tau^2)^2\tau^3$ than to lift $\overline{\tau}$ directly. In fact, using the ideas of Section 3, we can even reduce further the number of tests.

6. The general algorithm

6.1. **Supersolvable groups.** Algorithm 5.4 assumes that there exists at least one non-trivial cyclic normal subgroup in G. Since we use this algorithm inductively, this limits the general algorithm to supersolvable groups defined as follows.

Definition 6.1 (Supersolvable groups). *A finite group G is supersolvable if there exists an increasing sequence of subgroups $(H_i)_{i=0}^{k}$ such that*

1. *We have $H_0 = \{1\}$ and $H_k = G$*
2. *For each $0 \leqslant i < k$, the subgroup H_i is a normal subgroup of G*
3. *For each $0 \leqslant i < k$, the quotient H_{i+1}/H_i is cyclic.*

For $n = |G| \leqslant 100$, almost all groups are supersolvable: using the computer algebra system [Magma], it appears that 975 out of 1048 possible groups are supersolvable, and non supersolvable groups of order less than or equal to 100 have order a multiple of 12, or have order 56, 75 or 80. The group $\mathfrak{A}_4$ is the smallest non supersolvable group, of order 12.
We will assume the following structure theorem (see [Hall, page 169]).

Theorem 6.2. *Let G be a supersolvable group of order*

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

with $p_1 \geqslant p_2 \geqslant \ldots \geqslant p_r$ prime. Then there exists a sequence $(h_i)_{i=1}^{r}$ of elements of G such that for all $1 \leqslant i < r$, the subgroup H_i generated by $(h_j)_{j=1}^{i}$ is a normal subgroup of G of order

$$C_i = \prod_{j=1}^{i} p_j \quad .$$

In addition, any increasing sequence of normal subgroups of G can be refined to a sequence of the above type.

Corollary 6.3. *If m if an integer such that $1 \leqslant m < r$ and $p_m \neq p_{m+1}$, then there exists exactly one subgroup of order C_m and it is a normal subgroup. In addition, any subgroup whose order divides C_m is a subgroup of H_m.*

Corollary 6.4. *The p_1-Sylow subgroup of G is unique hence normal.*

We will also use the following easy result.

Theorem 6.5. *Let G be a finite group and p be the smallest prime divisor of the order of G. Then all the subgroups of index p are normal.*

If we find an element g of order C_m in G with m as in Corollary 6.3, then it generates H_m which is a normal subgroup, hence all the elements of C_m are powers of g. In addition, if we find that the group generated by g is not a normal subgroup, this shows that G is not supersolvable.

6.2. **The general algorithm in detail.**

Algorithm 6.6. *Let $T \in \mathbb{Z}[X]$ be a monic irreducible polynomial. If T is Galois with a supersolvable Galois group, the algorithm returns Galois automorphisms $(\sigma_i)_{i=1}^{r}$ of G such that for all $1 \leqslant m \leqslant r$ we have $\langle\sigma_i\rangle_{i=1}^{m} \trianglelefteq G$ and $\langle\sigma_i\rangle_{i=1}^{r} = G$. If not, the algorithm may terminate with a message indicating that T is not Galois or not supersolvable, or may not terminate at all.*

1. *Set $m_{\max} := 0$, $p_{\max} := 0$ and*

$$M := \{0 \leqslant m \leqslant r;\ m = 0 \text{ or } m \geqslant r-1 \text{ or } (1 \leqslant m < r-1 \text{ and } p_m \neq p_{m+1})\} \quad .$$

2. *For the first* $3n/2$ *prime numbers* $p > 2n$*, factor* T *modulo* p.
   (a) *If* T *has a square factor modulo* p*, choose another* p.
   (b) *If the factors of* T *do not have the same degree, return "T is not Galois" and terminate.*
   (c) *If the factors have the same degree, let* f *be the common degree of the factors of* T *modulo* p*. Compute the largest* $m \in M$ *such that* $C_m \mid f$.
   (d) *If* $m > m_{\max}$*, then set* $m_{\max} = m$ *and* $p_{\max} = p$.
3. *At the end of the tests, we have two possibilities.*
   - $m_{\max} > 0$ *(Good case)*
     (a) *Apply Algorithm 5.4 to* T *and* $p_{\max}$ *with* $d = f/C_{m_{\max}}$.
     (b) *If it fails, return "T is not Galois or not supersolvable" and terminate.*
   - $m_{\max} = 0$ *(Bad case)*
     (a) *Choose* p *such that* $p_1 \mid f_p$.
     (b) *Apply Algorithm 5.4 to* T *and* p *with* $d = f/p_1$.
     (c) *If Algorithm 5.4 fails, we start again in (a) with another* p.

   At the end of this step, Algorithm 5.4 gives us an automorphism $\sigma = \varphi_1^d$.
4. *If* σ *is of order* n*, return* σ *and terminate.*
5. *If* σ *is of order less than* n*, apply Proposition 5.6 to compute a suitable polynomial* R.
6. *Applying recursively the present algorithm to the polynomial* R*, find a family* $(\overline{\tau_1}, \dots, \overline{\tau_t})$ *which generates* $G/\langle\sigma\rangle$.
7. *Using Algorithm 5.11, compute* $(\tau_1, \dots, \tau_t)$*, return* $(\sigma, \tau_1, \dots, \tau_t)$ *and terminate.*

The first stage of the algorithm is essential, and the value $3n/2$ is quite arbitrary. Numerical experimentation seems to show that the value $n/2$ is sufficient for large values of n. This stage enables us to get the structure of the Galois group with high probability, and to choose optimally the order of the lift and in particular the subgroup H in Algorithm 5.4.

The algorithm may loop indefinitely only if T is Galois of Galois group G not supersolvable and whose Sylow subgroups corresponding to the largest prime factor of n are not cyclic. In practice, for $n \leqslant 100$ this happens only for $n = 36$, 72 and 75, and can be predicted with high probability during the first stage.

On the other hand, if the algorithm fails in the middle of the recursive process, it is possible to use another algorithm such as those mentioned in the preceding sections at that point, and then terminate the recursion. For instance, if T has Galois group isomorphic to $\mathbb{Z}/5\mathbb{Z} \times \mathfrak{A}_4$, the algorithm easily finds an element of order 5 and reduces the problem to the computation of the Galois automorphisms of a polynomial of degree 12 with Galois group $\mathfrak{A}_4$, which can easily be done, for example using the method of Section 3. The problem appearing with this kind of group is that we need much more detailed knowledge on the structure of groups of a given order than that given by Theorem 6.2

7. A complete example

In this section, we will compute the Galois automorphisms of the polynomial

$$\begin{aligned} T = x^{21} &- 7x^{20} - 21x^{19} + 238x^{18} - 245x^{17} - 1848x^{16} + 4732x^{15} + 1861x^{14} \\ &-18536x^{13} + 16856x^{12} + 14819x^{11} - 32431x^{10} + 8897x^9 + 16660x^8 \\ &-13533x^7 + 392x^6 + 3514x^5 - 1547x^4 + 161x^3 + 49x^2 - 14x + 1 \end{aligned}$$

with nonabelian Galois group, but we do not want to use this knowledge in the computation.

First we write $21 = p_1 p_2$ with $p_1 = 7$ and $p_2 = 3$. We set $M = \{0, 1, 2\}$.

Then we factor T modulo the first primes greater than 42. We get

p	43	47	53	59	61	67	71	73	79	83	89
f_p	7	3	3	3	3	3	7	3	3	7	3
$\sup\{m \in M; C_m \mid f_p\}$	1	0	0	0	0	0	1	0	0	1	0

We are in a good case as $m_{\max} = 1$. We have not found any primes with $f_p = 1$, so we continue the factorizations until finding that $p = 449$ is such a p, so we set $\ell = 449$. We compute the discriminant of T and B

$$\operatorname{Disc}(T) = (2)^{18}(7)^{32}(181)^6(2339)^6 = D^2$$

$$B \leqslant 10^{44}$$

We compute the roots of T modulo ℓ, $\{2, 35, 45, 51, 66, 79, 96, 109, 174, 180, 223, 225, 236, 301, 305, 342, 370, 373, 397, 440, 448\}$ and we lift them to the precision ℓ^{17} so that $\ell^{17} > 2B$. We set $p = p_{\max} = 43$, $f = f_p = 7$, $g = g_p = 3$ and $d = 1$, and we apply Algorithm 5.4. We have to test 3 maps ψ from $\{1, 2, 3\}$ to $(\mathbb{Z}/7\mathbb{Z})^*$,

$$\begin{matrix} 1 \mapsto 1 \\ 2 \mapsto 1 \\ 3 \mapsto 1 \end{matrix}\ , \quad \begin{matrix} 1 \mapsto 1 \\ 2 \mapsto 2 \\ 3 \mapsto 4 \end{matrix} \quad \text{and} \quad \begin{matrix} 1 \mapsto 1 \\ 2 \mapsto 4 \\ 3 \mapsto 2 \end{matrix}$$

The last one gives the automorphism
$\alpha \mapsto (-14791767248\alpha^{20} + 95714137173\alpha^{19} + 361204094449\alpha^{18}$
$- 3328779599255\alpha^{17} + 1864224944156\alpha^{16} + 28304237511321\alpha^{15}$
$- 55006453622284\alpha^{14} - 56486002410703\alpha^{13} + 244003596280558\alpha^{12}$
$- 120533924745455\alpha^{11} - 281694320179321\alpha^{10} + 330143856833197\alpha^{9}$
$+ 41475180195786\alpha^{8} - 222891172143821\alpha^{7} + 82722087561988\alpha^{6}$
$+ 36804911589073\alpha^{5} - 32245355397328\alpha^{4} + 6074801569203\alpha^{3}$
$+ 699019170541\alpha^{2} - 345689725163\alpha + 29710312476)/76627979$
corresponding to the following permutation of the ℓ-adics roots
$(2, 301, 96, 66, 370, 342, 440)(35, 225, 109, 180, 236, 305, 51)$
$(45, 223, 373, 79, 448, 397, 174)$.

We then compute the fixed field using Proposition 5.6 and we get $R = x^3 - x^2 - 9x + 1$. We then recursively apply the algorithm to R: we find that R is irreducible modulo 11 and then by lifting the Frobenius, we find the automorphism $-1/2x^2 + 7/2$ which generates the whole Galois group of the fixed field. We must now lift it to an automorphism of G. Using Algorithm 5.8, we get $u = 2$ and using Algorithm 5.11, we have to test the following seven permutations

$(2, 35, 45)(301, 109, 448)$ $(96, 236, 223)(66, 51, 397)$
$(370, 225, 373)(342, 180, 174)(440, 305, 79)$,
$(2, 35, 223)(301, 109, 397)$ $(96, 236, 373)(66, 51, 174)$
$(370, 225, 79)(342, 180, 45)(440, 305, 448)$,
$(2, 35, 373)(301, 109, 174)$ $(96, 236, 79)(66, 51, 45)$
$(370, 225, 448)(342, 180, 223)(440, 305, 397)$,
$(2, 35, 79)(301, 109, 45)$ $(96, 236, 448)(66, 51, 223)$
$(370, 225, 397)(342, 180, 373)(440, 305, 174)$,
$(2, 35, 448)(301, 109, 223)$ $(96, 236, 397)(66, 51, 373)$
$(370, 225, 174)(342, 180, 79)(440, 305, 45)$,
$(2, 35, 397)(301, 109, 373)$ $(96, 236, 174)(66, 51, 79)$
$(370, 225, 45)(342, 180, 448)(440, 305, 223)$,
$(2, 35, 174)(301, 109, 79)$ $(96, 236, 45)(66, 51, 448)$
$(370, 225, 223)(342, 180, 397)(440, 305, 373)$

The second permutation corresponds to the automorphism
$\alpha \mapsto (-194129435\alpha^{20} + 1263256585\alpha^{19} + 4695474072\alpha^{18} - 43866808007\alpha^{17} + 26044995959\alpha^{16} + 370791458219\alpha^{15} - 735728499700\alpha^{14} - 716698532307\alpha^{13} + 3233978982336\alpha^{12} - 1696567684697\alpha^{11} - 3658157589221\alpha^{10} + 4481659267398\alpha^{9} + 407309044781\alpha^{8} - 2976215526118\alpha^{7} + 1190032598839\alpha^{6} + 464793759571\alpha^{5} - 447754215324\alpha^{4} + 90593004738\alpha^{3} + 9006434498\alpha^{2} - 5016784391\alpha + 438769426)/423359$.

We now have a generating set for the Galois group, and we can compute all the elements.

8. Timings

The above algorithm has been implemented using the PARI library (see [PARI, function `nfgaloisconj`]), and the computations have been made on a 800Mhz Pentium III and 1 Gb of RAM. In the first benchmark, we have compared our implementation with Klüners' Algorithm in [Klüners1] as implemented in [KANT V4, version 2.2.7]. The given timings are the averages of the individual timings over the nb polynomials tested.

The polynomials used were chosen to represent all the abstract groups of order between 10 and 27, with the exception of one group of order 16 and one group of order 24. We were not able to compute automorphisms of larger polynomials with [KANT V4, function `OrderAutomorphism`] in less than one hour. The list of polynomials used is available from [Benchmark].

For the second bench, polynomials from our database were used. The abstract Galois groups of these polynomials are distinct. The last three polynomials are the minimal polynomials of $\sqrt[20]{2} + \zeta_{20}$, $\sqrt[24]{5} + \zeta_{24}$ and $\sqrt[22]{2}(\zeta_{22} - 2)$ respectively.

degree	nb	PARI	KASH
10	2	55 ms	115 ms
11	1	50 ms	10 ms
12	5	88 ms	266 ms
13	1	80 ms	60 ms
14	2	100 ms	385 ms
15	1	90 ms	20 ms
16	13	193 ms	662 ms
17	1	170 ms	220 ms
18	5	188 ms	1082 ms
19	1	270 ms	810 ms
20	5	332 ms	2906 ms
21	2	290 ms	2265 ms
22	2	340 ms	25 ms
23	1	280 ms	100 ms
24	14	616 ms	63859 ms
25	2	580 ms	4440 ms
26	2	560 ms	637560 ms
27	5	1584 ms	11538 ms

degree	nb	av. time
28	4	1.66 s
32	20	1.57 s
36	8	1.67 s
40	5	4.78 s
44	4	15.82 s
48	12	17.44 s
52	2	5.48 s
56	3	2,88 s
60	3	3,93 s
64	14	29.51 s
68	1	4,78 s
72	5	4,35 s
76	1	87,0 s
80	4	77.79 s
84	3	110.21 s
88	3	9.44 s
92	1	10.25 s
96	5	10.8 s
160	1	49 min
192	1	108 min
220	1	85 min

9. Acknowledgment

The author wishes to thank an anonymous referee for the application of the $d-1$–*test* and the reference to [ASZ], which have led to an important performance gain in the implementation of the algorithm.

References

[ASZ] John Abbot, Victor Shoup and Paul Zimmerman *Factoring in* $\mathbb{Z}[X]$*: The Searching Phase*, ISSAC 2000, 2000, to appear.

[AcKl] Vincenzo Acciaro and Jürgen Klüners, *Computing Automorphisms of Abelian Number Fields*, Math. Comp., **68**, 1999, 1179–1186.

[Benchmark] *List of polynomials of the benchmark*, `http://www.math.u-bordeaux.fr/~allomber/nfgaloisconj_benchmark.html`

[Cohen] Henri Cohen, *A Course in Computational Algebraic Number Theory*, Graduate Texts in Mathematics, **138**, Springer, 1993.

[Hall] Marshall Hall, *The theory of groups*, Macmillan, New York, 1959.

[Klüners1] Jürgen Klüners *Über die Berechnung von Automorphismen und Teilkörpern algebraischer Zahlkörper*, Thesis, Technischen Universität Berlin, 1997.

[Klüners2] Jürgen Klüners *On computing subfields — A detailed description of the algorithm*, J. Théorie des Nombres Bordeaux, **10**, 1998, 243–271.

[KlMa] Jürgen Klüners and Gunter Malle *Explicit Galois realization of transitive groups of degree up to 15*, J. Symb. Comput. **30**, 2000, 675-716.

[KANT V4] M. Daberkow, C. Fieker, J. Klüners, M. Pohst, K. Roegner, M. Schörnig and K. Wildanger, *KANT V4*, J. Symb. Comput., **24**, 1997, 267–283.

[Magma] W. Bosma, J. Cannon and C. Playoust, *The Magma algebra system I: The user language*, J. Symb. Comput., **24**, 1997, 235–265.

[PARI] PARI, C. Batut, K. Belabas, D. Bernardi, H. Cohen and M. Olivier, *User's Guide to PARI-GP*, version 2.2.1.

[Roblot] Xavier Roblot, *Algorithmes de factorisation dans les extensions relatives et applications de la conjecture de Stark à la construction de corps de classes de rayon*, Thesis, Université Bordeaux I, 1997.

Bill Allombert, Université Bordeaux I, Laboratoire A2X, 351 cours de la Libération, 33 405 Talence, France.

E-mail address: `allomber@math.u-bordeaux.fr`