Efficient Arithmetic in Successive Algebraic Extension Fields Using Symmetries

Sébastien Orange, Guénaël Renault, Kazuhiro Yokoyama

Efficient Arithmetic in Successive Algebraic Extension Fields Using Symmetries

Sébastien Orange, Guénaël Renault and Kazuhiro Yokoyama

Abstract. In this article, we present new results for efficient arithmetic operations in a number field K represented by successive extensions. These results are based on multi-modular and evaluation-interpolation techniques. We show how to use intrinsic symmetries in order to increase the efficiency of these techniques. Applications to splitting fields of univariate polynomials are presented.

Mathematics Subject Classification (2000). Primary 12Y05; Secondary 12F10, 12-04.

Keywords. Algorithms, Arithmetic, Algebraic extension field, Splitting field.

1. Introduction

Efficient computations over number fields are highly required for solving problems related to algebraic numbers effectively, in particular for algorithmic number theory. Since many of those computations are based on "low level" arithmetic operations such as addition, multiplication and inverse computation, making those operations very efficient should contribute the total efficiency of computations over algebraic extension fields.

A number field (an algebraic extension field of the rational number field) can be represented in different manners, by simple extension or by successive extensions. From a computational point of view, it becomes known that representation by successive extensions is much better than that by a simple extension. Because it may keep the original mathematical structure of algebraic extension fields and also the sparsity of the expressions, and certain coefficients growth arising in converting representation by simple extension can be avoided. Thus, we focus on efficient arithmetic operations in number fields which are represented by successive extensions and we provide two basic tools "modular evaluation-interpolation" and "use of symmetries" for making such operations very efficient.

A number field K is represented by successive extensions as follows:

$$K = K_n \supset K_{n-1} \supset K_{n-2} - \cdots \supset K_0 = \mathbb{Q},$$

where each field K_i is defined by adjoining an element α_i to K_{i-1} and thus it can be represented as a residue class ring of $K_{i-1}[x_i]$ factored by the minimal polynomial $g_i(x)$ of α_i over K_{i-1}. In this representation, arithmetic operations in K can be realized by computing those modulo a *triangular set* $\{g_1(x_1), \ldots, g_n(x_1, \ldots, x_n)\}$ of $\mathbb{Q}[x_1, \ldots, x_n]$, where variables α_i's are replaced with the variables x_i's in polynomials $g_1, \ldots, g_n$. Such a triangular set forms a Gröbner basis of the ideal M consisting of all algebraic relations among $\alpha_1, \ldots, \alpha_n$.

There are *cons* and *pros* of using such "triangular" representation. In our point of view, the most effective pro is that *intrinsic symmetries* of K (e.g. $\mathbb{Q}$-automorphisms) are easier to compute in this representation. On the other hand, representation by simple extension looks attractive for use, since arithmetic seems more efficient in this case. But this reason tends to be mitigated now. Actually, many researches have been done recently in order to increase the efficiency of arithmetic operations modulo triangular sets (e.g. see [5, 15]) and similar complexity as in representation by simple extension can now be obtained. These researches are based on fast FFT computation modulo triangular sets, where evaluation-interpolation techniques with deformation for enabling interpolation are mainly used (see [5]). Here, as an effective tool, we introduce modular techniques, where we project our problem modulo primes and deformation techniques become unnecessary. Thus, only modular computation of "evaluation-interpolation" is sufficient, which should bring very efficient computations.

Contribution of the article. Here, we introduce the use of multi-modular techniques for arithmetic in K represented by successive extensions in different manner to exiting ones (e.g. [12, 13]). The main reason of introducing those techniques is that it is the easiest way to parallelize algorithms which can be used nowadays with increasing number of cores in processors. Moreover, some computations can provide p-adic approximations of the variety corresponding to the ideal M defined by a triangular Gröbner basis for different primes. This is the case, in particular, when K is the splitting of a univariate polynomial and when one wants to compute its Galois group (see [11, 20, 22]).

We show how the intrinsic symmetries can be used in order to increase the efficiency of arithmetics in K. This gain mainly appears during the evaluation-interpolation process modulo a prime. This gain is measured theoretically as well as in practice. In our knowledge, this is the first time that such use of intrinsic symmetries is shown for arithmetic operations modulo a triangular set. Such intrinsic symmetries have already been used in the case of splitting field computation but not for its arithmetic (e.g see [14, 16, 19]). Note that our approach is different from the ones where the Galois group and the splitting field of a polynomial are computed at the same time. In this case, the successive knowledge of the Galois action guides the computation (e.g. see [9, 10]).

Structure of the article. In Section 2, some preliminaries on modular techniques are presented. In Section 3, we describe how arithmetic operations in K can be efficiently realized by multi-modular techniques and Section 4 provides results about the use of intrinsic symmetries for computation of these arithmetic operations. Section 5 is devoted to the application of our results to the case where K is the splitting field of a univariate polynomial with integer coefficients and Section 6 gives some practical results of our implementations.

2. Mathematical Preliminaries on Modular Techniques

In this section, we provide some fundamental results on modular techniques. These results are used, in Sections 4 and 3, to develop efficient arithmetic operations over algebraic extension fields. For Gröbner bases, see [7] and for algebraic number theory, see [6].

General Setting: Let $\mathbb{Q}$ be the field of rational numbers, $K = \mathbb{Q}(\alpha_1, \ldots, \alpha_n)$ an algebraic extension field obtained by adjoining algebraic numbers $\alpha_1, \ldots, \alpha_n$ and D the extension degree $|K : \mathbb{Q}|$.

For dealing with K, we consider a polynomial ring $\mathbb{Q}[x_1, \ldots, x_n]$ and the following ring homomorphism, where each α_i is assigned to x_i:

$$\begin{array}{rccc} \varphi: & \mathbb{Q}[x_1, \ldots, x_n] & \longrightarrow & K \\ & f(x_1, \ldots, x_n) & \longmapsto & f(\alpha_1, \ldots, \alpha_n) \end{array}$$

Let M be the kernel of φ. Then M is the maximal ideal corresponding to all algebraic relation among algebraic numbers $\alpha_1, \ldots, \alpha_n$ and K is identified to $\mathbb{Q}[x_1, \ldots, x_n]/M$. For each $x_i \in \mathbb{Q}[x_1, \ldots, x_n]/M$, we denote its minimal polynomial by $m_i(x_i)$. Since $m_i(x_i)$ generates the ideal $M \cap \mathbb{Q}[x_i]$, $m_i(x)$ is the minimal polynomial of α_i over $\mathbb{Q}$.

Thus, any element a of K can be expressed as a polynomial $A(x_1, \ldots, x_n) \in \mathbb{Q}[x_1, \ldots, x_n]$ such that $A(\alpha_1, \ldots, \alpha_n) = a$. In order to assign a unique polynomial A to each element of K, we use the reduced Gröbner basis G of M with respect to some monomials order $\succ$. Then A is expressed by its normal form $NF_G(A)$ with respect to G which is a uniquely determined $\mathbb{Q}$-linear sum of terms t in T_{red}, where T_{red} denotes all reduced terms with respect to G.

To simplify our setting, we assume that all α_i are algebraic integers and we set $\alpha = (\alpha_1, \ldots, \alpha_n)$. Moreover, we assume that G is the reduced Gröbner basis of triangular form with respect to a lexicographic order $x_1 \prec \cdots \prec x_n$ which corresponds to the following tower of successive extensions:

$$\mathbb{Q}(\alpha_1) \subseteq \mathbb{Q}(\alpha_1, \alpha_2) \subseteq \cdots \subseteq \mathbb{Q}(\alpha_1, \ldots, \alpha_n)\,.$$

More precisely, we have $G = \{g_1(x_1), g_2(x_1, x_2), \ldots, g_n(x_1, \ldots, x_n)\}$ where each $g_i(x_1, \ldots, x_i)$ is a monic polynomial in x_i and $g_i(\alpha_1, \ldots, \alpha_{i-1}, x_i)$ is irreducible over $\mathbb{Q}(\alpha_1, \ldots, \alpha_{i-1})$. The degree d_i of $g_i(x_1, \ldots, x_i)$ in x_i coincides with the extension

degree $|\mathbb{Q}(\alpha_1, \ldots, \alpha_i) : \mathbb{Q}(\alpha_1, \ldots, \alpha_{i-1})|$. Thus, we have

$$T_{red} = \{x_1^{e_1} \cdots x_n^{e_n} \mid 0 \leq e_i < d_i\}.$$

2.1. Lucky prime and projection

One of our main tool for efficient arithmetic operations over extension fields is the use of the modular projection of M. Here, we present fundamental results.
Let p be a prime number. From now on, we use the following notations:

- $\mathbb{Z}_p^0 = \{a/b \mid a, b \in \mathbb{Z} \text{ and } p \text{ does not divide } b\}$;
- $proj_p$ is the projection from $\mathbb{Z}_p^0$ to $\mathbb{F}_p = \mathbb{Z}/p\mathbb{Z}$ and also its natural extension to the projection from a polynomial ring over $\mathbb{Z}_p^0$ to that over $\mathbb{F}_p$;
- $\overline{M} = proj_p(M \cap \mathbb{Z}_p^0[x_1, \ldots, x_n])$.

It can be easily shown that $\overline{M}$ is an ideal of $\mathbb{F}_p[x_1, \ldots, x_n]$.

Definition 1. *A prime p is said to be* lucky *of M if the following conditions hold:*
(1) For each x_i, its minimal polynomial $m_i(x_i)$ belongs to $\mathbb{Z}_p^0[x_i]$ and $proj_p(m_i(x_i))$ is square-free.
(2) The reduced Gröbner basis G of M is included in the ring $\mathbb{Z}_p^0[x_1, \ldots, x_n]$ and the image $proj_p(G)$ is the reduced Gröbner basis of $\overline{M}$.

For a lucky prime p of M, we have

$$D = \dim_{\mathbb{Q}} \mathbb{Q}[x_1, \ldots, x_n]/M = \dim_{\mathbb{F}_p} \mathbb{F}_p[x_1, \ldots, x_n]/\overline{M} \ .$$

Remark 2. *Suppose that K is the splitting field of some monic integral polynomial $f(x)$ in $\mathbb{Z}[x]$, where $\alpha_1, \ldots, \alpha_n$ are all zeros of $f(x)$. In this case, it is shown in* [22] *that a prime p is lucky if and only if $proj_p(f(x))$ is square-free.*

Definition 3. *For a lucky prime p of M, a* modular zero *with respect to p is a zero of the projected ideal $\overline{M}$ in $\overline{\mathbb{F}}_p^n$, where $\overline{\mathbb{F}}_p$ denotes the algebraic closure of $\mathbb{F}_p$.*

For efficient computations, we suppose that the followings:

Assumption 1: The modular zeros of M for each lucky prime p can be efficiently computed.

Assumption 2: Approximated values of zeros of M as complex numbers can be efficiently computed with error analysis. This hypothesis is mainly used for the computation of the theoretical bounds over the coefficients of computed results.

Now, let p be a lucky prime of M. By $V(M)$ and $V_p(M)$ we denote the set of all zeros of M and that of modular zeros of M with respect to p, respectively. For simplicity, we write V_p for $V_p(M)$ when the situation is clear. Then, we have $\#V(M) = D$ and $V(M) = \{\sigma(\alpha) \mid \sigma \in \mathcal{E}\}$, where $\mathcal{E}$ is the set of embeddings of K into the algebraic closure $\bar{\mathbb{Q}}$ of K or $\mathbb{C}$. Moreover, by Lemma 2.3 shown later, $\overline{M}$ is proven to be radical and we have $\#V(M) = \#V_p(M) = D$, The set $V_p(M)$ can be obtained as a set of zeros over a finite extension $\mathbb{F}_q$ of $\mathbb{F}_p$. It is very preferable for efficient computation that all modular zeros are *rational*, that is, those are in $\mathbb{F}_p^n$.

Remark 4. *In practice, our main target, where our modular method can work very efficiently, is the splitting field of an integral polynomial $f(x)$. For this case, Assumption 1 and 2 are satisfied. Because, we can find rather easily a lucky prime p such that the modular zeros are all rational, that is, $V_p \subset \mathbb{F}_p^n$. Those modular zeros are computed for determination of the Galois group of f and also for computing the Gröbner basis of M. Also, as each zero α consists of roots of $f(x)$, approximate values of zeros can be computed by simply estimating the size of roots of $f(x)$.*

First we show the *commutativity* of two operations, the projection and the normal form computation. If two polynomials A, B belong to $\mathbb{Z}_p^0[x_1, \ldots, x_n]$, the result C of the monomial division A by B still belongs to $\mathbb{Z}_p^0[x_1, \ldots, x_n]$ and $proj_p(C)$ coincides with the corresponding result of the monomial division $proj_p(A)$ by $proj_p(B)$. Thus, for normal forms of polynomials with respect to G, we have the following:

Lemma 2.1. *For any $A(x_1, \ldots, x_n)$ in $\mathbb{Z}_p^0[x_1, \ldots, x_n]$, its normal form $NF_G(A)$ with respect to the Gröbner basis G also belongs to $\mathbb{Z}_p^0[x_1, \ldots, x_n]$ and $proj(NF_G(A))$ coincides with the normal form $NF_{proj_p(G)}(proj_p(A))$ of $proj_p(A)$ with respect to $proj_p(G)$.*

Next, we recall some properties of characteristic polynomials and minimal polynomials of elements with respect to M or $\overline{M}$. For a polynomial $A(x_1, \ldots, x_n) \in \mathbb{Z}_p^0[x_1, \ldots, x_n]$, a linear map φ_A is defined by multiplication of A on $\mathbb{Q}[x_1, \ldots, x_n]/M$:

$$\begin{array}{rccc} \varphi_A : & \mathbb{Q}[x_1, \ldots, x_n]/M & \longrightarrow & \mathbb{Q}[x_1, \ldots, x_n]/M \\ & B & \mapsto & AB \end{array}$$

With respect to the linear basis T_{red} of $\mathbb{Q}[x_1, \ldots, x_n]/M$ the matrix representation M_A of the map φ_A can be obtained by normal forms of At for $t \in T_{red}$. By Lemma 2.1, M_A is a matrix over $\mathbb{Z}_p^0$. Thus, its characteristic polynomial $\Phi_A(y)$ belongs $\mathbb{Z}_p^0[y]$, where y is new variable. We note that $\Phi_A(y) = \det(yE - M_A) = \prod_{\beta \in V(M)}(y - A(\beta))$, where E denotes the identity matrix, and its constant term coincides with $\det(-M_A) = (-1)^D \prod_{\beta \in V(M)} A(\beta)$. Moreover, $\Phi_A(A)$ belongs to M.

Similarly, a linear map $\varphi_{proj_p(A)}$ of $\mathbb{F}_p[x_1, \ldots, x_n]$ is also defined by multiplication of $proj_p(A)$ and its matrix representation $M_{proj_p(A)}$ can be determined with respect to T_{red}, since T_{red} is also a linear basis of the quotient algebra $\mathbb{F}_p[x_1, \ldots, x_n]/\overline{M}$. Then, we have the following lemma.

Lemma 2.2. *We have $proj_p(M_A) = M_{proj_p(A)}$, where we extend the projection $proj_p$ to matrices over $\mathbb{Z}_p^0$, and $proj_p(\Phi_A) = \Phi_{proj_p(A)}$ where $\Phi_{proj_p(A)}$ is the characteristic polynomial of $M_{proj_p(A)}$.*

The minimal polynomial m_A of A with respect to M is a factor of Φ_A over $\mathbb{Q}$ and it is defined as the smallest factor such that $m_A(A)$ belongs to M. We note that the polynomial $m_A(y)$ contains every irreducible factor of $\Phi_A(y)$. Thus, if A belongs to $\mathbb{Z}_p^0[x_1, \ldots, x_n]$, it can be shown by Gauss's lemma that

$m_A(y)$ belongs to $\mathbb{Z}_p^0[y]$. Moreover, the minimal polynomial $m_{proj_p(A)}(y)$ is a factor of $\Phi_{proj_p(A)}(y) = proj_p(\Phi_A(y))$ and is also a factor of $proj_p(m_A(y))$ since $proj_p(m_A(A)) = proj_p(m_A)(proj_p(A))$ belongs to $\overline{M}$.

Since $proj_p(m_i)$ is square-free, we have the next lemma.

Lemma 2.3. *For each x_i, $proj_p(m_i)$ coincides with the minimal polynomial of x_i with respect to $\overline{M}$. Thus, $\overline{M}$ is radical by Seidenberg's theorem.*

We show some properties of polynomial representation of algebraic integers of K. Let a be an algebraic integer of K whose polynomial representation is $A(x_1, \ldots, x_n) \in \mathbb{Q}[x_1, \ldots, x_n]$ such that $A = NF_G(A)$, $d(K)$ the discriminant of K and $\mathcal{D}$ the discriminant of "polynomial basis" $\{t(\alpha) \mid t \in T_{red}\}$ derived from $\{\alpha_1, \ldots, \alpha_n\}$. (Then $d(K)$ is a factor of $\mathcal{D}$.) As $A = \sum_{t \in T_{red}} c_t t$, where c_t in $\mathbb{Q}$, we have the following by using the fact that $\mathcal{D} = \det([Tr_{K/\mathbb{Q}}(t(\alpha) \times t'(\alpha))]_{t,t' \in T_{red}})$ and $Tr_{K/\mathbb{Q}}(a \times t(\alpha)) = \sum_{t' \in T_{red}} c_t Tr_{K/\mathbb{Q}}(t(\alpha) \times t'(\alpha))$ is integer for each t in T_{red}.

Lemma 2.4. *The denominator of A is a factor of $\mathcal{D}$, where the denominator of A means the LCM of the denominators of all coefficients c_t as rational numbers, that is, the smallest positive integer δ such that δA belongs to $\mathbb{Z}[T_{red}]$.*

On the other hand, the *discriminant composition formula* in [18] implies that $\mathcal{D}$ is a product of factors taken in $\{\tau(\alpha_i) - \sigma(\alpha_i) \mid 1 \leq i \leq n,\ \sigma, \tau \in \mathcal{E}\}$. For each lucky prime p, since $proj_p(m_i(x_i))$ is square-free in $\mathbb{F}_p[x_1, \ldots, x_n]$, its discriminant does not vanish modulo p and we also have the following lemma.

Lemma 2.5. *Each lucky prime p does not divide $\mathcal{D}$ and hence it does not divide the denominator of A.*

2.2. Modular Evaluation-Interpolation technique

In this subsection, we recall the evaluation-interpolation method used for computing field arithmetic operations modulo $\overline{M}$.

Let p be a lucky prime and suppose that $\overline{M}$ has all its roots in some extension field $\mathbb{F}_q$ of $\mathbb{F}_p$. Of course, for efficient computation, it is desirable that $q = p$. Thanks to the triangular property of $\overline{M}$, its corresponding variety is *equiprojectable* (see [3]). More precisely, let's set $proj_p(G) = \{\overline{g}_1(x_1), \overline{g}_2(x_1, x_2), \ldots, \overline{g}_n(x_1, \ldots, x_n)\}$ which is the triangular Gröbner basis of $\overline{M} \subset \mathbb{F}_p[x_1, \ldots, x_n]$ and consider the corresponding variety $V_p \subset \mathbb{A}(\mathbb{F}_q, n)$. Then the variety V_p has an intrinsic recursive definition as follows: For an integer $0 < i < n$, letting V_i be the variety of $\langle \overline{g}_1, \overline{g}_2, \ldots, \overline{g}_i \rangle$, the elements $(\beta_1, \beta_2, \ldots, \beta_i)$ of V_i are defined by $(\beta_1, \ldots, \beta_{i-1}) \in V_{i-1}$ and a root β_i of $\overline{g}_i(x_i, \beta_{i-1}, \ldots, \beta_1)$. In particular, the cardinal of V_p is given by the product of the degrees d_i of $\overline{g}_i$ in x_i.

In this case, the classical evaluation-interpolation technique can be performed in $\mathbb{F}_q[x_1, x_2, \ldots, x_n]/\mathbb{F}_q \otimes_{\mathbb{F}_p} \overline{M}$, where $\mathbb{F}_q \otimes_{\mathbb{F}_p} \overline{M}$ is the ideal generated by $proj_p(G)$ in $\mathbb{F}_q[x_1, \ldots, x_n]$, and, moreover, univariate interpolation can be generalized to multivariate polynomials in $\mathbb{F}_q[x_1, x_2, \ldots, x_n]/\mathbb{F}_q \otimes_{\mathbb{F}_p} \overline{M}$. More precisely, we can define two functions:

Definition 5. *Let p be a lucky prime. The function* `Eval` *takes a polynomial P in $\mathbb{F}_p[x_1, \ldots, x_n]$ and a set U of evaluation points and it returns a finite sequence $(P(\beta))_{\beta \in U}$. From the output of* `Eval`(P, V_p) *and the corresponding variety V_p, the function* `Interpol` *will reconstruct the polynomial P.*

The reconstruction of the polynomial P can be done by multivariate interpolation in the following manner.

Let $\beta \in V_p$ and E_β the polynomial in $\mathbb{F}_q[x_1, \ldots, x_n]/\mathbb{F}_q \otimes_{\mathbb{F}_p} \overline{M}$ verifying $E_\beta(\beta') = 0$ for $\beta' \in V_p \setminus \{\beta\}$ and $E_\beta(\beta) \neq 0$. The polynomial E_β can be computed by considering the equiprojectable $\mathbb{F}_q$-subvarieties of V_p of smaller dimension. From the polynomials E_β, we can interpolate the polynomial P by the following generalized Lagrange formula:

$$P = \sum_{\beta \in V_p} P(\beta) \frac{E_\beta}{E_\beta(\beta)}$$

Such a formula was introduced in [8] for reconstruction of the polynomials g_i in general and results in [20] restate it in a Galois theoretical point of view. In [5], a recursive algorithm is given for such an interpolation. Moreover, they proved that evaluation and interpolation can be done with $\mathcal{O}(D \log^3 D)$ operations in the base field with $D = d_1 d_2 \cdots d_n$.

Remark 6. *As to the splitting field case, if all modular zeros are rational for a lucky p, that is, $V_p \subset \mathbb{F}_p^n$, we can compute p-adic approximations of those zeros by Hensel lifting. (In this case, we consider all zeros of M in $\mathbb{Z}_p^n$ and an approximation of such zero α means $\alpha \pmod{p^k}$ for some positive integer k.) Then all evaluation and interpolation operations can be done over modulo p^k.*

General framework of multi-modular and modular evaluation-interpolation: Here, we describe our framework in a general form that will be used in the sequel for arithmetic in the field K. For any operation we want to do in the field K, we proceed as follows:

(1) We first project the operands modulo sufficiently many lucky primes.
(2) For each primes, we compute the operation with the projected operands by modular evaluation-interpolation process.
(3) Then, by CRT and rational reconstruction, we reconstructed the result in the field K.

We choose each lucky prime p so that all modular zeros in V_p are rational or smaller extension of $\mathbb{F}_p$ is required to obtain those. By Chevotarev's density Theorem, the ratio of such primes can be estimated by the Galois group of the Galois closure of K. (See [22] for splitting field case.) The correctness of the computed result is examined by the zero-test shown in the Section 3.1.

3. Arithmetic in extension fields with modular technique

In this section we present arithmetic in the number field K in general setting by using our *multi-modular and modular evaluation-interpolation framework*.

3.1. Zero-test over K

The most important technique for arithmetic in K is *zero-test*. Usually, when the Gröbner basis G of M is given, the fact that the reduction modulo M is unique gives such a test: the normal form returned is then 0. Actually, to perform the identification of two elements A and B in K, one have to test the equality $A = B$. When two polynomials A and B are not reduced modulo M, to perform this identification, it is better to check the equality $A - B = 0$ instead of computing the normal forms of A and B.

Normal form computations can be done by successive Euclidean pseudo-divisions by the polynomials of G which can be particularly costly. In order to construct an efficient zero-test, we apply the modular method described as follows.

Let p be a lucky prime and consider a given polynomial expression $A(\alpha)$ in K. By removing denominators, we can assume w.l.o.g. that A in $\mathbb{Z}[x_1, \ldots, x_n]$. By Lemma 2.1, $NF_G(A)$ belongs to $\mathbb{Z}_p^0[x_1, \ldots, x_n]$ and we have the following:

Lemma 3.1. *The numerator of $NF_G(A)$ is divisible by p if and only if $NF_{proj_p(G)}(proj_p(A)) = 0$. Moreover, the last condition is equivalent to*

$$\forall \beta \in V_p(M),\ proj_p(A)(\beta) = 0\,.$$

As $\overline{M}$ is radical, the NullStellenSatz shows the last statement of the lemma.

Thus, by normal form computation modulo $proj_p(G)$ or evaluation by modular zeros we can check if the numerator of the normal form is divisible by p.

Gathering a number of lucky primes $p_1, \ldots, p_s$, we may examine if the numerator of the normal form is divisible by the product $\prod_{i=1}^{s} p_i$. Then it arises a problem: *How many such primes are necessary to prove $A(\alpha) = 0$?* This can be solved very effectively by using the notion of norm without bounding the coefficients of $NF_G(A)$ via tracing its computation.

Bound via Norm Approximation: The norm $N_{K/\mathbb{Q}}(A(\alpha))$ can be defined as

$$N_{K/\mathbb{Q}}(A(\alpha)) = \prod_{\sigma}(A(\sigma(\alpha_1), \ldots, \sigma(\alpha_n))$$

where σ ranges all embeddings of K into its algebraic closure $\overline{K}$ and thus,

$$N_{K/\mathbb{Q}}(A(\alpha)) = \prod_{\alpha' \in V(M)} A(\alpha')$$

and $(-1)^D N_{K/\mathbb{Q}}(A(\alpha))$ coincides with the constant term of the characteristic polynomial Φ_A of $A(x_1, \ldots, x_n)$ with respect to M. Since $A \in \mathbb{Z}[x_1, \ldots, x_n]$, $A(\alpha)$ is

an algebraic integer and $N_{K/\mathbb{Q}}(A(\alpha))$ is an integer. Thus, since $A(\alpha) = 0$ if and only if $N_{K/Q}(A(\alpha)) = 0$, we may use a bound on the integer $N_{K/\mathbb{Q}}(A(\alpha))$.

Lemma 3.2.

$$proj_p(N_{K/\mathbb{Q}}(A(\alpha))) = \prod_{\beta \in V_p(M)} proj_p(A)(\beta).$$

Proof. By Lemma 2.2, the characteristic polynomial $\Phi_{proj_p(A)}$ of $proj_p(A)$ with respect to $\overline{M}$ coincides with the projection image $proj_p(\Phi_A)$. Since the constant term of $\Phi_{proj_p(A)}$ is given by the product $(-1)^D \prod_{\beta \in V_p(M)} proj_p(A)(\beta)$, comparing the constant terms, the result follows. □

Proposition 3.3. *If $NF_{proj_p(G)}(proj_p(A)) = 0$, that is, $proj_p(A)(\beta) = 0$ for all zero β in $V_p(M)$, then*

$$N_{K/\mathbb{Q}}(A(\alpha)) \equiv 0 \pmod{p^D}.$$

Proof. If $proj_p(NF_G(A)) = NF_{proj_p(G)}(proj_p(A)) = 0$ in $\mathbb{F}_p[x_1, \ldots, x_n]$, then $NF_G(A)$ can be written as $p \times \frac{A'}{d'}$ where $A' \in \mathbb{Z}[x_1, \ldots, x_n]$ and d' is the denominator of $NF_G(A)$. For the algebraic integers $d'NF_G(A)(\alpha)$ and $A'(\alpha)$, we have

$$\begin{aligned} p^D N_{K/\mathbb{Q}}(A'(\alpha)) &= N_{K/\mathbb{Q}}(pA'(\alpha)) \\ &= N_{K/\mathbb{Q}}(d'NF_G(A)(\alpha)) \\ &= {d'}^D N_{K/\mathbb{Q}}(NF_G(A)(\alpha)) \\ &= {d'}^D N_{K/\mathbb{Q}}(A(\alpha)) \end{aligned}$$

We note that $NF_G(A)(\alpha) = A(\alpha)$ for α in $V(M)$ and the norm of an algebraic integer is an integer. By Lemma 2.1, p is prime to d' and hence $N_{K/\mathbb{Q}}(A(\alpha))$ should be divided by p^D □

Here, we give a simple estimation using numerical approximation of the norm. First we compute some bound $\mathbf{B}_i$ on each α_i and its conjugates and thus we have some bound $\mathbf{B}_0$ on $|A(\alpha)|$ and the absolute values of its conjugates, that is,

$$\mathbf{B}_0 \geq |A(\beta)| \quad \text{for all } \beta \in V(M).$$

Then, we have $\mathbf{B}_0^D > |N_{K/\mathbb{Q}}(A(\alpha))|$.

Proposition 3.4. *Suppose that $p_1, p_2, \ldots, p_s$ are lucky primes and that, for each p_i, $NF_{proj_{p_i}(G)}(proj_{p_i}(A)) = 0$. If we have $\prod_{i=1}^s p_i > \mathbf{B}_0$, then $A(\alpha) = 0$, that is, $NF_G(A) = 0$. Thus, the necessary number of primes for proving $A(\alpha) = 0$ can be bounded by $O(\log(\mathbf{B}_0))$.*

Proof. As $NF_{proj_{p_i}(G)}(proj_{p_i}(A)) = 0$ for each p_i, we have $N_{K/\mathbb{Q}}(A(\alpha)) \equiv 0 \pmod{p_i^D}$ for each p_i, where $D = \#V(M) = |K : \mathbb{Q}|$, by Proposition 3.3. Then we have $N_{K/\mathbb{Q}}(A(\alpha)) \equiv 0 \pmod{\prod_{i=1}^s p_i^D}$.

On the other hand, since $\prod_{i=1}^{s} p_i > \mathbf{B}_0$, we have

$$\prod_{i=1}^{s} p_i^D > \mathbf{B}_0^D > |N_{K/\mathbb{Q}}(A(\alpha))|,$$

and we conclude $A(\alpha) = 0$. □

Here, we note that if $A(\alpha)$ is zero then its conjugate $A(\beta)$ is also zero for all $\beta \in V(M)$. Therefore, if $A(\alpha) = 0$ it is expected that we have sufficiently small bound $\mathbf{B}_0$.

The cost of zero-test depends on $\mathbf{B}_0$ and the cost of normal form computation over $\mathbb{F}_p$ or that of evaluation with modular zeros. As in the cases of the next subsection, evaluation of some function in algebraic numbers can be done very efficiently than its normal form computation. Because for such cases, we need only one evaluation for each modular zero but we need numbers of successive normal form computations.

Also, the value $\mathbf{B}_0$ is expected to be small if $A(\alpha) = 0$. If we want to have smaller bound, we have to compute evaluations D times for its precise estimation with error analysis, and there exists "trade-off". One of the simplest bound can be computed very efficiently as follows:

Simple Bound: Let $\mathbf{C}_i$ be the largest absolute value among those of conjugates of α_i, which can be computed by its minimal polynomial $m_i(x_i)$, and $\tilde{A}$ the polynomial obtained from A by changing all negative coefficients to positive. Then $\tilde{A}(\mathbf{C}_1, \ldots, \mathbf{C}_n)$ gives a bound on $|A(\beta)|$ for all $\beta \in V(M)$.

3.2. Simple arithmetic in K

Among arithmetics in K, multiplication and inverse computation have an fundamental place. Usually, multiplication is executed by polynomial multiplication and normal form computation, and inverse computation is executed by a variant of the extended Euclidean algorithm. In the next subsections, we present *Modular Evaluation-Interpolation technique* for these operations.

3.2.1. Multiplication. Suppose that two elements a, b in K which are expressed as polynomials $A(\alpha), B(\alpha)$ in α, are given to compute the polynomial expression $C(\alpha)$ of $c = a \times b$. For this computation, we can apply *modular evaluation-interpolation technique* as follows:

By removing denominators, we can assume that A, B in $\mathbb{Z}[x_1, \ldots, x_n]$. Let p be a lucky prime and G the reduced Gröbner basis of M. Then $C = NF_G(AB)$ and, by Lemma 2.1, C can belong to $\mathbb{Z}_p^0[x_1, \ldots, x_n]$. Let d be the denominator of C, that is, d is the smallest positive integer such that dC belongs to $\mathbb{Z}[x_1, \ldots, x_n]$ and set $\tilde{C} = dC$. For each zero $\beta = (\beta_1, \ldots, \beta_n)$ in $V_p(M)$, we have

$$proj_p(AB)(\beta) = proj_p(A)(\beta)\ proj_p(B)(\beta)$$

Moreover, using all zeros in $V_p(M)$, we have a polynomial C_p in $\mathbb{F}_p[x_1, \ldots, x_n]$ by *interpolation* such that for any β in $V_p(M)$

$$proj_p(C)(\beta) = proj_p(C_p)(\beta)$$

and hence

$$proj_p(C) = C_p.$$

Thus, gathering such images for sufficiently many lucky primes $p_1, \ldots, p_s$ and CRT computation, we have $d^{-1}\tilde{C} \bmod \prod_{i=1}^{s} p_i$ by which we can recover C by *rational reconstruction*. For the necessary number of primes, we can apply our *zero-test* as follows:

By several lucky primes $p_1, \ldots, p_s$, we can compute a candidate $C'(\alpha)$ for $C(\alpha) = A(\alpha)B(\alpha)$. Then, by simply considering *zero-test* of $AB - C'$, we have a bound $\mathbf{B}$ for it. If $\prod_{i=1}^{s} p_i^D$ exceeds $\mathbf{B}$, we conclude that C' is the product AB. Otherwise, we take other lucky primes $p_{s+1}, \ldots, p_t$ so that $\prod_{i=1}^{t} p_i^D > \mathbf{B}$ and check $proj_{p_i}(C')(\beta) = proj_{p_i}(AB)(\beta)$ for each $i = s+1, \ldots, t$.

Simple Bound: Let d' be the denominator of a candidate C' such that $d' \in \mathbb{Z}$ and $d'C' \in \mathbb{Z}[x_1, \ldots, x_n]$. Also $\mathbf{B}_0$ is some bound on $\{|(d'AB - d'C')(\beta)| \mid \beta \in V(M)\}$. Then we can set $\mathbf{B} = \mathbf{B}_0^D$ and the number of necessary primes is bounded by $O(\log(\mathbf{B}_0))$.

If C' does not coincides with C, we need to gather additional primes and recalculate another candidate till we obtain C. Thus, the total number of primes can be bounded by $O(\log(\mathbf{B}_0))$, where $\mathbf{B}_0$ is some bound on $\{|(dAB - dC)(\beta)| \mid \beta \in V(M)\}$ and d is the denominator of C.

3.2.2. Inverse computation. Suppose that an elements a in K which is expressed as a polynomial $A(\alpha)$ is given to compute the polynomial expression $B(\alpha)$ of of its inverse a^{-1}, where B is reduced with respect to G. For this computation, we can also apply *modular evaluation-interpolation technique* as follows.

By removing denominators, we can assume that A in $\mathbb{Z}[x_1, \ldots, x_n]$. Then a is an algebraic integer. Let d be the denominator of B, that is, d is the smallest positive integer such that dB belongs to $\mathbb{Z}[x_1, \ldots, x_n]$.

Lemma 3.5. *If a lucky prime p does not divide the norm $N_{K/\mathbb{Q}}(a)$, then, for any β in $V_p(M)$, $proj_p(A)(\beta) \neq 0$ and p does not divide d. Thus, there are only finitely many lucky primes q such that there is a zero β in $V_q(M)$ with $A(\beta) = 0$.*

Proof. Assume that a lucky prime p does not divide $N_{K/\mathbb{Q}}(a)$. By Lemma 3.2 $proj_p(N_{K/\mathbb{Q}}(A(\alpha)) \neq 0$ implies that $proj_p(A)(\beta) \neq 0$ for any β in $V_p(M)$.

Next consider $c = \prod_{\beta \in V(M), \beta \neq \alpha} A(\beta)$. Then we have $N_{K/\mathbb{Q}}(A(\alpha)) = A(\alpha) \times c$ and c is an algebraic integer. Thus, $a^{-1} = c/N_{K/\mathbb{Q}}(a)$ and the polynomial expression of c belongs to $\frac{\mathbb{Z}[x_1, \ldots, x_n]}{\mathcal{D}}$ (see Lemma 2.5). Thus, d divides $\mathcal{D}N_{K/\mathbb{Q}}(a)$ and, as p does not divide $\mathcal{D}$, if p does not divide $N_{K/\mathbb{Q}}(a)$ then p does not divide d. □

Let p be a lucky prime such that $proj_p(A)(\beta) \neq 0$ for each $\beta = (\beta_1, \ldots, \beta_n)$ in $V_p(M)$. By Lemma 3.5, the polynomial B belongs to $\mathbb{Z}_p^0[x_1, \ldots, x_n]$. and for each $\beta = (\beta_1, \ldots, \beta_n)$ in $V_p(M)$, we have

$$proj_p(B)(\beta) = proj_p(A)(\beta)^{-1}.$$

Thus, using all zeros in $V_p(M)$, we construct a polynomial B_p in $\mathbb{F}_p[x_1, \ldots, x_n]$ by *interpolation* such that, for any β in $V_p(M)$, $proj_p(B)(\beta) = B_p(\beta)$ and hence

$$proj_p(B) = B_p.$$

Thus, gathering such images for sufficiently many lucky primes $p_1, \ldots, p_s$ and CRT computation, we have $d^{-1}(dB) \bmod \prod_{i=1}^{s} p_i$ by which we can recover B by *rational reconstruction*.

For the number of necessary primes, we can use some bound on zero-test of $AB - 1$. That is, for a candidate B', we apply zero-test of $AB' - 1$ and if the computed bound still exceeds the current product $\prod_{i=1}^{s} p_i$ of used primes, we apply other primes for verification or new construction.

4. Arithmetic in algebraic extension fields using symmetries

In this section, we present how to use *symmetries* to increase the efficiency of the arithmetic.

In all the sequel, we consider the action of the symmetric group S_n over a polynomial ring R with n variables $x_1, \ldots, x_n$ by permuting variables: For $\gamma \in S_n$ and $P(x_1, \ldots, x_n) \in R$, the action of γ on $P(x_1, \ldots, x_n)$ is defined by

$$\gamma \cdot P := P(x_{\gamma(1)}, \ldots, x_{\gamma(n)}) .$$

4.1. Zero-test and stabilizer of $\overline{M}$

A first example of an efficient use of symmetries during computations in K is for zero-test modulo M. The natural symmetries that one can use in our multi-modular technique come from the stabilizer of the projected ideal $\overline{M}$.

Remark 7. *There is a case where the triangular Gröbner basis G of M is not computed but the modular varieties $V_p(M)$ are computed for different lucky primes p. In such a case, computation of the normal form of an element modulo G is replaced with that of its modular evaluation-interpolation. The use of the symmetries in the modular evaluation-interpolation process will be discussed in the next subsection.*

For a lucky prime p, set $\overline{A} = proj_p(A)$ and

$$\overline{G} = proj_p(G) = \{\overline{g}_1(x_1), \overline{g}_2(x_1, x_2), \ldots, \overline{g}_n(x_1, \ldots, x_n)\} .$$

We want to test the equality $NF_{\overline{G}}(\overline{A}) = 0$. Such a test can be done by successive Euclidean pseudo-divisions by polynomials of the Gröbner basis $\overline{G}$.

Nevertheless, this test can be particularly improved by using the setwise stabilizer H of $\overline{M}$ in the symmetric group S_n naturally acting on the set of variables $\{x_1, \ldots, x_n\}$:

$$H = \{\sigma \in S_n \mid \forall g \in \overline{M},\ \sigma.g \in \overline{M}\}.$$

When there is a conjugate pair among $\alpha_1, \ldots, \alpha_n$, this group is not trivial. For example, when K is the splitting field of a polynomial $f(x)$, H is the Galois group of f.

Remark 8. *A* strong generating set *of H, which is a fundamental tool in algorithmic group theory, can be computed efficiently from the triangular Gröbner basis $\overline{G}$ of $\overline{M}$ (see* [1, 2, 16]*) and therefore can be computed once for all before being used during any zero-test.*

Since each σ in H stabilizes $\overline{M}$ setwise, it follows that $\sigma.\overline{A}$ belongs to $\overline{M}$ if and only if $\overline{A}$ belongs to $\overline{M}$. As a straightforward consequence of this observation, we have the following lemma.

Lemma 4.1. *For any σ in H, $NF_{\overline{G}}(\sigma.\overline{A}) = 0$ if and only if $NF_{\overline{G}}(\overline{A}) = 0$*

Therefore, to improve the efficiency of the zero-test by successive Euclidean pseudo-divisions, instead of reducing $\overline{A}$ by polynomials of $\overline{G}$, one can reduce one of its conjugate under the action of H. Thus, for the test, we can choose one of its conjugate, say $\overline{A}_{min}$, so that the leading term $LT(\overline{A}_{min})$ has smaller (possibly the smallest) order in $\{LT(\sigma.\overline{A}) \mid \sigma \in H\}$. Such a choice improves the efficiency of the zero-test, since

- when the greatest variable x_r appearing in $\overline{A}_{min}$ is lower than the one of A, the zero-test may be done with smaller number of Euclidean pseudo-divisions, because only Euclidean pseudo-divisions by g_r, g_{r-1}, $\ldots$, g_1 are needed;
- even if x_r is the greatest variable appearing in A, we can have $LT(\overline{A}_{min}) \prec LT(A)$ which may decrease significantly the cost of the Euclidean pseudo-division by g_r.

Moreover, to avoid unnecessary Euclidean pseudo-divisions in the case where $\overline{A} \notin \overline{M}$, we can also notice that if

$$\overline{A} = \sum_i x_r^i \overline{A_i}(x_1, \ldots, x_{r-1})$$

is a polynomial reduced with respect to $\overline{g_r}$ (i.e. if $deg_{x_r}(\overline{A}) < deg_{x_r}(g_r)$), then $\overline{A}$ belongs to $\overline{M}$ if and only if all the polynomials $\overline{A_i}$ belong to $\overline{M}$. This means that, if one polynomial $\overline{A_i}$ is not reduced to 0 modulo polynomials of $\overline{G}$, $\overline{A}$ can not be an element of $\overline{M}$ and no other computation is needed.

This remark leads to elaborate the following recursive algorithm based on the successive Euclidean pseudo-divisions by polynomials of $\overline{G}$. The correctness of this algorithm is a straightforward consequence of the two last paragraphs.

Algorithm 1: `Zero_Test`$(\overline{A}, \overline{G}, H))$

Input : • *a polynomial* $\overline{A} \in \mathbb{F}_p[x_1, \ldots, x_n]$
• *the Gröbner basis* $\overline{G} = \{\overline{g}_1(x_1), \overline{g}_2(x_1, x_2), \ldots, \overline{g}_n(x_1, \ldots, x_n)\}$ *of* $\overline{M}$
• *the setwise stabilizer* H *of* $\overline{M}$
Output : true if the reduction of $\overline{A}$ *by* $\overline{G}$ *is 0, and false otherwise*

if $TotalDegree(\overline{A}) \neq 0$ /* $TotalDegree(\overline{A})$ returns the total degree of $\overline{A}$ */
$\overline{A} := \overline{A}_{min}$ where $\overline{A}_{min} \in \{\sigma.\overline{A} \mid \sigma \in H\}$ is such as $LT(\overline{A}_{min}) = Min\{LT(\sigma.\overline{A}) \mid \sigma \in H\}$.
$\overline{A} :=$ Reduction of $\overline{A}$ by g_r where r is the highest index of variables appearing in $\overline{A}$
end if

if $TotalDegree(\overline{A}) = 0$ **then**
if $\overline{A} = 0$ **then**
$ans := true$
else
$ans := false$
end if
else
$\mathcal{S} := \{\overline{A_i}\}$ such that $\overline{A} = \sum_i x_r^i \overline{A_i}(x_1, \cdots, x_{r-1})$ where r is the highest index of variables appearing in $\overline{A}$
repeat
Choose $\overline{A_i}$ in $\mathcal{S}$
Exclude $\overline{A_i}$ of $\mathcal{S}$
$ans :=$ `Zero_Test`$(\overline{A_i}, \overline{G}, H)$
until $(ans = false$ or $\mathcal{S} = \emptyset)$
end if
Return ans

In the worst case, when H is trivial and $\overline{A}$ is reduced to 0, the number of Euclidian divisions is the same as for an usual normal form computation.

Remark 9. *Many improvements of this algorithm can be done as follows:*

1. *For finding* $\overline{A}_{min}$ *from* $\overline{A}$*, since the highest index* r *of variables appearing in* $\overline{A}$ *must be not smaller than that of* $\overline{A}_{min}$*, we only need to consider the action of the setwise stabilizer* H_r *of* H *of* $\{1, \ldots, r\}$*, that is,* $\overline{A}_{min}$ *can be found in the* H_r*-orbit of* $\overline{A}$*. This can reduce the number of comparisons needed for the determination of* $\overline{A}_{min}$*.*
2. *All such setwise stabilizers* $H_1, \ldots, H_{n-1}$ *of* H *can be pre-computed only once.*

Example 10. *If* H *acts transitively on* $\{1, \ldots, n\}$ *and if one wants to compute the normal form of* $\overline{A} = g_1(x_n)$ *with respect to* $\overline{G}$*, the classical algorithm may compute* n *Euclidean pseudo-divisions. Such a situation appears when* $g_2, \ldots, g_n$ *are the Cauchy moduls of* g_1 *and, in this case,* H *is the symmetric group of degree* k *(see* [21]*). With Lemma 4.1, only one Euclidean pseudo-division is needed since there exists a permutation* σ *in* H *such that* $\sigma.g_1(x_n) = g_1(x_1)$*.*

4.2. Modular Evaluation-interpolation technique in presence of symmetries

In this subsection, we investigate the use of the symmetries induced by the operand during the modular evaluation-interpolation process. Like in the last subsection, H is the stabilizer of the modular ideal $\overline{M}$ and, from now on, we write simply V_p

for $V_p(M)$. Clearly, the setwise stabilizer in S_n of the projected variety V_p is H too (for the natural action of the symmetric group on the components of elements of the affine space). From now on, we assume the following hypothesis.

Assumption 3: The action of H over V_p is supposed to be faithful.

Even if one can apply ideas for using symmetries presented here without this assumption, it helps for measuring precisely the impact on the theoretical and practical complexity. Consequently, the variety V_p can be decomposed as a disjoint union of H-orbits and we have the following result.

Lemma 4.2. *If $H \subset S_n$ is the stabilizer of $\overline{M}$ then there exists a finite sequence (a_i), where $a_i \in V_p$, such that*

$$V_p = \cup_i H \cdot a_i$$

with $H \cdot a_i \cap H \cdot a_j = \emptyset$ as soon as $i \neq j$. (Each a_i can be called a representative *of an H-orbit of V_p.)*

Thus, in this case, the evaluation process during modular evaluation-interpolation computation can be done in parallel. More precisely, we have, for any polynomial $P \in \mathbb{F}_p[x_1, \ldots, x_n]$,

$$\texttt{Eval}(P, V_p) = \cup_i \texttt{Eval}(P, H \cdot a_i).$$

We note that if $K/\mathbb{Q}$ is Galois extension, H acts transitively on V_p and thus $V_p = H \cdot a$ for any a in V_p.

Next, we focus on the case where the polynomial P to evaluate has symmetries too. In this case, such symmetries can be used to decrease the number of evaluations needed for the determination of $\texttt{Eval}(P, V_p)$. To simplify our argument, we assume that for any modular zero a in V_p every its component differs to each other, that is, the stabilizer of a in H is $\{1\}$ and the cardinality of the H-orbit Ha coincides with the order $|H|$. For the splitting field case, this assumption always holds for any lucky prime.

Remark 11. *By considering the ideal generated by M and $x_i - x_j$, It can be shown that there are only finitely many primes p such that a modular zero in V_p has the same components. In this case, by an adequote linear transformation, we can let every modular zero have different components.*

Proposition 4.3. *Let $P \in \mathbb{F}_p[x_1, \ldots, x_n]$ and H' the stabilizer of P in H. $\texttt{Eval}(P, V_p)$ can be done in $|V_p|/|H'|$ evaluations of P.*

Proof. ¿From the decomposition given in Lemma 4.2 a value of P over V_p can be given by $P(h \cdot a_i)$ with $h \in H$ and representatives a_i. Since P is H'-invariant then one can store $[P(\overline{h} \cdot a_i) : \overline{h} \in H//H']$ only in order to compute $\texttt{Eval}(P, V_p)$. If one wants to obtain a specific value of $\texttt{Eval}(P, V_p)$, says $P(h \cdot a_i)$, one have first to compute the representative $\overline{h}$ of h in the transversal $H//H'$ and then retrieves the corresponding value in $[P(\overline{h} \cdot a_i) : \overline{h} \in H//H']$. □

In the same way, symmetries can be used in order to increase the efficiency of the interpolation process.

Proposition 4.4. *We use the same notation as in Lemma 4.2. Let P be a polynomial in $\mathbb{F}_p[x_1, \ldots, x_n]$ that one wants to reconstruct its* representative *modulo the ideal $\overline{M}$. Let H' be the stabilizer of P in H. The computation of such reconstruction can be realized with an interpolation over the values given by* $\texttt{Eval}(P, \cup_i (H//H') \cdot a_i)$.

Proof. Here, in order to reconstruct the normal form of P with respect to $\overline{G}$, it is sufficient to reconstruct P over its values taken from V_p. Let R be the polynomial reconstructed from $\texttt{Interpol}([P(a) : a \in V_p], V_p)$ and S the one coming from $\texttt{Interpol}([P(a) : a \in \cup_i H//H' \cdot a_i], V_p)$. Clearly, R corresponds to the normal form P with respect to $\overline{G}$. Since P is H'-invariant, so is S and thus R and S coincide on V_p. Hence S is a representative of P modulo $\overline{M}$. □

Remark 12. *It is important to note that the polynomial reconstructed in the last proposition* may not coincide with *the normal form of P with respect to $\overline{G}$. But, combining it with zero-test, efficient arithmetic over K can be attained.*

For arithmetic, one may be interested in many computations between different operands, and when one wants to check the equality, one can use the function `Zero_Test` *of the last subsection. As this computation can be done by modular evaluation-interpolation over the whole variety, and since the polynomial constructed after such an interpolation has its total degree bounded by D, its complexity is given $\mathcal{O}(D \log^3(D))$ (which is the worst one in this case), but it is required only a few times.*

In the modular evaluation-interpolation process for arithmetic in K, two operands A and B are given (in the case of inversion, we set $B = 1$). In order to use symmetries in this process, we first compute the stabilizer H' of A in H and H'' of B in H. Then, the group $H' \cap H''$ stabilizes the result coming from operation between A and B and we can deduce the following result from the two propositions above.

Theorem 4.5. *The modular evaluation-interpolation process for operations between A and B has a complexity bound by $\mathcal{O}(\delta \log^3(\delta))$ with $\delta = |V_p|/|H'' \cap H'|$.*

5. Application to arithmetic in splitting fields

The computation in the splitting field of a polynomial f is the extremal case, where the methods presented in the preceding sections can be applied very effectively. Here we consider the case where $f(x)$ is a monic irreducible integral polynomial of degree n and K is its splitting field, that is, $K = \mathbb{Q}(\alpha_1, \ldots, \alpha_n)$, where $\alpha_1, \ldots, \alpha_n$ are all roots of $f(x)$.

As described in [20], if a prime p such that $proj_p(f)$ is square-free, p is lucky for the maximal ideal M consitsing of all algebraic relation among $\alpha_1, \ldots, \alpha_n$. (See Remark 2.) This ideal M is called *the splitting ideal* of f and has a triangular Gröbner basis $G = \{g_1(x_1), \ldots, g_n(x_1, \ldots, x_n)\}$, where $g_i(\alpha_1, \ldots, \alpha_{i-1}, x_{i-1})$ is the minimal polynomial of α_i over $\mathbb{Q}(\alpha_1, \ldots, \alpha_{i-1})$.

The computation of a symmetric representation G_f of the Galois group of a monic integral polynomial f can be done by using p-adic Stauduhar approach (see [11, 22]) which provides the exact action of the group over p-adic approximations of the roots of f. In other words, this computation can provide a modular variety V_p for a number of lucky primes p for the splitting ideal M of f and the action of G on V_p can be *synchronized* for these lucky primes p (see [20]) and is, by definition, faithful. Thus, Assumption 1 and Assumption 3 hold. Moreover, from the coefficients of f and Landau-Mignotte classical results, one can obtain approximations of the complex roots of f efficiently and thus Assumption 2 also holds.

For using symmetries, the corresponding variety is composed only of the orbit of $(\alpha_1, \ldots, \alpha_{i-1})$, More precisely, the variety is $G_f \cdot (\alpha_1, \ldots, \alpha_{i-1})$. Thus all the results given in Section 4.2 can be applied here.

In general, it is easy to compute a first good approximation of the stabilizer of operands by just reading their support. If a target polynomial A has its support $S_A = \{x_{i_1}, \ldots, x_{i_k}\}$ then the corresponding element $A(\alpha_{i_1}, \ldots, \alpha_{i_k})$ in K is in fact an element of the subfield $\mathbb{Q}(\alpha_{i_1}, \ldots, \alpha_{i_k})$ (in other words, A is stable under the action of $\mathrm{Stab}(G, [i_1, \ldots, i_k])$). Thus, we can easily apply Theorem 4.5 from the support of each operands.

For example, if one wants to compute the product of two elements represented by A and B then one first read the support S of A and B. Let x_i be the highest variable in S. Thus one can compute the product AB by considering the triangular subset $\{g_1, \ldots, g_i\}$ or the corresponding sub-variety. This can be done in general, but suppose the set $S = \{x_{e_1}, \ldots, x_{e_s} = x_i\}$ very sparse, then one can use the symmetries to increase the efficiency. As explained before, the result is in $\mathbb{Q}(\alpha_{e_1}, \ldots, \alpha_{e_s})$ thus, one have to compute the corresponding sub-variety in order to apply modular evaluation-interpolation technique. This equiprojectable variety can be easily described from the action of the Galois group. More explicitly, it corresponds to an *orbit-tree* where elements of the orbit of α_{e_1} under the action of the Galois group G_f are roots and the sons of an element α_{e_j} in this tree are the elements of the orbit of $\alpha_{e_{j+1}}$ under the action of $\mathrm{Stab}(G_f, [e_1, e_2, \ldots, e_j])$. Such orbit tree can be easily computed from G_f and it can be assumed that it has been already precomputed (see [17] for more details on orbit-tree).

Dynamic Sparse Representation: As explain in the Remark 12, the representation of elements in K can be now somehow *dynamic*. More precisely, the computed elements are *sparsely* represented in different subfields of K_f. By this *dynamic* approach, computations of many arithmetic operations gain in efficiency, since normal form computations in K_f may have the worst time complexity but will be used only a few times. We may call this approach *dynamic sparse representation*.

Thus, we can restate the modular evaluation-interpolation recursive process of [5] efficiently for our case. Hereafter, we present such algorithms using subprocedures with names giving easily their definition. Sequences considered in these

algorithms are indexed by elements of a group transversal. For a giving operation `op` between two operands A and B, the *evaluation* operation can be implemented as follows. (Note that the sequences are indexed by elements of groups transversals.)

Algorithm 2: `Eval-op`(A,B)

$S_A := \mathrm{Support}(A)$
$S_B := \mathrm{Support}(B)$
$S_{AB} := S_A \cup S_B$
$Trans := \mathrm{Stab}(G_f, S_{AB})\backslash\backslash G_f$
$vA := \mathtt{Eval-op}(A, [\alpha^\sigma \ : \ \sigma \in \mathrm{Stab}(G_f, S_A)\backslash\backslash G_f])$
$vB := \mathtt{Eval-op}(B, [\alpha^\sigma \ : \ \sigma \in \mathrm{Stab}(G_f, S_B)\backslash\backslash G_f])$
$vA \,\mathrm{op}\, B := [\mathrm{op}(vA[\overline{\sigma}^A], vB[\overline{\sigma}^B]) \ : \ \sigma \in \mathrm{Stab}(G_f, S_{AB})\backslash\backslash G_f]$
/* $\overline{\sigma}^A$ is the representative of σ in $\in \mathrm{Stab}(G_f, S_A)\backslash\backslash G_f]$*/
Return vAB

For the *interpolation* operation, we can restate it, in a general form, by using the set of indexes which contains the support of the output.

Algorithm 3: `Interpol`$(vA \,\mathrm{op}\, B, S = [e_1, \ldots, e_s])$

if S is empty **then**
  Return $vA \,\mathrm{op}\, B[1]$
end if
$Trans := \mathrm{Stab}(G_f, [e_1, \ldots, e_{s-1}])\backslash\backslash G_f$
for $\sigma \in Trans$ **do**
  $lpt := [(\alpha_{e_1^\sigma}, \ldots, \alpha_{e_{s-1}^\sigma}, o) \ : \ o \in \mathrm{Stab}(G_f, [e_1^\sigma, \ldots, e_{s-1}^\sigma]) \cdot e_s]$
  $T_\sigma(X) := \mathtt{InterpolUnivariate}(lpt, vA \,\mathrm{op}\, B[\sigma])$
end for
$d_{e_s} := |\,\mathrm{Stab}(G_f, [e_1, \ldots, e_{s-1}])|/|\,\mathrm{Stab}(G_f, [e_1, \ldots, e_s])|$
for $i = 0, \ldots, d_{e_s} - 1$ **do**
  $vCoeff := [\mathrm{CoeffOfDeg}(i, T_\sigma) \ : \ \sigma \in Trans]$
  $T_i := \mathtt{Interpol}(vCoeff, [e_1, \ldots, e_{s-1}])$
end for
Return $\sum_{i=0}^{d_{e_s}-1} T_i \, X_s^i$.

6. Practical results

In this section, we present practical experiments with the algorithms presented in the preceding sections. We present the real gain of efficiency of using symmetries for the zero-test and during the evaluation-interpolation process. All these implementations were realized with the computational algebra system Magma 2.17 (see [4]).

Zero-test. In the following table, we compare execution timings of the function `NormalForm` which can be used as a zero-test and the implementation of the function `Zero_Test` (see Algorithm 1).

We used two different triangular sets for our benchmarks:

- The first one, G_1, is the output of the algorithm described in [19] applied to $f(x) = x^8 - 3x^7 - 5x^6 + 14x^5 + 8x^4 - 16x^3 - 2x^2 + 5x - 1 \in \mathbb{Q}[x]$. This polynomial is extracted from the database `Galpols` of MAGMA. By this way, a Gröbner basis G_1 of a splitting ideal of f is obtained. The residue class ring $\mathbb{Q}[x_1, \ldots, x_8]/\langle G_1 \rangle$ is a splitting field of f and the setwise stabilizer of the ideal $\langle G_1 \rangle$ is up to an isomorphism the Galois group of f. This subgroup of S_8 is generated by $\{(1,2), (3,4), (5,6), (7,8), (1,5)(2,6)(3,7)(4,8)\}$ and its cardinal is 128.
- The second triangular basis G_2 is the set
$\{x_8 + x_7,\ x_7^2 + (x_6 + x_3),\ x_6 + x_5 + x_4,\ x_5^2 + x_5x_4 + x_4^2,$
$x_4^3 - 1,\ x_3 + x_2 + x_1,\ x_2^2 + x_2x_1 + x_1^2,\ x_1^3 - 1\} \subset \mathbb{Q}[x_1 \ \ldots \ x_n]$
With this second ideal $\langle G_2 \rangle$, we are not in a splitting field case. The setwise stabilizer H of $\langle G_2 \rangle$ is generated by $\{(1,2), (4,5), (7,8), (3,6)(4,7)(5,8)\}$ and contains 16 permutations.

In this table, timings are in seconds and computations had been done with a Pentium(R) Dual-Core 2,3 GHz processor with 4 GB of RAM. In the column `Zero_Test`, when a polynomial belongs to the ideal, some lucky primes are needed to certify the result. In practice, the multi-modular approach enables the parallelization of computations. However, here we do not use such capabilities since MAGMA do not provide parallel features. We just simulate them. In all cases, primes used are the smallest lucky primes needed for certifying the result. In the fourth column, the maximum timings of computations modulo the different primes is given and $(\times m)$ means that m primes are used for certification. Timings needed by the function `NormalForm` of MAGMA used as an ideal membership test appear in the last column. For the computation of the setwise stabilizers of the projected ideals in these two cases, we used an implementation of the algorithm `EFG` of [16]. For any projected ideals of $\langle G_1 \rangle$ used for these zero-tests, this computation requires less than 0.06 s and, in the case of $\langle G_2 \rangle$, less than 0.02 s.

Case	Polynomial A	$A \in G$	`Zero_Test`	`NormalForm`
G_1	A_1	true	0.02 ($\times$18)	40.14
G_1	A_2	false	0.02	5.04
G_1	A_3	false	< 0.01	18.3
G_1	A_4	true	7 ($\times$13)	7.03
G_1	A_5	false	0.02	4
G_1	A_6	true	1.18 ($\times$17)	28.62
G_2	A_7	true	6.26 ($\times$16)	7.96
G_2	A_8	false	< 0.01	1.48
G_2	A_9	true	1.04 ($\times$24)	1.15
G_2	A_{10}	false	0.02	19.68

$A_1 = f(x_1)^2 + f(x_1)$	$A_2 = (x_8 + x_7^2)^8$
$A_3 = (x_1 + 2x_2^2 + 3x_3^3 + 4x_4^4)^8$	$A_4 = \left(\sum_{i=1}^{8} x_i^5\right)^2 - 74529$
$A_5 = (x_1^5 + x_5^5 + x_8^5)^2$	$A_6 = \left(\frac{f(x_8)-f(x_7)}{x_8-x_7}\right)^2$
$A_7 = \left(\sum_{i=1}^{8} x_i^5\right)^{10}$	$A_8 = \left(\sum_{i=1}^{8} i x_i^5\right)^6$
$A_9 = (x_8x_6 - x_8x_3)^{40} - 3^{19}[x_5x_4(x_2 + x_1) + (x_5 + x_4)x_2x_1 - 2]$	$A_{10} = x_8^{200}$

In general, the gain of efficiency mainly depends on the size of the stabilizer (the greater the stabilizer is, the more efficient is our approach). Moreover, experiments suggest that a significant part of efficiency in modular computations is due to the fact that polynomials are sparse modulo small prime numbers.

Multiplication with Modular Evaluation-interpolation. In this paragraph, execution timings for a modular evaluation-interpolation process using symmetries or not are compared. More precisely, we consider the case of modular multiplications in a splitting field. We use the modular variety corresponding to the splitting field of $f = x^{12} - 6x^{11} + x^{10} + 50x^9 - 50x^8 - 166x^7 + 187x^6 + 296x^5 - 258x^4 - 304x^3 + 107x^2 + 142x + 23$ modulo the lucky prime 15973 for which f splits completely. The order of the Galois group of f, which is also the cardinality of V_p, is equal to 3072. The following table shows the timings in seconds of modular evaluation-interpolation on V_p of multiplications of two elements. We compare the case where the computation is done without using symmetries by considering an almost full support of the computed result (from the least variable up to greatest one in the product) with the case of sparse one (the union of the support of the operands). In a Gröbner point of view, we compare the same computation done modulo the entire modular basis G_p (when we consider the almost full support) with the sparse computation by using sparse sub-variety and evaluation-interpolation techniques.

The almost full supports are denoted with AF in the table. As mentioned above, the case where the almost full support is used can be seen as the one where symmetries are not used.

Support	Timings	Support	Timings
AF $[x_1, \ldots, x_{12}]$	30.98	AF $[x_1, \ldots, x_8]$	1.9
$[x_8, \ldots, x_{12}]$	0.12	$[x_1, x_4, x_7, x_8]$	0.20
$[x_2, x_4, x_6, x_8, x_{12}]$	0.07	$[x_6, x_7, x_8]$	0.10
AF $[x_1, \ldots, x_{10}]$	7.8	AF $[x_1, \ldots, x_5]$	0.15
$[x_3, x_4, x_7, x_8, x_{10}]$	0.07	$[x_1, x_2, x_3]$	0.10
$[x_1, x_3, x_{10}]$	0.00	$[x_2, x_3]$	0.00

As one can see, using symmetries provides an important gain of efficiency which well corresponds to the theory (see Theorem 4.5). As in the case of zero-test computations, the gain of efficiency mainly depends on the size of the Galois group and on the size of the support. Using dynamic sparse representation of the elements in K_f brings gain of efficiency when the operands have sparse support. Moreover, this gain can be computed from the supports, the almost full in comparison of the sparse one, more precisely, it corresponds to the index between the subfield represented by the sparse support and the one defined by the almost full. Thus, the use of the dynamic sparse representation can be chosen during the computation in function of the support of the operands.

References

[1] I. Abdeljaouad-Tej, S. Orange, G. Renault, and A. Valibouze. Computation of the decomposition group of a triangular ideal. *Appl. Algebra Engrg. Comm. Comput.*, 15(3-4):279–294, 2004.

[2] H. Anai, M. Noro, and K. Yokoyama. Computation of the splitting fields and the Galois groups of polynomials. pages 29–50, 1996.

[3] A. Aubry, A. Valibouze. Using Galois ideals for computing relative resolvents. *J. Symbolic Comput.*, 30(6):635–651, 2000.

[4] W. Bosma, J. Cannon, and C. Playoust. The Magma algebra system. I. The user language. *J. Symbolic Comput.*, 24(3-4):235–265, 1997. Computational algebra and number theory (London, 1993).

[5] A. Bostan, M. Chowdhury, J. van der Hoeven, and E. Schost. Homotopy methods for multiplication modulo triangular sets. *ArXiv e-prints*, January 2009.

[6] H. Cohen. *A Course in Computational Algebraic Number Theory*, volume 138 of *Graduate Texts in Mathematics*. Springer-Verlag, 1993.

[7] J. Cox, D. Little and D O'Shea. *Ideals, Varieties, and Algorithms*. Undergraduate Texts in Mathematics. Springer-Verlag, 1991.

[8] X. Dahan and É. Schost. Sharp estimates for triangular sets. In *Proceedings of the 2004 international symposium on Symbolic and algebraic computation*, pages 103–110. ACM, 2004.

[9] G. M. Diaz-Toca and H. Lombardi. Dynamic Galois Theory. *J. Symb. Comput.*, 45(12):1316–1329, 2010.

[10] Lionel Ducos. Construction de corps de décomposition grâce aux facteurs de résolvantes. *Comm. Algebra*, 28(2):903–924, 2000.

[11] K. Geissler and J. Klüners. Galois group computation for rational polynomials. *J. Symbolic Comput.*, 30(6):653–674, 2000. Algorithmic methods in Galois theory.

[12] L. Langemyr. Algorithms for a multiple algebraic extension. In *Effective Methods in Algebraic Geometry*, volume 94 of *Progress in Mathematics*, pages 235–248. Birkhäuser, 1991.

[13] L. Langemyr. Algorithms for a multiple algebraic extension II. In *Applied Algebra, Algebraic Algorithms and Error-Correcting Codes*, volume 539 of *Lecture Notes in Computer Science*, pages 234–233. Springer-Verlag, 1991.

[14] M. Lederer. Explicit constructions in splitting fields of polynomials. *Riv. Mat. Univ. Parma (7)*, 3*:233–244, 2004.

[15] X. Li, M. Moreno Maza, and É. Schost. Fast arithmetic for triangular sets: From theory to practice. *J. Symb. Comput.*, 44(7):891–907, 2009.

[16] S. Orange. *Calcul de corps de décomposition - Utilisations fines d'ensembles de permutations en théorie de Galois effective*. PhD thesis, LIP6, Université Paris VI, 2006.

[17] S. Orange, G. Renault, and K. Yokoyama. Computation schemes for splitting fields of polynomials. In *ISSAC '09: Proceedings of the 2009 international symposium on Symbolic and algebraic computation*, pages 279–286, New York, NY, USA, 2009. ACM.

[18] M. Pohst and H. Zassenhaus. *Algorithmic algebraic number theory*, volume 30 of *Encyclopedia of Mathematics and its Applications*. Cambridge University Press, Cambridge, 1997. Revised reprint of the 1989 original.

[19] G. Renault and K. Yokoyama. A Modular Method for Computing the Splitting Field of a Polynomial. In *Algorithmic Number Theory Symposium*, volume 4076, 2006.

[20] G. Renault and K. Yokoyama. Multi-modular algorithm for computing the splitting field of a polynomial. In *ISSAC '08: Proceedings of the twenty-first international symposium on Symbolic and algebraic computation*, pages 247–254, New York, NY, USA, 2008. ACM.

[21] N. Rennert and A. Valibouze. Calcul de résolvantes avec les modules de Cauchy. *Experiment. Math.*, 8(4):351–366, 1999.

[22] K. Yokoyama. A modular method for computing the Galois groups of polynomials. *J. Pure Appl. Algebra*, 117/118:617–636, 1997. Algorithms for algebra (Eindhoven, 1996).

Sébastien Orange
LMAH, Université du Havre
25 rue Philippe Lebon
76600 Le Havre, France
e-mail: `sebastien.orange@univ-lehavre.fr`

Guénaël Renault
UPMC, Univ. Paris 06
INRIA, Paris-Rocquencourt, POLSYS Project
CNRS, UMR 7606, LIP6
Case 169, 4, Place Jussieu, F-75252 Paris, France
e-mail: guenael.renault@lip6.fr

Kazuhiro Yokoyama
Rikkyo University
3-34-1 Nishi Ikebukuro,
Toshima-ku, Tokyo
171-8501, Japan
e-mail: kazuhiro@rikkyo.ac.jp