# On the Complexity of Linear Algebra Operations over Algebraic Extension Fields

Amir Hashemi[1,2(✉)] and Daniel Lichtblau[3]

[1] Department of Mathematical Sciences, Isfahan University of Technology, 84156-83111 Isfahan, Iran
Amir.Hashemi@cc.iut.ac.ir
[2] School of Mathematics, Institute for Research in Fundamental Sciences (IPM), 19395-5746 Tehran, Iran
[3] Wolfram Research, 100 Trade Center Dr, Champaign, IL 61820, USA
danl@wolfram.com

**Abstract.** In this paper, we study the complexity of performing some linear algebra operations such as Gaussian elimination and minimal polynomial computation over an algebraic extension field. For this, we use the theory of Gröbner bases to employ linear algebra methods as well as to work in an algebraic extension. We show that this has good complexity. Finally, we report an implementation of our algorithms in Wolfram Mathematica and illustrate its effectiveness via several examples.

**Keywords:** Gaussian elimination · Minimal polynomial · Polynomial ideals · Gröbner bases · FGLM algorithm · Algebraic extension fields · Complexity analysis

## 1 Introduction

In field theory, a field extension $\mathcal{K} \subset \mathcal{L}$ is called *algebraic* if every element of $\mathcal{L}$ is a root of some non-zero and monic polynomial over $\mathcal{K}$. In this paper, we are interested in analysing the complexity of performing some linear algebra operations over an algebraic extension field $\mathcal{L}$. In this direction, we concentrate only on carrying out Gaussian elimination on a matrix over $\mathcal{L}$ as well as computing the minimal polynomial of a square matrix over $\mathcal{L}$.

More precisely, assume that $f_1 \in \mathcal{K}[x_1]$ is a monic polynomial of degree $d_1 \geq 2$ over the field $\mathcal{K}$. Then, additions in $\mathcal{K}[x_1]/\langle f_1\rangle$ need $d_1$ operations whereas multiplications require $O(d_1 \log(d_1) \log(\log(d_1)))$ operations. We refer to [9] for more details. If $\alpha_1$ denotes the class of $x_1$ in $\mathcal{K}[x_1]/\langle f_1\rangle$, then this quotient ring is denoted by $\mathcal{K}[\alpha_1]$. Doing an induction, assume that for each $2 \leq i \leq n$, $f_i$ is a monic and reduced polynomial of degree $d_i \geq 2$ in $x_i$, over the ring $\mathcal{K}[x_1, \ldots, x_{i-1}]/\langle f_1, \ldots, f_{i-1}\rangle$. Thus, we obtain the multiple algebraic extension $\mathcal{K}[x_1, \ldots, x_n]/\langle f_1, \ldots, f_n\rangle$ which is denoted by $\mathcal{L} := \mathcal{K}[\alpha_1, \ldots, \alpha_n]$ for


F. Boulier et al. (Eds.): CASC 2023, LNCS 14139, pp. 141–161, 2023.
https://doi.org/10.1007/978-3-031-41724-5_8

simplicity. Let $D = d_1 \cdots d_n$. Performing additions and multiplications in $\mathcal{L}$ require $O(D)$ and $O(4^n D \log(D) \log(\log(D)))$ operations respectively; see [23, Theorem 1] and [18] for more details. Lebreton [22] showed that in the latter bound the number 4 can be replaced by 3. We will mostly be concerned with doing linear algebra over $\mathcal{L}$. Unless stated otherwise, we will work with square matrices of dimension $m \times m$. In prior work, Moreno Maza et al. [26, Theorem 2] proposed an algorithm to compute the inverse of a matrix over $\mathcal{L}$ using

$$O(4^n D(m^{\omega+1/2} + n \max\{d_1, \ldots, d_n\}^{(\omega-1)/2}) \log(D) \log(\log(D)))$$

operations, where $\omega < 2.3728639$ denotes the optimal exponent of matrix multiplication (see [2,21]). If in this bound, we remove the term $m^{\omega+1/2}$, then one gets the cost of calculating the inverse of an element in $\mathcal{L}$, see [26, Theorem 1]. Our focus will be on straightforward but practical implementations of linear algebra on matrices with elements in $\mathcal{L}$. As such, we will not attempt to use asymptotically fast matrix multiplication (so our exponent will be 3), but much of the analysis that follows can be carried over to the asymptotic regime.

An important issue that we address in this paper is the computation of the minimal polynomial of a matrix over an algebraic extension field. The best deterministic approach to compute the minimal polynomial of an $m \times m$ over $\mathcal{K}$ is due to Storjohann [29] (by computing the Frobenius normal form of the matrix) which needs $O(m^3)$ field operations.

Note that in the setting of [9,23,26] (and indeed in much of the literature), the ideal $\langle f_1, \ldots, f_n \rangle$ is generated by a triangular set. In this paper we instead consider an arbitrary zero-dimensional ideal which is not necessarily represented by a triangular set, and show how one is able to perform various kinds of linear algebra computations in $\mathcal{L}$. In consequence, the complexity bounds that we present may not be comparable with the existing bounds for ideals generated by triangular sets. If we assume that, for each $i$, $f_i$ is irreducible over $\mathcal{K}[x_1, \ldots, x_i]/\langle f_1, \ldots, f_{i-1} \rangle$ then $\mathcal{L}$ becomes a field. However this additional assumption is not required in the sequel, and we can work with $\mathcal{L}$ as an extension ring (in which case we do not always have invertibility of ring elements, and hence might be unable to make polynomials in the ring monic). We will note when more restrictive assumptions are being made, such as a field given by a tower of irreducible algebraic extensions or by a primitive element. In the latter case, when the base field is prime, computations can be particularly fast, as we will see in the experimental results.

Since an algebraic extension $\mathcal{L}$ can be represented more generally as a quotient of a polynomial ring by a zero-dimensional ideal, Gröbner bases are a basic tool for doing effective computations in $\mathcal{L}$. Thus, in this paper, by applying particular tools developed for zero-dimensional Gröbner bases, we investigate the complexity of performing some linear algebra operations over the field $\mathcal{L}$. We note that [17] presented an efficient algorithm for computing the minimal polynomial of a matrix over $\mathcal{L}$ by using Gröbner bases. In this paper we will discuss the arithmetic complexity of the method given in [17].

The notion of *Gröbner bases* as well as the first algorithm for their construction were introduced by Buchberger in 1965 in his Ph.D. thesis [7,8]. In 1979,

he improved this algorithm by applying two criteria (known as Buchberger's criteria) to remove some of the superfluous reductions, [5]. Later, [14] described an efficient algorithm to install these criteria on Buchberger's algorithm. Since then, several improvements have been proposed to speed-up the computation of Gröbner bases. In particular, in [13], using linear algebra techniques, the FGLM algorithm was proposed to convert a Gröbner basis (of a zero-dimensional ideal) with respect to any term ordering into a Gröbner basis for the same ideal with respect to another ordering. We exploit FGLM techniques in analyzing worst-case complexity for algorithms we present in this paper. In [6], Buchberger also showed how one might employ Gröbner bases to do computations in algebraic number fields. This general technique plays a role in our implementation section.

The paper [27] also describes a method for computing a matrix minimal polynomial over a finite field. In contrast to the present work, they count field operations as units. This is regardless of whether the field is prime or a power of a prime. The present work, in contrast, accounts for all operations in the base ring (that is, the rationals or underlying prime field). Thus this also takes into consideration the complexity of the extension field representation.

The structure of the paper is as follows. Section 2 reviews the basic notations and terminologies used throughout the paper. In Sect. 3, we discuss the complexity of performing some linear algebra operations over an algebraic extension field. Section 4 describes implementations of our approach presented in Sect. 3 along with experimental results.

## 2 Preliminaries

Throughout this article, we use the following notations. Let $\mathcal{P} = \mathcal{K}[x_1, \ldots, x_n]$ be the polynomial ring where $\mathcal{K}$ is a field. We consider a sequence $f_1, \ldots, f_k$ of non-zero polynomials in $\mathcal{P}$ and the ideal $\mathcal{I} = \langle f_1, \ldots, f_k \rangle$ generated by this sequence. We assume that each $f_i$ has total degree $d_i \geq 2$. Furthermore, we denote by $\mathcal{R}$ the quotient ring $\mathcal{P}/\mathcal{I}$. Any element of this ring is given by $[f] := f + \mathcal{I}$ where $f \in \mathcal{P}$.

For us, a *term* is a power product $x^\alpha := x_1^{\alpha_1} \cdots x_n^{\alpha_n}$ of the variables $x_1, \ldots, x_n$ where $\alpha = (\alpha_1, \ldots, \alpha_n)$. Let us fix a term ordering $\prec$. The *leading term* of a polynomial $f \in \mathcal{P}$, denoted by $\mathrm{LT}(f)$, is the greatest term (with respect to $\prec$) appearing in $f$. The coefficient of $\mathrm{LT}(f)$ in $f$ is called the *leading coefficient* of $f$ and is denoted by $\mathrm{LC}(f)$. The product $\mathrm{LM}(f) := \mathrm{LC}(f) \cdot \mathrm{LT}(f)$ is the *leading monomial* of $f$. The *leading term ideal* of $\mathcal{I}$ is defined as $\mathrm{LT}(\mathcal{I}) = \langle \mathrm{LT}(f) \mid 0 \neq f \in \mathcal{I} \rangle$. For a finite set $G \subset \mathcal{P}$, $\mathrm{LT}(G)$ denotes the set $\{\mathrm{LT}(g) \mid g \in G\}$.

A finite subset $G \subset \mathcal{I}$ is called a *Gröbner basis* for $\mathcal{I}$ with respect to $\prec$, if $\mathrm{LT}(\mathcal{I}) = \langle \mathrm{LT}(G) \rangle$. A Gröbner basis is called *minimal* if all leading coefficients are unity and in addition it contains no redundant elements, that is, no leading term is divisible by the leading term of a different element. From here on we assume all Gröbner bases to be minimal. A minimal Gröbner basis is called *reduced* if no term in any polynomial in the basis is divisible by the leading term of a different element. One of the most immediate and important applications

of Gröbner bases is the following result (which is referred to in the literature as Macaulay's theorem) allowing us to find a basis for $\mathcal{R}$ as a $\mathcal{K}$-vector space.

**Proposition 1 ([11, Proposition 4, page 250]).** *Let $G$ be a Gröbner basis of the ideal $\mathcal{I} \subset \mathcal{P}$. Then, the* normal set $N(G) := \{[u] \mid u \text{ is a term and } u \notin \langle \mathrm{LT}(G) \rangle\}$ *forms a basis for $\mathcal{R}$ as a $\mathcal{K}$-vector space. This is known as the normal set for the basis $G$.*

By abuse of notation, we will refer to basis elements of $N(G)$ by the minimal terms that generate them.

It is well-known that the remainder of the division a polynomial $f$ by a Gröbner basis $G$ with respect to $\prec$ is unique and is denoted by $\mathrm{NF}_G(f)$. We shall notice that $\mathrm{NF}_G$ provides a $\mathcal{K}$-linear map from $\mathcal{P}$ to $\mathcal{R}$ and in consequence we have $\mathcal{R} = \{[\mathrm{NF}_G(f)] \mid f \in \mathcal{P}\}$. From the finiteness theorem (see [11, Theorem 6, page 251]), we know that if $\mathcal{I}$ is zero-dimensional then $N(G)$ is finite and its size is the dimension of $\mathcal{R}$ as a $\mathcal{K}$-vector space. Subsequently, this size will be considered as a factor in our complexity analysis. An immediate corollary to the above proposition is that if the product of terms $t_1 t_2$ belongs to $N(G)$ then each factor lies in $N(G)$.

**Definition 1 ([15, page 52]).** *Let $\mathcal{I} \subset \mathcal{P}$ be any zero-dimensional ideal. We define the* degree *of $\mathcal{I}$ as the cardinality of $N(G)$; and we denote it by* $\deg(\mathcal{I})$.

**Definition 2.** *Let $t$ be a power product in the normal set $N(G)$ and $x$ be a variable in the defining ideal. If $xt$ does not lie in $N(G)$ then we call it a* boundary term. *The set of all boundary terms associated to $G$ is denoted by $B(G)$.*

Since all elements of $\mathrm{LT}(G)$ lie in $B(G)$, we see that $|G| \leq |B(G)| \leq n|N(G)|$. Also recall a simple result in Proposition 2.1 of [13]: each element of $B(G)$ is either an element of $\mathrm{LT}(G)$ or else a product of the form $x_i t$ where $t \in B(G)$. We refer to [4,11] for more details on the theory of Gröbner bases.

Now let us recall some facts concerning algebraic extension fields. A finite algebraic extension field $\mathcal{L}$ of $\mathcal{K}$ is a field $\mathcal{K}(\alpha_1, \ldots, \alpha_n)$ where the $\alpha_i$'s are algebraic over $\mathcal{K}$. According to Kronecker's construction, we have the $\mathcal{K}$-algebra homomorphism

$$\psi : \mathcal{P} \to \mathcal{K}(\alpha_1, \ldots, \alpha_n)$$

defined by $x_i \mapsto \alpha_i$. It is well-known that there exist polynomials $f_1, \ldots, f_n \in \mathcal{P}$ such that $\mathrm{Ker}(\psi) = \langle f_1, \ldots, f_n \rangle$. From now on, we denote this ideal by $\mathcal{I}$; it is a maximal (and zero-dimensional) ideal of $\mathcal{P}$. It is clear that $\mathcal{K}(\alpha_1, \ldots, \alpha_n)$ is isomorphic to $\mathcal{K}[x_1, \ldots, x_n]/\mathcal{I}$ as a $\mathcal{K}$-algebra (with each $\alpha_i$ being the equivalence class of $x_i$ modulo $I$). For more details on the relation of the Gröbner bases to the algebraic extension fields, we refer to [1,6].

In the subsequent sections we work with the quotient ring $\mathcal{K}[x_1, \ldots, x_n]/\mathcal{I}$ where $\mathcal{I}$ is not necessarily represented by a triangular set. Instead it will be represented by the reduced Gröbner basis $G = \{g_1, \ldots, g_t\}$ with respect to a given term ordering $\prec$. In some cases this might include finding a primitive element that generates $\mathcal{I}$ (in which case there is an obvious equivalence to a

triangular set representation). We define $D$ to be $\deg(\mathcal{I})$ (that is, $D$ is the size of the normal set). By the well-known Bézout theorem, we have $D \leq d^n$ where $d$ is the maximum degree of a generating set of $\mathcal{I}$. We give some indication of the complexity of computing a primitive element that generates $\mathcal{I}$ in Sect. 4. We shall note that in our complexity analysis in the next section, we do not take into account the complexity of computing the reduced Gröbner basis $G$. Dickenstein et al. [12] have shown that if the zero-dimensional ideal $\mathcal{I}$ is generated by polynomials of degree at most $d$ then its reduced Gröbner basis with respect to $\prec$ can be computed within the arithmetic complexity $d^{O(n^2)}$, see also [16].

For reasons that will be clarified later in this paper, we may also assume that algebraic extensions have primitive elements, that is, can be generated by a single algebraic element of the multiplicative group of the field (in practice this will be a linear combination of the given set of generating elements). See [4,19] for details regarding computation and use of primitive elements.

A common way of defining an extension field using multiple elements is to have the $i$-th element defined as a solution of a monic polynomial $f_i$ in the new variable $x_i$, with coefficients of the non-leading terms being polynomials in the prior elements. In particular, if $\{f_1, \ldots, f_n\}$ forms a triangular set then we have such a representation. In the setting of triangular sets, the size $D$ of the extension is easily seen to be the product of the degrees of the $f_i$ in the corresponding main variables $x_i$. So we have $n \leq \log_2(D)$ or, stated differently, $D \geq 2^n$ (we tacitly assume no extension elements are trivial, that is, linear combinations of previous elements, so all generators are algebraic elements of degree at least 2). In the case that $\{f_1, \ldots, f_n\}$ is a triangular set, we will refer to the corresponding extension as a "tower extension". Note that algebraic fields need not be given as tower extensions, as the next example shows.

*Example 1.* The ideal given by the polynomials $\{x^2 + xy + 2, y^2 + yz - 3, z^2 - zx + zy + 4x + 3y + 5\}$ is in terms of $n = 3$ variables and hence $2^n = 8$. A Gröbner basis for this ideal is $\{2004 - 1656z + 83z^2 + 210z^3 - 90z^4 + 61z^5 - 3z^6 + 3z^7, 356844 + 252448y + 202412z - 27327z^2 + 35961z^3 - 14627z^4 + 834z^5 - 807z^6, -166836 + 378672x - 205968z - 11873z^2 - 58193z^3 - 857z^4 - 2934z^5 + 219z^6\}$. So the normal set has the size $D = 7$ and this is less than $2^n$.

## 3 Complexity Results

In this section, we discuss the complexity analysis of computing the inverse of an algebraic number as well as performing some of the well-known linear algebra operations over an algebraic extension field. We assume unless stated otherwise that the extension field is defined by $n$ algebraic elements and represented by a Gröbner basis $G$.

### 3.1 Multiplication Table

In some of the theorems that follow we will require a fast means of reducing products of pairs of elements in the normal set $N(G)$ into linear combinations

of elements in $N(G)$. To this end we create a table of these products and their corresponding reduced forms. We will assume that table elements can be stored and found in $O(n)$ time; in implementations this might be accomplished using for example a hash table on the exponent vectors. Once we have such a table, every reduction of such a product is $O(nD)$ operations where $n$ is the number of algebraic elements defining the ideal (this is simply the cost of writing that many terms).

Given a polynomial $p_1$ of length $l_1$ and a reducing polynomial $p_2$ of length $l_2$, where terms in $p_1$ are comprised of products of two normal set elements and those in $p_2$ are only normal set elements, we make the assumption that the reduction can be performed in $O(nl_2)$ steps, that is, the length of the polynomial being reduced does not matter. In practice this can be achieved for example by using a dense data structure for the elements in $N(G)$ and hashing all exponent vectors to locate their position in that structure; we regard this as a preprocessing step. Since we will also need to look up term reductions after we compute them, we have another cost of $O(nD^2)$. For our purposes we will assume $n \leq D$. In the theorem below we ignore these costs because they are smaller than the actual complexity. We now give the complexity of computing a multiplication table, as we will use this in the sequel (in particular in Subsect. 3.2). Moreover, as this is an extension of the FGLM method of basis conversion [13], it is thus of interest in its own right.

**Theorem 1.** *Given a reduced Gröbner basis $G$ for an ideal $\mathcal{I}$ defined by $n$ algebraic elements, with normal set $N(G)$ of size $D$, we can compute a multiplication table for all pairs in $N(G)$ in $O(D^4)$ arithmetic operations.*

*Proof.* Denote the elements of $N(G)$ as $u_1, u_2, \ldots, u_D$ with $u_1 \prec u_2 \prec \cdots \prec u_D$. We have at most $O(D^2)$ distinct power products in the set of product pairs. As the first step, we order these products. We consider first the elements of $N(G)$, then the elements of $\mathrm{LT}(G)$, next the elements of $B(G) \setminus \mathrm{LT}(G)$ and finally the remaining term products. For this ordering, we use the same term ordering $\prec$ as was used for the computation of $G$. It is well known that sorting a set of size $k$ comprised of elements of size $n$ in this way is no worse than $O(nk \log(k))$, so this will not dominate the complexity analysis. Now, we hash this sorted list of terms and it costs $O(nD^2)$ arithmetic operations (for simplicity we can use e.g. natural numbers $1, 2, 3, \ldots$ as the range of the hash function). Within the complexity $O(n)$ we can determine whether a term $u$ belongs to $\mathrm{LT}(G)$ or not. The same holds for membership in other subsets of term products. Below, we keep the normal form of each term in the form $b_1u_1 + \cdots + b_Du_D$ and assume that each elements $g \in G$ is represented of the form $\mathrm{LT}(g) - b_1u_1 + \cdots - b_Du_D$.

Now assume that we are given a product $u$. Then four cases may occur:

Case (1) $u \in N(G)$: This case comprises a "base case", that is, we need no replacements for them.

Case (2) $u \in \mathrm{LT}(G)$: Testing for membership in $\mathrm{LT}(G)$ is $O(n)$. In this case, we have the normal form of $u$ with no calculations other than to list the $O(D)$ terms.

Case (3) $u \in B(G) \setminus \mathrm{LT}(G)$: In this case, we are able to write $u$ as $xt$ for a variable $x$ and a term $t$ with $t \in B(G)$. Since $t \prec u$, it already has a rewrite as a sum of elements in $N(G)$. Thus we have $u = xt = x(b_1u_1 + \cdots + b_Du_D)$. As each $xu_i \prec u$ we have $xu_i = c_{i,1}u_1 + \cdots + c_{i,D}u_D$. Thus we can rewrite $u$ at cost $O(n^2 + nD + D^2)$. Here the $D^2$ contribution is for the actual rewriting, the $n^2$ is the cost of finding such $t \in B(G)$ (we have to check up to $n$ variables, and each check is $O(n)$ to compute the exponent vector of $u/x_j$ and then to do a lookup on that vector), and the $nD$ component comes from having to locate $D$ reductions for the $xu_i$ terms.

Case (4) $u \notin N(G) \cup B(G)$: Then we can find a variable $x$ and term $t$ such that $u = xt$ and $t \notin N(G)$. Since $t \prec u$ it already has a reduction and thus we have $u = x(b_1u_1 + \cdots + b_Du_D)$ for base ring elements $a_i$ and terms $u_i \in N(G)$. For each such product we have $xu_i \in B(G)$. Since we already handled terms from $B(G)$ in case (3), by applying an induction, we have $xu_i = c_{i,1}u_1 + \cdots + c_{i,D}u_d$. Here the main point is also the choice of the variable $x$. Indeed, any variable $x$ appearing in $u$ will work, and the corresponding term $t = u/x$ will already have a reduction due to the order in which we compute these. The cost of finding $x$ and the lookup cost for the reduction of $t$ are both clearly $O(n)$. Similarly the cost of finding reductions for the $O(D)$ terms $xu_i$ is $O(nD)$. Thus we can reduce $u$ at cost $O(n + nD + D^2)$.

As we have $O(nD)$ terms for case (3) and $O(D^2)$ terms for case (4), and $n \leq D$, the total cost is bounded by $O(D^4)$. □

We shall note that this proof is in essence the same argument as in the FGLM reference [13], except that, in our paper, we also take into account the number of generators $n$. However, since we have $n \leq D$ the overall complexity given in [13] does not require this accounting. We remark that this bound is pessimistic. Indeed, it is commonly the case that $|G|$ is $O(D)$ rather than $O(nD)$. Also we need not consider elements in $B(G)$ that are not also in the set of products of pairs in $N(G)$. This is relevant for instance when $G$ is a lexicographic Gröbner basis and the smallest variable is in general position (so the shape lemma applies). In this case the set of products is actually $O(D)$ and only one Gröbner basis reduction is needed since only one element from $B(G)$ appears in the set of products.

In the special case where we have a tower extension, we can work with a lexicographical ordering. In this case the original polynomials defining the extension are already a Gröbner basis although possibly not fully reduced. Thus we have $|G| = n$, so we can drop a factor of $D$ in the complexity analysis. Also in this case we have $D = d_1d_2 \cdots d_n$ where $d_i$ is the degree in the extension-generating variable $x_i$ of the $i^{th}$ polynomial $f_i$, and by assumption of nontriviality we have $d_i \geq 2$. The set of products from $N(G)$ lies in the Minkowski sum of $N(G)$ with itself, and as the normal set lies in a rectangular prism in $\mathbb{Z}^n$, this sum has cardinality $2^nD \leq D^2$. Thus we compute rewrites for strictly fewer than $D^2$ terms in computing the multiplication table. Specifically, for any $d_i \geq 3$ we have a factor $d_i/2$ reduction in the number of operations for the largest component of the complexity.

### 3.2 Algebraic Inverse

Based on the structure of the FGLM algorithm, Noro [28] presented a simple and effective method for computing the inverse of an algebraic number (another method is given in [6]). To explain Noro's method, let $N(G) = \{b_1, \ldots, b_D\}$ be a basis for the $\mathcal{K}$-vector space $\mathcal{P}/\mathcal{I}$ with $\mathcal{I} = \langle f_1, \ldots, f_n \rangle$ (recall we take as basis the normal set for a given Gröbner basis $G$ of the extension ideal $\mathcal{I}$). Furthermore, let $\sigma$ be an element of $\mathcal{K}(\alpha_1, \ldots, \alpha_n)$. There exists a polynomial $f$ such that $f = \psi^{-1}(\sigma)$, where $\psi$ is the map from Sect. 2 taking $x_i$ to $\alpha_i$. Then the inverse of $\sigma$ is $\sum_{i=1}^{D} c_i \psi(b_i)$, where the $c_i$'s belong to $\mathcal{K}$ and satisfy $\sum_{i=1}^{D} c_i \mathrm{NF}_G(f b_i) = 1$.

Let $d = \deg(f)$ and $\tau$ be the number of non-zero terms of $f$. To simplify the final complexity bounds, we assume here and throughout that $d$ and $\tau$ are less than or equal to $D$. If these inequalities do not hold, then it suffices to compute the normal form of $f$ with respect to $G$, and this does not change the correctness of this approach. These simplifications are considered in the following subsections.

In the next theorem we assume we have already precomputed a multiplication table, so that cost is not included in the complexity analysis.

**Theorem 2.** *The arithmetic complexity of computing the inverse of $\sigma$ is $O(nD^3)$.*

*Proof.* First we form a generic linear combination $p$ of the normal set elements, that is, $p = \sum_{i=1}^{D} c_i \psi(b_i)$. This will be our inverse and so we must determine values of the parameters. We next multiply by $\sigma$ at cost $O(nD^2)$. We now reduce $\sigma p - 1$. Using the precomputed multiplication table (see Theorem 1) we rewrite each of the $O(D^2)$ terms as a linear combination of $N(G)$ at cost $O(nD)$. Thus the total of reducing this product is $O(nD^3)$. We set each coefficient to zero. This gives a linear system of $D$ equations in $D$ unknowns. The arithmetic cost of solving is bounded by $O(D^3)$ and so the $O(nD^3)$ reduction is the dominating term in the complexity. □

**Theorem 3.** *Keeping the above notations, and assuming our extension is given by a primitive element, the arithmetic complexity of computing the inverse of $\sigma$ is $O(D^3)$.*

*Proof.* As before, we form a generic linear combination $p = \sum_{i=1}^{D} c_i \psi(b_i)$. Again we must determine values of the parameters. We next multiply by $\sigma$ at cost $O(D^2)$. The primitive element representation implies that the product has fewer than $2D = O(D)$ distinct terms. We now reduce $\sigma p - 1$ by the polynomial that defines our primitive element. Since each reduction of the top monomial reduces the degree, this entails $O(D)$ reduction steps. As the reducing polynomial has at most $O(D)$ terms, the complexity of each reducing step is also $O(D)$, so the total cost of reducing $\sigma p - 1$ is $O(D^2)$. Setting the reduced polynomial to zero coefficient-wise gives $D$ linear equations in the $D$ unknown parameters. Solving this system is $O(D^3)$ operations in the base field. As this dominates the prior parts we achieve the claimed bound. □

We remark that if the field in question is an algebraic extension of a prime field by a single irreducible polynomial (hence has a primitive element), well known asymptotically fast methods for computing products and inverses (e.g. based on Fourier-type transforms and the half-GCD respectively) become quite practical. In such cases these are in fact what we use. When the base field is a prime field but the extension is not given by a primitive element, then some of the linear algebra analysis from the next subsection, which uses the multiplication table, will still apply for converting to a primitive element representation and back again.

In the rest of this subsection, we compare our complexity bound presented in Theorem 2 to the bound that one can obtain using the FGLM techniques. In doing so, let us recall some useful results regarding the FGLM algorithm, see [13] for more details. Let $\mathcal{I} \subset \mathcal{P}$ be a zero-dimensional ideal and $D := \deg(\mathcal{I})$. The FGLM algorithm receives as input the reduced Gröbner basis $G_1$ with respect to $\prec_1$, and outputs the reduced Gröbner basis $G_2$ with respect to another term ordering $\prec_2$. The main advantage of this algorithm is the use of linear algebra techniques that make it very efficient in practice. The basic ingredient of this algorithm is the efficient computation of the normal form of a polynomial with respect to $G_1$. For this, one needs to construct the matrix corresponding to the linear map $\phi_i : N(G) \to N(G)$ with $\phi_i(b_\ell) = \mathrm{NF}_{G_1}(x_i b_\ell)$ for each $\ell$ where $N(G) = \{b_1, \ldots, b_D\}$. This leads to the construction of the FGLM table $T(G) = (t_{ij\ell})$ where $t_{ij\ell}$ denotes the $j$-coordinate with respect to $N(G)$ of $\phi_i(x_i b_\ell)$. It is shown that cost of computing the FGLM table is $O(nD^3)$, [13, Proposition 3.1] and this complexity is the dominant factor in the complexity analysis of transforming $G_1$ to $G_2$.

**Theorem 4.** *The arithmetic complexity of computing the inverse of $\sigma$, by using the FGLM table, is $O(nD^5)$.*

*Proof.* Using the Noro's method, we shall compute $\mathrm{NF}_G(fb_\ell)$ for an arbitrary $\ell$. For this, we consider the complexity of computing $\mathrm{NF}_G(x_i b_\ell)$ where $x_i$ is a variable. Using the FGLM table, it is equal to $t_{i1\ell}b_1 + \cdots + t_{iD\ell}b_D$. One can see easily that the cost of computing $\mathrm{NF}_G(x_r x_i b_\ell)$ is $O(D^2)$ field operations. In consequence, the complexity of computing $\mathrm{NF}_G(mb_\ell)$ is $O(ndD^2)$ where $m$ is a term of degree $d$. Since $f$ has $\tau \leq D$ terms then the complexity of $\mathrm{NF}_G(fb_\ell)$ is $O(nD^4)$. Performing these operations for all $\ell$ has the complexity $O(nD^5)$. Note that the bound $O(nD^5)$ includes also the complexity $O(nD^3)$ for computing the FGLM table. Finally, finding the inverse of $\sigma$ is equivalent to finding the $c_i$'s such that $\sum_{i=1}^{D} c_i \mathrm{NF}_G(fb_i) = 1$ and this has the cost $D^3$, ending the proof. □

**Corollary 1.** *By taking into account the complexity of computing a multiplication table (Theorem 1), the worst-case complexity of computing an algebraic inverse by using Theorem 2 is $O(nD^4)$ which is lower than the corresponding worst-case bound that one obtains using the FGLM techniques (Theorem 4).*

*Remark 1.* By applying dynamic evaluation and modular techniques, Langemyr in [20] gave an almost optimal algorithm, i.e. in computing time $O(S^{\delta+1})$ for all

$\delta > 0$, where $S$ is the best known a priori bound on the length of the output, for computing the inverse of $\sigma$.

### 3.3 Gaussian Elimination

In this subsection, we discuss the complexity of performing Gaussian elimination on a given matrix over $\mathcal{K}(\alpha_1, \ldots, \alpha_n)$.

**Theorem 5.** *Let $A$ be a matrix of size $s \times t$ over $\mathcal{K}(\alpha_1, \ldots, \alpha_n)$. Keeping the notations presented in Subsect. 3.2, and assuming either that we have a primitive element or that we have precomputed a multiplication table for $N(G)$, the arithmetic complexity of performing Gaussian elimination on $A$ is given by $O(\min(s,t)stnD^3)$.*

*Proof.* We know that for each $i, j$ there exists polynomial $f_{i,j}$ such that $f_{i,j} = \psi^{-1}(A[i,j])$. Let $d$ be the maximum of $\deg(f_{i,j})$'s and $\tau$ the maximum number of non-zero terms of the $f_{i,j}$'s. From the above discussion we have $n, d, \tau \leq D$. Let $\mathrm{Row}(i, A)$ denote the $i$-th row of $A$. Assume that we want to reduce the first column of $A$ by using $A[1,1]$. Let $\sigma$ be $A[1,1]$. Now, to perform a row reduction operation, one can first compute $\sigma^{-1}$ (see Theorem 2 and corollaries), multiply all the entries of $\mathrm{Row}(1, A)$ by $\sigma^{-1}$ and then expand each entry of $\sigma^{-1}\mathrm{Row}(1, A)$. Recall the cost of inverting $\sigma$ was bounded by $O(nD^3)$. Then the number of field operations for this part is $O(nD^3 + tnD^2)$. Note that $\sigma^{-1}$ and each entry in the first row have length at most $D$ in terms of the elements of $N(G)$, hence all products have length bounded by $D^2$. Reducing each term in such a product under the assumption of a table or a primitive element is $O(nD)$ and gives rise to a result of length $O(D)$, so the full reduction cost is no worse than $O(nD^3)$. As there are $t$ elements to consider, the complexity of making the pivot 1 is $O(tnD^3)$ (and, as with element inversion, using a primitive element can bring this step to $O(tD^2)$ since product lengths become bounded by $2D$). Finally, we shall reduce $\mathrm{Row}(i, A)$ with $i > 1$ by using the new row; i.e. $\sigma^{-1}\mathrm{Row}(1, A)$. The number of field operations to reduce one row is easily seen to be the same as the step of making the pivot equal to 1. We shall repeat this operation for $i = 2, \ldots, s$. All in all, reducing $s - 1$ rows by the first row costs $O(nD^3 + tnD^3 + (s-1)tnD^3)$ which is dominated by $O(stnD^3)$. The number of pivots to reduce beneath is equal to the rank of the matrix, which is bounded by $\min(s,t)$ and this ends the proof. □

We remark that for many purposes one need not make pivots equal to 1, and so the complexity of inverting an element can be avoided. If we work with a primitive element extension and also avoid computing inverses then the complexity above is reduced by a factor of $nD$, to $O(\min(s,t)stD^2)$, excluding costs of pre- and post-processing for using a primitive element. If asymptotically fast methods are used for multiplying algebraic elements, this reduces further to $\tilde{o}(\min(s,t)stD)$ (where the "soft-Oh" notation hides logarithmic factors in $D$).

### 3.4 Minimal Polynomial

In this subsection, we analyse the complexity of computing the minimal polynomial of a square matrix over $\mathcal{K}(\alpha_1, \ldots, \alpha_n)$ by using the algorithm presented in [17]. For the reader's convenience, we recall it here (see Algorithm 1). To explain the complexity of this algorithm, let $A$ be an $m \times m$ matrix over $\mathcal{K}(\alpha_1, \ldots, \alpha_n)$. Then for each $i, j$ there exists a polynomial $f_{i,j}$ such that $f_{i,j} = \psi^{-1}(A[i, j])$. In order to reduce the complexity, we first replace each $f_{i,j}$ by its normal form with respect to $G$. Without loss of generality, assume that $f_{i,j} = \mathrm{NF}_G(f_{i,j})$. Let $d$ be the maximum of $\deg(f_{i,j})$'s and $\tau$ the maximum number of non-zero terms of the $f_{i,j}$'s. In consequence we have $n, d, \tau \leq D$. Furthermore let $p(s) = a_m s^m + a_{m-1} s^{m-1} + \cdots + a_0$ be the minimal polynomial of $A$ where each $a_i \in \mathcal{K}(\alpha_1, \ldots, \alpha_n)$ will be determined. We shall need to compute the sequence $A^2, A^3, \ldots, A^m$. From $p(A) = 0$ we can derive $m^2$ algebraic equations between the $a_i$'s, say $g_{1,1}, \ldots, g_{m,m}$. As we interleave reductions with each step, it is easy to see that these polynomials have degree at most $D$ (in terms of the $x_i$'s). In Algorithm 1, $|X|$ denotes the size of a set $X$.

**Algorithm 1** MinPoly

**Require:** $A_{m \times m}$ a non-zero matrix, and $G$ a Gröbner basis for the ideal $\mathcal{I}$
**Ensure:** The minimal polynomial $p(s)$ of $A$
1: $g_{i,j} := \sum_{t=0}^{m} a_t A^t[i, j]$ for $i, j = 1, \ldots, m$
2: $\mathcal{J} := \langle q_{1,1}, \ldots, q_{m,m} \rangle$ where $q_{i,j} = \mathrm{NF}_G(g_{i,j})$ for each $i$ and $j$
3: $G_1$ := A minimal Gröbner basis for $\mathcal{I} + \mathcal{J}$ with respect to the lexicographical ordering with $x_j \prec_{plex} a_0 \prec_{lex} \cdots \prec_{lex} a_m$ for each $j$
4: $\ell$ := The highest integer $i$ such that $a_i$ appears in a polynomial in $G_1$
5: **if** $a_0, \ldots, a_\ell \in G_1$ **then**
6: **Return** $(s^{\ell+1})$
7: **end if**
8: $r$ := The integer $i$ with $a_0, \ldots, a_{i-1} \in G_1$ and $a_i \notin G_1$ (if $a_0 \notin G_1$, set $r := 0$)
9: $G_2 := G_1|_{a_r=1}$
10: $p := \mathrm{NF}_{G_2}(x^r + a_{r+1} s^{r+1} + \cdots + a_\ell s^\ell)$
11: $\sigma :=$ AlgebraicInverse$(a_\ell)$
12: $p := \sigma \cdot p$
13: **Return** $(p)$

We note that the costly step of Algorithm 1 is the computation of a Gröbner basis of the ideal $\mathcal{I} + \mathcal{J}$ with respect to the mentioned ordering (see the line 2). Below we present a simple and efficient way to compute such a basis. The first point is that we need only a minimal Gröbner basis for $\mathcal{I} + \mathcal{J}$, rather than the reduced one. Let $q_{i,j} = \mathrm{NF}_G(g_{i,j})$ for each $i, j$. Order the $q_{i,j}$'s from the highest leading term to the lowest. Assume that $q_1, \ldots, q_{m^2}$ is this sequence of polynomials. We want to construct recursively, for each $i$, the polynomials $h_i$ and $\tilde{h}_i$. At the beginning, we let $h_1 = q_1$. Suppose that $h_1$ as a polynomial in terms of the $a_i$'s can be written as $p_\ell a_\ell + \cdots + p_0 a_0$ where $p_\ell \neq 0$. Since

$[p_\ell] \in \mathcal{R}$ is invertible, we let $\tilde{h}_1 = w_\ell h_1$ where $[w_\ell p_\ell] = [1]$. Thus, $\mathrm{LT}(\tilde{h}_1) = a_\ell$. Now, for each $i = 2, \ldots, m^2$, we define $h_i = \mathrm{NF}_{\{\tilde{h}_1,\ldots,\tilde{h}_{i-1}\}}(q_i)$. Consider $h_i$ as a polynomial in terms of the $a_i$'s of the form $h_i = p_{i_0}a_{i_0} + \cdots + p_0 a_0$ where $p_{i_0} \neq 0$. We know that $[p_{i_0}] \in \mathcal{R}$ is invertible. Define $\tilde{h}_i = w_{i_0} h_i$ where $[w_{i_0} p_{i_0}] = [1]$. It yields that $\mathrm{LT}(h_i) = a_{i_0}$.

**Proposition 2.** *$G \cup \{\tilde{h}_1, \ldots, \tilde{h}_{m^2}\} \setminus \{0\}$ is a minimal Gröbner basis for $\mathcal{I} + \mathcal{J}$.*

*Proof.* Proceeding by induction, we first show that if $G \cup \{\tilde{h}_1, \ldots, \tilde{h}_{i-1}\}$ forms a minimal Gröbner basis, then $G \cup \{\tilde{h}_1, \ldots, \tilde{h}_i\}$ is a minimal Gröbner basis for the ideal it generates. Since we have

$$\gcd(\mathrm{LT}(\tilde{h}_i), \mathrm{LT}(h)) = 1 \quad \forall h \in G \cup \{\tilde{h}_1, \ldots, \tilde{h}_{i-1}\}$$

the claim follows immediately from Buchberger's first criterion. From the construction of the $\tilde{h}_i$'s, it follows that the ideal generated by $G \cup \{\tilde{h}_1, \ldots, \tilde{h}_{m^2}\} \setminus \{0\}$ is equal to $\mathcal{I} + \mathcal{J}$, which ends the proof. □

*Remark 2.* For the proof of the correctness of Algorithm 1, we refer to [17, Theorem 1]. Our presentation of this algorithm is slightly different from the original version.

*Example 2.* In this example we illustrate the above process step by step to compute the minimal polynomial of a given matrix. Let us consider the matrix presented in [17, Example 2]. We wish to compute the minimal polynomial of the following matrix over the field $\mathbb{Z}_5(\alpha_1, \alpha_2) = \mathbb{Z}_5[x_1, x_2]/\langle x_1^2 + 1, x_2^2 + x_1\rangle$. Let

$$A = \begin{bmatrix} \alpha_1 & 1 & 0 \\ \alpha_1 + \alpha_2 & 2 & 1 \\ 1 & 3 & \alpha_1\alpha_2 + 1 \end{bmatrix}.$$

It is easy to see that $G = \{x_1^2 + 1, x_2^2 + x_1\}$ is a Gröbner basis with respect to $x_1 \prec_{lex} x_2$ for the ideal $\mathcal{I}$ it generates. Let $p(s) = a_3 s^3 + a_2 s^2 + a_1 s + a_0$ be a polynomial vanishing on $A$. Then, with the above notations, we have

$$\begin{aligned}
q_{1,1} &= a_0 + x_1 a_1 + (x_1 + x_2 + 4)a_2 + (2x_1x_2 + x_1 + 2x_2 + 4)a_3 \\
q_{1,2} &= a_1 + (x_1 + 2)a_2 + (3x_1 + x_2 + 1)a_3 \\
q_{1,3} &= a_2 + (x_1x_2 + x_1 + 3)a_3 \\
q_{2,1} &= (x_1 + x_2)a_1 + (x_1x_2 + 2x_1 + 2x_2)a_2 + (x_1 + x_2)a_3 \\
q_{2,2} &= a_0 + 2a_1 + (x_1 + x_2 + 2)a_2 + (4x_1x_2 + 4x_1 + 4x_2 + 3)a_3 \\
q_{2,3} &= a_1 + (x_1x_2 + 3)a_2 + (4x_1x_2 + 2x_1 + x_2)a_3 \\
q_{3,1} &= a_1 + (x_1x_2 + 4x_1 + 3x_2 + 1)a_2 + (2x_1 + x_2 + 3)a_3 \\
q_{3,2} &= 3a_1 + 3x_1x_2a_2 + (3x_1x_2 + 2x_1 + 3x_2 + 3)a_3 \\
q_{3,3} &= a_0 + (x_1x_2 + 1)a_1 + (2x_1x_2 + x_1 + 4)a_2 + (4x_1x_2 + 3x_1 + 4x_2 + 4)a_3.
\end{aligned}$$

Now, we set $h_1 = q_{1,1}$. The coefficient of this polynomial in terms of the $a_i$'s is $2x_1x_2 + x_1 + 2x_2 + 4$. The inverse of this polynomial is $-x_1x_2 - 3 = 4x_1x_2 + 2$.

Therefore, $\tilde{h}_1 = (4x_1x_2 + 2)a_0 + (2x_1 + x_2)a_1 + (x_1x_2 + 2x_1 + 3x_2 + 2)a_2 + a_3$. Following the similar approach, we get

$$\tilde{h}_2 = (x_1x_2 + x_1 + 4x_2 + 3)a_0 + (x_1x_2 + 4x_1 + x_2 + 4)a_1 + a_2$$
$$\tilde{h}_3 = (2x_1 + 2x_2 + 1)a_0 + a_1.$$

One observes that $h_4 = \cdots = h_9 = 0$. Thus $G \cup \{\tilde{h}_1, \tilde{h}_2, \tilde{h}_3\}$ is the desired Gröbner basis for $\mathcal{I} + \mathcal{J}$. By the notations used in the algorithm we have $r = 0$ and $\ell = 3$. Putting $a_0 = 1$ in this basis, and computing the normal form of $p(s)$ with respect to this basis leads to $q(s) := (4x_1x_2 + x_1 + 4x_2 + 4)s^3 + (4x_1 + 3x_2)s^2 + (3x_1 + 3x_2 + 4)s + 1$. The inverse of $4x_1x_2 + x_1 + 4x_2 + 4$ is $\sigma := 2x_2 + 2x_1$. By multiplying $q(s)$ with $\sigma$, we get the minimal polynomial $s^3 + (4\alpha_1\alpha_2 + 4\alpha_1 + 2)s^2 + (2\alpha_1\alpha_2 + 2\alpha_1 + 3\alpha_2 + 4)s + 2\alpha_1 + 2\alpha_2$ for $A$.

*Remark 3.* We remark that we can emulate linear algebra by computing a module Gröbner basis (see for example [24,25]). The $a_i$'s can be seen as defining the matrix columns (these are sometimes called "tag variables" in the literature, and no S-polynomials are formed between distinct pairs of these. This can be enforced either by using a basis algorithm that provides for degree bounds, or else by the expedient of adding relations that all products of $a_i$ pairs vanish. Computing a module Gröbner basis is one means of implementing the approach described in the remarks preceding Proposition 2. We use this as one of the methods in the implementation section.

*Remark 4.* Following the notations used in Algorithm 1, assume that $a_0, \ldots, a_\ell$ belong to $G_1$. Since $a_{\ell+1}$ does not appear in $G_1$ then $A^{\ell+1} = 0$ and in turn we have $a_0 = \cdots = a_\ell = 0$. In this case, the minimal polynomial of $A$ is $p(s) = s^{\ell+1}$. For example, if we consider the matrix

$$A = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix}$$

over the field $\mathbb{Z}_5(\alpha_1, \alpha_2)$ (see the above example) then we have $A^2 = 0_{2\times 2}$ and in turn $G_1 = \{x_1^2 + 1, x_2^2 + x_1, a_0, a_1\}$. Thus, $p(s) = s^2$.

*Remark 5.* For the efficiency of the algorithm, we can apply Algorithm 1 in an iterative way by enumerating the matrices $I, A, A^2, \ldots$ and stopping whenever a linear dependency is detected.

**Theorem 6.** *Keeping the above notations, and assuming either that we have a primitive element or that we have precomputed a multiplication table for $N(G)$, the arithmetic complexity of computing the minimal polynomial of the matrix $A$ is $O(m^4nD^3)$.*

*Proof.* As the first step, we shall compute $A^2, A^3, \ldots, A^m$. From linear algebra, it is well-known that the arithmetic complexity of computing $X^2$ where $X$ is a matrix of size $m \times m$ is bounded by $O(m^3)$. However, since the entries of $A$ are polynomials containing at most $D$ non-zero terms, then we shall take

into account the cost of expanding the entries of $A^2$. The cost of multiplying two polynomials in $n$ variables with $D$ terms is $O(nD^2)$. Thus, to compute $A^2$, we need $O(m^3nD^2)$ field operations. Next we compute the normal form with respect to $G$ of the entries of $A^2$. Recall from the proof of Theorem 2 that reducing an individual product in $A^2$ has complexity $O(nD^3)$. Therefore the total complexity of this operation for all entries of $A^2$ is $O(m^3nD^2 + m^2nD^3)$. In consequence, the number of field operations to calculate $A^2, A^3, \ldots, A^m$ is $O(m^4nD^2 + m^3nD^3)$. Within this complexity, we obtain the $g_{i,j}$'s and each $g_{i,j}$ has at most $D$ terms. The complexity of the rest of the computation is equivalent to the cost of performing Gaussian elimination on a matrix of dimensions $m^2 \times m$, which we showed in Theorem 5 to be $O(m^4nD^3)$ and this finishes the proof. □

*Remark 6.* Assume that $p(s) = a_m s^m + a_{m-1}s^{m-1} + \cdots + a_0$ is the minimal polynomial of the matrix $A$. It is well-known that if $a_0$ is non-zero then $A$ is invertible and its inverse can be compute using the equality $A^{-1} = -a_0^{-1}(a_m A^{m-1} + a_{m-1}A^{m-2} + \cdots + a_1)$. Thus the complexity of Theorem 6 holds true for computing the inverse of $A$ as well. In this case, the determinant of $A$ is $a_0$.

One of our implementations does division-free linear algebra directly. In this case Gröbner basis usage is restricted to interleaving extension field reductions with the matrix operations. We avoid inverting elements in this implementation (and thus typically do not obtain a monic minimal polynomial). As mentioned earlier, this helps to reduce the complexity.

We (mostly) avoid a factor of $n$ if we work with a primitive element. The factor will instead appear in pre- and post-processing steps, where we first replace the $n$ original defining elements by polynomials in the primitive element, and at the end reverse this replacement. We will say more about this when we describe experiments. Another advantage, as noted before, is that we also reduce the complexity of the linear algebra by a factor of $D$.

A further probabilistic complexity improvement is to work not with powers $A^t$ but instead with $A^t v$ where $v$ is a random vector in the base field (this appears to be a folklore approach and we have not found a definitive reference for its origin). This gives rise to a Monte Carlo algorithm that reduces the complexity of the deterministic one by a factor of $m$. This works reliably when the base field is either infinite or a prime field, that is, large compared to the matrix dimension. We remark that similar ideas are used in [27] and in some of the references cited in that work.

## 4 Notes on Implementation and Experimental Results

As mentioned earlier, a reasonable way to work with linear algebra over an algebraic field is to compute a Gröbner basis over a module, with new variables for the matrix columns and a Position-over-Term (POT) term ordering to enforce left-to-right reduction. An implementation of matrix minimal polynomial computation over an algebraic number field can be found in the Wolfram Function Repository:

https://resources.wolframcloud.com/FunctionRepository/resources/MatrixMinimalPolynomial

This way of computing the basis is the one described following Algorithm 1. There are several ways the computations can be made more efficient than the most naive implementation would provide. One improvement is to use a degree-based Gröbner basis for handling the algebraic numbers. This is incorporated into the overall Gröbner basis computations as follows. In order to do linear algebra row reduction, the module variables need to be ordered lexicographically, so we use a block ordering with these ranked highest and lexically between one another. The variables representing the algebraics defining the extension field come lexically after the module variables, and are ordered between themselves by the graded reverse lexicographic (GRL) term order.

Again as noted earlier, sizes (degree and number of terms) in the step of taking matrix products are controlled by interleaving reductions in each powering step. When the base field is the rationals $\mathbb{Q}$ and $n$ and $D$ are fixed, integer sizes grow as $O(m)$ (and it is straightforward to show that this is a worst-case upper bound). Since the arithmetic complexity in terms of matrix dimension $m$ is $O(m^4)$ (again holding all else constant, that is, ignoring the effect of the algebraic extension), we expect the bit complexity to scale as $O(m^6)$ if the bit sizes are too small to allow for arithmetic operations using asymptotically fast methods.

### 4.1 Dependence on Matrix Dimension

In order to assess complexity empirically we conducted a simple experiment. We construct a family of random examples and time them using the code mentioned in the implementation section. Each member of the family is indexed by dimension, from 6 to 34. Matrix elements are random integers between -5 and 5, except the $(1, 1)$ element is $x$ and the $(2, 2)$ element is $y$, where $(x, y)$ satisfy the algebraic relations $9x^2 - 2, 8y^3 - 3$. Timings in seconds are given below.

| | | | | | |
|---|---|---|---|---|---|
| 0.1514 | 0.2596 | 0.4203 | 0.6413 | 0.9326 | 1.3277 |
| 1.9964 | 2.7591 | 3.8360 | 5.8487 | 8.0759 | 11.994 |
| 19.174 | 26.695 | 39.879 | 56.576 | 78.567 | 108.49 |
| 146.50 | 203.79 | 264.15 | 348.10 | 492.03 | 643.46 |
| 843.17 | 1091.1 | 1428.2 | 1758.3 | 2319.5 | |

We fit these to polynomials of degree 5, 6, 7 and 8. This is done in a numerically stable way by reweighting the values (dividing by the dimension raised to the degree of the fit), fitting to a Laurent polynomial, and undoing the effect of weighting to obtain an ordinary polynomial. We then assess relative errors between polynomial values vs. computed values.

The maximum percentage relative errors for these four fits, from degree 5 to 8, are 15.8, 7.2, 5.7 and 4.7 respectively. The norms of the relative errors show a similar drop between degrees 5 and 6, followed by a tapering: they are (.439, .172, .134, .124). The principal of parsimony argues in favor of degree 6 being optimal.

### 4.2 Dependence on Normal Set Size

We now describe experiments to assess complexity in terms of size of the extension field. For this purpose we chose to control coefficient growth by working in an algebraic extension of the prime field $\mathbb{Z}_{7919}$. We used straightforward linear algebra code with division-free row reduction. At each step we interleaved reductions by the extension field. In one variation we pre-process by (i) computing a primitive element, (ii) solving for all defining algebraics in as polynomials in this element, and (iii) replacing them in the matrix by their equivalent primitive element polynomials. When the linear algebra is finished and we have our polynomial, we post-process by replacing powers of the primitive element with reduced polynomials in the original extension variables. We remark that we do not compute algebraic element inverses using this method. This saves a factor of $D$ in the complexity, at the expense of obtaining a result, that is not monic. We also use the Monte Carlo probabilistic method since it makes for faster code. The goal is to show that experiments are consistent with the claimed complexity being no worse than cubic in $D$. For this purpose we removed the costly step of inversion in setting pivots to unity. By also taking advantage of the speed gain from the Monte Carlo variation, we are able to run the experiments over a fairly large range of extension degrees (recall that this speed gain applies to the matrix dimension component of the complexity and not the component due to $D$, so this does not interfere with the goal of these particular experiments).

In the first part of this experiment our algebraic extension is given by the polynomials $(3^j x^j - j, 2^j y^{j+1} - (j+1))$ where $j$ varies from 16 to 45. We use the same input matrix for all extensions. It is comprised of random integers between -5 and 5, with the $(1,1)$ element replaced by $x$ and the $(1,2)$ element replaced by $y$. The sizes $D$ of the normal sets, and corresponding timings, are in the table below.

| | | | | |
|---|---|---|---|---|
| ((272, 2.16) | (306, 2.77) | (342, 3.57) | (380, 4.24) | (420, 5.16) |
| (462, 6.40) | (506, 7.90) | (552, 9.05) | (600, 11.10) | (650, 12.79) |
| (702, 14.91) | (756, 18.01) | (812, 21.06) | (870, 26.24) | (930, 29.19) |
| (992, 36.32) | (1056, 37.65) | (1122, 41.64) | (1190, 50.50) | (1260, 63.87) |
| (1332, 64.01) | (1406, 69.76) | (1482, 77.50) | (1560, 100.1) | (1640, 109.6) |
| (1722, 119.6) | (1806, 133.9) | (1892, 139.5) | (1980, 158.9) | (2070, 191.0)) |

This fits reasonably well to a polynomial quadratic in $D$: $275.14 - 1.42894x + 0.004678x^2$. The largest relative error is under 0.1 (so less than 10%), and the norm of the vector of relative errors is 0.28. These show but little change when we use a cubic or higher degree fit. We show a log-log plot of times vs. $D$, translated to go through the origin, along with the line $y = 2x$.

The experimental complexity in this case appears to be not much larger than $O(D^2)$ (in particular, if we adjust the slope in the graph from 2 to 2.2, we get even closer alignment), and this is better than the predicted value. This is in part due to the use of an extension that has a sparse Gröbner basis. We mention also that in this experiment, cost is dominated by the Gröbner basis

computation and the creation and use of a multiplication table to convert back from a primitive element to the algebraic elements that originally define the field extension. A variant of this experiment uses the same extension, but works with a 20 digit prime modulus. The results were similar, with each data point typically around 50% slower than the corresponding one with the smaller prime modulus (Fig. 1).

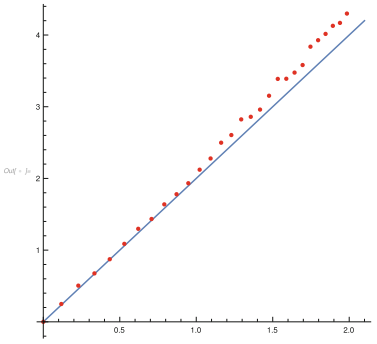


**Fig. 1.** Translated log-log plot of normal set size vs. computation times, primitive element sparse case

Our next experiment uses dense polynomials to define the extension. Leading terms in the two variables are the same as in the last experiment, but now we fill in with random coefficients times lesser power products. In this example the complexities of different steps vary considerably. The main costs (as $D$ increases) are in (i) computing the primitive element Gröbner basis, (ii) computing the table of replacements to rewrite powers of the primitive element in terms of the normal set for the basis using the original variables and (iii) performing the substitutions at the end (using this table) and expanding the result. Possibly these are due to specifics of the implementation so we note two details. We handle (i) using an implementation of the Gröbner walk [10] that has path perturbation [3]. For (ii) we do repeated reductions of successive powers of the primitive element and create a substitution table that we apply in (iii) to the minimal polynomial; this is similar to how we create and utilise a multiplication table when not working with a primitive element. A log-log plot, now using a slope of 2.5, suggests an experimental complexity of $O(D^{2.5})$. A fit indicates that this gives a smaller relative error (by a factor of 2 or so) than a quadratic.

If we ignore the cost of computing a primitive polynomial for the extension, translating the input to that form, and translating the result back to a form expressed in the original field elements, the cost can be seen to be softly linear in the degree extension $D$. We show this in Fig. 4, now plotted against a line of slope 1.1.

If we forego the use of a primitive element then the speed-limiting factors change. This must happen, since both the cost of computing a primitive element

that of converting the resulting minimal polynomial to use the original variables are removed. The new speed bump is in computing the normal set replacements for products of all pairs of terms in the normal set. This is quite fast for the case where the Gröbner basis polynomials are sparse; in the case where they are dense it becomes the bottleneck. The plot in Fig. 3 indicates a plausible complexity scaling as $O(D^2)$ (Fig. 2).

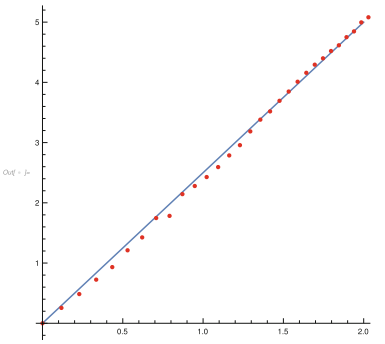


**Fig. 2.** Translated log-log plot of normal set size vs. computation times, primitive element dense case plus computing and translating to-from primitive element representation

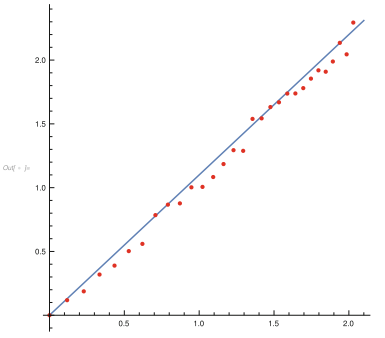


**Fig. 3.** Translated log-log plot of normal set size vs. computation times, primitive element dense case linear algebra only

Note that the plausible $O(D^2)$ complexity in no way contradicts the more conservative bounds from the theory presented in the last section, in particular Theorem 6. These examples have a fixed number of generators at $n = 2$. Thus the Minkowski polytope for the sum of normal set elements is $O(D)$ (rather than $O(D^2)$) and products of algebraic elements are likewise $O(D)$ in length. Hence we remove two factors of $D$ in the overall complexity analysis.

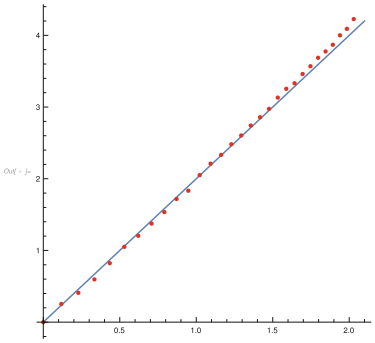


**Fig. 4.** Translated log-log plot of computation times vs. normal set size, dense case without primitive element

There is another idiosyncrasy of implementation that seems worthy of remark. Our implementation that uses a Gröbner basis for the linear algebra step has no need to compute algebraic inverses. Setting pivots to unity happens automatically as part of the Gröbner computation, as S-polynomials are formed between leading row elements and the extension-defining polynomials. On the one hand, this introduces an inefficiency as compared to straight linear algebra. On the other, it appears to vastly reduce coefficient swell in the case where our base field is the rationals. The practical trade-offs of these implementations might warrant further study, independent of the theoretical bounds presented in this work.

**Acknowledgements.** The authors would like to thank the reviewers for their many comments on our manuscript that helped us to improve it. The research of the first author was in part supported by a grant from IPM (No. 14020413).

## References

1. Adams, W.W., Loustaunau, P.: An Introduction to Gröbner Bases, vol. 3. American Mathematical Society, Providence (1994)
2. Alman, J., Williams, V.V.: A refined laser method and faster matrix multiplication. In: Proceedings of the 2021 ACM-SIAM Symposium on Discrete Algorithms (SODA), pp. 522–539. Society for Industrial and Applied Mathematics (2021)
3. Amrhein, B., Gloor, O., Küchlin, W.: On the walk. Theor. Comput. Sci. **187**, 179–202 (1997)
4. Becker, T., Weispfenning, V.: Gröbner Bases: A Computational Approach to Commutative Algebra. In cooperation with Heinz Kredel, Springer, New York (1993). https://doi.org/10.1007/978-1-4612-0913-3
5. Buchberger, B.: A criterion for detecting unnecessary reductions in the construction of Gröbner-bases. In: Ng, E.W. (ed.) EUROSAM 1979. LNCS, vol. 72, pp. 3–21. Springer, Heidelberg (1979). https://doi.org/10.1007/3-540-09519-5_52

6. Buchberger, B.: Gröbner bases: an algorithmic method in polynomial ideal theory. In: Multidimensional Systems Theory, Progress, Directions and Open Problems. Mathematics Application. vol.16, pp. 184–232. D. Reidel Publ. Co. (1985)
7. Buchberger, B.: Ein Algorithmus zum Auffinden der Basiselemente des Restklassenringes nach einem nulldimensionalen Polynomideal. Ph.D. thesis, Universität Innsbruck (1965)
8. Buchberger, B.: Bruno Buchberger's PhD thesis 1965: an algorithm for finding the basis elements of the residue class ring of a zero dimensional polynomial ideal. J. Symb. Comput. **41**(3–4), 475–511 (2006)
9. Cantor, D.G., Kaltofen, E.: On fast multiplication of polynomials over arbitrary algebras. Acta Inf. **28**(7), 693–701 (1991)
10. Collart, S., Kalkbrener, M., Mall, D., Solernó, P.: Converting bases with the Gröbner walk. J. Symb. Comput. **24**, 465–469 (1997)
11. Cox, D., Little, J., O'Shea, D.: Ideals, Varieties, and Algorithms. An Introduction to Computational Algebraic Geometry and Commutative Algebra, 3rd edn. Springer, New York (2007). https://doi.org/10.1007/978-0-387-35651-8
12. Dickenstein, A., Fitchas, N., Giusti, M., Sessa, C.: The membership problem for unmixed polynomial ideals is solvable in single exponential time. Discrete Appl. Math. **33**(1–3), 73–94 (1991)
13. Faugère, J.C., Gianni, P., Lazard, D., Mora, T.: Efficient computation of zero-dimensional Gröbner bases by change of ordering. J. Symb. Comput. **16**(4), 329–344 (1993)
14. Gebauer, R., Möller, H.M.: On an installation of Buchberger's algorithm. J. Symb. Comput. **6**(2–3), 275–286 (1988)
15. Hartshorne, R.: Algebraic Geometry. Corr. 8rd printing, vol. 52. Springer, New York (1977). https://doi.org/10.1007/978-1-4757-3849-0
16. Hashemi, A., Heintz, J., Pardo, L.M., Solernó, P.: Intrinsic complexity for constructing zero-dimensional Gröbner Bases. In: Boulier, F., England, M., Sadykov, T.M., Vorozhtsov, E.V. (eds.) CASC 2020. LNCS, vol. 12291, pp. 245–265. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-60026-6_14
17. Hashemi, A., M.-Alizadeh, B.: Computing minimal polynomial of matrices over algebraic extension fields. Bull. Math. Soc. Sci. Math. Roum. Nouv. Sér. **56**(2), 217–228 (2013)
18. van der Hoeven, J., Lecerf, G.: Accelerated tower arithmetic. J. Complexity **55**, 26 (2019). id/No 101402
19. Kreuzer, M., Robbiano, L.: Computational Commutative Algebra. II. Springer, Berlin (2005). https://doi.org/10.1007/3-540-28296-3
20. Langemyr, L.: Algorithms for a multiple algebraic extension II. In: Mattson, H.F., Mora, T., Rao, T.R.N. (eds.) AAECC 1991. LNCS, vol. 539, pp. 224–233. Springer, Heidelberg (1991). https://doi.org/10.1007/3-540-54522-0_111
21. Le Gall, F.: Powers of tensors and fast matrix multiplication. In: Proceedings of ISSAC 2014, pp. 296–303. ACM Press, New York (2014)
22. Lebreton, R.: Relaxed Hensel lifting of triangular sets. J. Symb. Comput. **68**, 230–258 (2015)
23. Li, X., Moreno Maza, M., Schost, É.: Fast arithmetic for triangular sets: from theory to practice. J. Symb. Comput. **44**(7), 891–907 (2009)
24. Lichtblau, D.: Practical computations with Gröbner bases (2009). https://www.researchgate.net/publication/260165637_Practical_computations_with_Grobner_bases
25. Lichtblau, D.: Applications of strong Gröbner bases over Euclidean domains. Int. J. Algebra **7**(5–8), 369–390 (2013)

26. Moreno Maza, M., Schost, É., Vrbik, P.: Inversion modulo zero-dimensional regular chains. In: Gerdt, V.P., Koepf, W., Mayr, E.W., Vorozhtsov, E.V. (eds.) CASC 2012. LNCS, vol. 7442, pp. 224–235. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-32973-9_19
27. Neunhöffer, M., Praeger, C.E.: Computing minimal polynomials of matrices. LMS J. Comput. Math. **11**, 252–279 (2008)
28. Noro, M.: An efficient implementation for computing Gröbner bases over algebraic number fields. In: Iglesias, A., Takayama, N. (eds.) ICMS 2006. LNCS, vol. 4151, pp. 99–109. Springer, Heidelberg (2006). https://doi.org/10.1007/11832225_9
29. Storjohann, A.: An $O(n^3)$ algorithm for the Frobenius normal form. In: Proceedings of the 1998 International Symposium on Symbolic and Algebraic Computation, ISSAC 1998, Rostock, Germany, 13–15 August 1998, pp. 101–104. ACM Press, New York (1998)