Inversion Modulo Zero-dimensional Regular Chains

Marc Moreno Maza, Éric Schost, and Paul Vrbik

Department of Computer Science, Western University
moreno@csd.uwo.ca
eschost@csd.uwo.ca
pvrbik@csd.uwo.ca

Abstract. We consider the questions of inversion modulo a regular chain in dimension zero and of matrix inversion modulo such a regular chain. We show that a well-known idea, Leverrier's algorithm, yields new results for these questions.

1 Introduction

Triangular sets, and more generally regular chains, constitute a useful data structure for encoding the solutions of algebraic systems. Among the fundamental operations used by these objects, one finds a few low-level operations, such as multiplication and division in dimension zero. Higher-level algorithms can then be built upon these subroutines: for instance, the authors of [**?**] outline a probabilistic and modular algorithm for solving zero-dimensional polynomial systems with rational coefficients. Their algorithm requires matrix inversion modulo regular chains.

Despite a growing body of work, the complexity of several basic questions remains imperfectly understood. In this article, we consider the question of inversion modulo a triangular set in dimension zero, and by extension, matrix inversion modulo such a triangular set. We show that a well-known idea, Leverrier's algorithm, surprisingly admits new results for these questions.

Triangular sets. We adopt the following convention: a *triangular set* is a family of polynomials $\mathbf{T} = (T_1, \ldots, T_n)$ in $k[X_1, \ldots, X_n]$, where k is a field. We require that for all i, $T_i \in k[X_1, \ldots, X_i]$ is monic in X_i and reduced with respect to $\langle T_1, \ldots, T_{i-1} \rangle$. Note that the slightly more general notion of a *regular chain* allows for non necessarily monic T_i; in that case, the requirement is that the leading coefficient of T_i be invertible modulo $\langle T_1, \ldots, T_{i-1} \rangle$. These regular chains may be called "zero-dimensional", since they encode finitely many points. Note that we *do not* require that the ideal $\langle \mathbf{T} \rangle$ be radical.

Multiplication modulo triangular sets. In the context of triangular sets, the first non-trivial algorithmic question is modular multiplication. For this, and for the question of inversion in the following paragraph, the input and output are polynomials reduced modulo $\langle \mathbf{T} \rangle$. We thus denote by $R_{\mathbf{T}}$ the residue class

ring $k[X_1, \ldots, X_n]/\langle T_1, \ldots, T_n \rangle$. For all $i \leq n$, let us write $d_i = \deg(T_i, X_i)$; the n-tuple $(d_1, \ldots, d_n)$ is the *multi-degree* of $\mathbf{T}$. Then, the set of monomials

$$M_{\mathbf{T}} = \left\{ X_1^{e_1} \cdots X_n^{e_n} \mid 0 \leq e_i < d_i \text{ for all } i \right\}$$

is the canonical basis of the k-vector space $R_{\mathbf{T}}$; its cardinality is the integer $\delta_{\mathbf{T}} = d_1 \cdots d_n$, which we call the *degree* of $\mathbf{T}$. In all our algorithms, elements of $R_{\mathbf{T}}$ are represented on this basis.

As of now, the best known algorithm for modular multiplication features the following running time [**?**]. For $x \geq 1$, write $\lg(x) = \log(\max(x, 2))$. Then, there exists a universal constant K such that given A, B in $R_{\mathbf{T}}$, one can compute $AB \in R_{\mathbf{T}}$ using at most $K 4^n \delta_{\mathbf{T}} \lg(\delta_{\mathbf{T}}) \lg\lg(\delta_{\mathbf{T}})$ operations in k.

Inversion modulo triangular sets. For inversion, several questions can be posed. In this paper we consider the problem: *given $A \in R_{\mathbf{T}}$, decide whether A is invertible, and if so, compute its inverse.* We are also interested in its the generalization to matrices over $R_{\mathbf{T}}$: *given a $(d \times d)$ matrix $A \in \mathscr{M}_d(R_{\mathbf{T}})$, decide whether it is invertible, and if so, compute its inverse.* We simply call this the problem of *invertibility test / inversion* in $R_{\mathbf{T}}$ (or in $\mathscr{M}_d(R_{\mathbf{T}})$).

This question should be contrasted with the following one: given $A \in R_{\mathbf{T}}$, decompose the ideal $\langle \mathbf{T} \rangle$ into a product of pairwise coprime ideals of the form $\langle \mathbf{T}_1 \rangle \cap \cdots \cap \langle \mathbf{T}_r \rangle$, all $\mathbf{T}_i$ being triangular sets, such that for all $i \leq r$, A is either a unit modulo $\langle \mathbf{T}_i \rangle$, or zero modulo $\langle \mathbf{T}_i \rangle$; we also compute the inverse of A modulo all $\langle \mathbf{T}_i \rangle$ that are among the first category. A similar, albeit more complex, question could be raised for matrices over $R_{\mathbf{T}}$. To distinguish it from the previous problem, we call this question the *quasi-inverse* computation.

When the ideal $\langle \mathbf{T} \rangle$ is maximal, so $R_{\mathbf{T}}$ is a field, the two questions are the same. Without this assumption the question of computing quasi-inverses is more complex than the inversion problem: when A is a zero-divisor modulo $\langle \mathbf{T} \rangle$, the first approach would just return "not invertible"; the second approach would actually require us to do some extra work.

As of now, most known algorithms naturally handle the second, more general problem. Indeed, the natural approach is the following: to compute an inverse in the residue class ring $R_{\mathbf{T}} = k[X_1, \ldots, X_n]/\langle T_1, \ldots, T_n \rangle$, we see it as $R_{\mathbf{T}'}[X_n]/\langle T_n \rangle$, where $\mathbf{T}'$ is the triangular set $(T_1, \ldots, T_{n-1})$ in $k[X_1, \ldots, X_{n-1}]$. Then, testing if $A \in R_{\mathbf{T}}$ is invertible, and inverting it when possible, is usually done by computing its extended GCD with T_n in $R_{\mathbf{T}'}[X_n]$, see [**?**,**?**,**?**,**?**]. This approach requires several quasi-inverse computations in $R_{\mathbf{T}'}$ (namely those of all leading terms that arise during the extended GCD algorithm). Even if A is invertible in $R_{\mathbf{T}'}$, some of these leading terms may be zero-divisors, thus we may have to decompose $\mathbf{T}$.

Main results. Our two main results concern the inversion problem, first for elements of $R_{\mathbf{T}}$, then for matrices over $R_{\mathbf{T}}$.

In what follows, in addition to $\delta_{\mathbf{T}}$, let $s_{\mathbf{T}} = \max(d_1, \ldots, d_n)$. Our theorems also involve the quantity ω, which denotes the exponent of matrix multiplication [**?**, Ch. 15]: explicitly, this means that ω is such that over any ring $\mathbb{A}$, matrices

of size d can be multiplied in d^ω operations $(+, \times)$ in $\mathbb{A}$. We take $2 < \omega \leq 3$, the best known value being $\omega \leq 2.3727$ [**?**].

Theorem 1. *There exists a constant C such that: If $1, \ldots, s_{\mathbf{T}}$ are units in k, then one can perform an invertibility test / inversion in $R_{\mathbf{T}}$ using*

$$C 4^n n\, \delta_{\mathbf{T}}\, s_{\mathbf{T}}^{(\omega-1)/2} \lg(\delta_{\mathbf{T}}) \lg\lg(\delta_{\mathbf{T}})$$

operations in k.

Dropping logarithmic factors, we see that the cost of inversion modulo $\langle \mathbf{T} \rangle$ grows like $4^n \delta_{\mathbf{T}}\, s_{\mathbf{T}}^{(\omega-1)/2}$, whereas the cost of multiplication modulo $\langle \mathbf{T} \rangle$ grows like $4^n \delta_{\mathbf{T}}$. In other words, the overhead for inversion grows like $s_{\mathbf{T}}^{(\omega-1)/2}$, which is between $s_{\mathbf{T}}^{1/2}$ and $s_{\mathbf{T}}$, depending on ω.

The second theorem describes the cost of matrix invertibility test and inversion.

Theorem 2. *There exists a constant C such that: If $1, \ldots, s_{\mathbf{T}}$ are units in k, then one can perform an invertibility test / inversion in $\mathscr{M}_d(R_{\mathbf{T}})$ using*

$$C 4^n \delta_{\mathbf{T}} \left(d^{\omega+1/2} + n s_{\mathbf{T}}^{(\omega-1)/2} \right) \lg(\delta_{\mathbf{T}}) \lg\lg(\delta_{\mathbf{T}})$$

operations in k.

Previous work. As stated above, most previous works on the invertibility question in $R_{\mathbf{T}}$ actually give algorithms for quasi-inverses, using *dynamic evaluation* techniques [**?**]. Unfortunately, managing the decompositions induced in quasi-inverse computations in an efficient manner leads to very complex algorithms: as of now, the fastest algorithm for quasi-inverse follows from [**?**,**?**], and features a running time of the form $\lambda^n \prod_{i \leq n} d_i \lg(d_i)^4 \lg\lg(d_i)$, for some non-explicit constant λ (conservative estimates give $\lambda \geq 60$).

Dynamic evaluation techniques carry over to matrix inversion, and make it possible to implement Gaussian elimination with coefficients in $R_{\mathbf{T}}$, handling decompositions of $\mathbf{T}$ when zero-divisors are met. The complexity of such a process seems quite complex to analyze; to our knowledge, this has not been done yet.

The algorithms from [**?**,**?**] apply half-GCD techniques in a recursive manner, together with fast Chinese remaindering techniques to handle splitting. We mention here another approach from [**?**]: using evaluation / interpolation techniques, the Authors extend it in [**?**] to an algorithm with cost growing like $2^n \sum_{i=1}^{n} \left(i^2 d_1 \cdots d_i d_i^{i+1} \right)$.

The main ingredient in our theorems is Leverrier's algorithm [**?**], a method for computing the characteristic polynomial of a matrix by means of the computation of the traces of its powers. Once the characteristic polynomial is known, it can be used to express the inverse of a matrix A as a polynomial in A — we still refer to this extension to inverse computation as Leverrier's algorithm, somewhat inappropriately.

This algorithm has been rediscovered, extended and improved in work by (among others) Souriau [**?**], Faddeev [**?**], Csanky [**?**], and Preparata and Sarwate [**?**]. The latter reference introduces the "baby steps / giant steps" techniques that are used herein; note on the other hand that the focus in these references is on the parallel complexity of characteristic polynomial or the inverse, which is not our main interest here.

Similar "baby steps / giant steps" techniques have been discovered in other contexts (algorithms on polynomials and power series) by Brent and Kung [**?**] and Shoup [**?**,**?**]. In these references, though, no mention was made of applications to modular inversion.

Acknowledgments. We acknowledge the support of the Canada Research Chairs Program and of NSERC.

2 Leverrier's algorithm

In this paper, we are interested in inversion algorithms which:

1. invert dense $(d \times d)$ matrices with entries in a ring $\mathbb{A}$;
2. invert elements in the $\mathbb{A}$-algebra $\mathbb{A}[X]/\langle T\rangle$, for some degree d monic polynomial T in $\mathbb{A}[X]$.

When we use these results, we take $\mathbb{A}$ of the form $R_{\mathbf{T}}$, for some triangular set $\mathbf{T}$. Our goal is to perform as little invertibility tests / inversions in $\mathbb{A}$ as possible: we thus rely on Leverrier's algorithm, which only does one. With $\mathbb{A}$ of the form $R_{\mathbf{T}}$, this allows us to avoid unnecessary splittings of $\mathbf{T}$.

Since both scenarios share many similarities, we strive to give a unified presentation, at the cost of a slight increase in notational burden.

2.1 Setup and main result

The following setup enables us to handle both cases above at once. Let $\mathbb{A}$ be our base ring and let $\mathscr{M}_d(\mathbb{A})$ be the free $\mathbb{A}$-algebra of $(d \times d)$ matrices over $\mathbb{A}$. We consider an $\mathbb{A}$-algebra $\mathbb{B}$ that is free of rank e as an $\mathbb{A}$-module, and which admits an $\mathbb{A}$-algebra embedding $\phi : \mathbb{B} \to \mathscr{M}_d(\mathbb{A})$; we assume $d \leq e$. The two above scenarios fit into this description:

1. In the first case, $\mathbb{B}$ is the whole $\mathbb{A}$-algebra $\mathscr{M}_d(\mathbb{A})$ and ϕ is the identity; here, $e = d^2$;
2. In the second case, $\mathbb{B}$ is the $\mathbb{A}$-algebra $\mathbb{A}[X]/\langle T\rangle$. It can be identified to a subalgebra of $\mathscr{M}_d(\mathbb{A})$ by means of the mapping ϕ that maps $A \in \mathbb{B} = \mathbb{A}[X]/\langle T\rangle$ to the $(d \times d)$ matrix of multiplication by A. In this case, the rank of $\mathbb{B}$ is $e = d$.

To any element $A \in \mathbb{B}$, we associate its *trace* $\mathrm{tr}(A) \in \mathbb{A}$, defined as the trace of the matrix $\phi(A) \in \mathscr{M}_d(\mathbb{A})$, and its *characteristic polynomial* $\chi_A \in \mathbb{A}[X]$, defined as the characteristic polynomial of the matrix $\phi(A)$; the latter is a monic polynomial

of degree d in $\mathbb{A}[X]$. Finally, the *determinant* $\det(A)$ of A is defined similarly, as the determinant of $\phi(A)$.

For our computations, we suppose that a basis B of the $\mathbb{A}$-module $\mathbb{B}$ is known. In both cases above, we have a canonical choice: matrices with a single non-zero entry, equal to one, in the first case, and the monomial basis $1, X, \ldots, X^{d-1}$ in the second case.

An addition in $\mathbb{B}$ then takes e operations $(+, \times)$ in $\mathbb{A}$. For multiplication, things are less straightforward: we let $\mathsf{M}(\mathbb{B})$ be such that one multiplication in $\mathbb{B}$ can be done using $\mathsf{M}(\mathbb{B})$ operations $(+, \times)$ in $\mathbb{A}$. The other black-box we need is for determining the trace: we let $\mathsf{T}(\mathbb{B})$ be such that the traces of all basis elements of B can be computed in $\mathsf{T}(\mathbb{B})$ operations $(+, \times)$ in $\mathbb{A}$. We give details below on $\mathsf{M}(\mathbb{B})$ and $\mathsf{T}(\mathbb{B})$ for our two main cases of interest.

Then, Leverrier's algorithm, combined with baby steps / giant steps techniques, yields the following result.

Proposition 1. *Suppose that $1, \ldots, d$ are units in $\mathbb{A}$. Given $A \in \mathbb{B}$, one can decide whether A is invertible, and if so compute its inverse, using*

$$\mathsf{T}(\mathbb{B}) + O\left(\sqrt{d}\,\mathsf{M}(\mathbb{B}) + d^{(\omega-1)/2}e\right)$$

operations $(+, \times)$ in $\mathbb{A}$, and one invertibility test / inversion in $\mathbb{A}$.

We will prove this result explicitly. Still, although this result may not have appeared before in this exact form, its specializations to our two cases of interest are not exactly new. As we said in the introduction, when $\mathbb{B} = \mathscr{M}_d(\mathbb{A})$, this approach is essentially Preparata and Sarwate's algorithm [**?**]. When $\mathbb{B} = \mathbb{A}[X]/\langle T\rangle$, this is in essence a combination of results of Brent and Kung [**?**] and Shoup [**?**,**?**], although these references do not explicitly discuss inverse computation, but respectively modular composition and minimal polynomial computation.

Our first case of interest is $\mathbb{B} = \mathscr{M}_d(\mathbb{A})$, with rank $e = d^2$. In this case, computing the traces of all basis elements is straightforward, so $\mathsf{T}(\mathbb{B})$ takes linear time $O(e) = O(d^2)$. Matrix multiplication takes time $\mathsf{M}(\mathbb{B}) = d^\omega$, so that we end up with a total of

$$O\left(d^{\omega+1/2}\right)$$

operations $(+, \times)$ in $\mathbb{A}$, as is well-known.

Our second case of interest is $\mathbb{B} = \mathbb{A}[X]/\langle T\rangle$, with rank $e = d$. In this case, computing the traces of all basis elements requires some work (namely, computing the Taylor series expansion of a rational function), and can be done in $O(\mathsf{M}(d))$ operations $(+, \times)$ in $\mathbb{A}$, see [**?**] — here, and in what follows, $\mathsf{M}(d)$ is a multiplication time function, such that we can multiply degree d polynomials in $\mathsf{M}(d)$ base ring operations [**?**, Ch. 9]. Multiplication in $\mathbb{B}$ takes time $O(\mathsf{M}(d))$ as well, so we end up with a total of

$$O\left(\sqrt{d}\,\mathsf{M}(d) + d^{(\omega+1)/2}\right) = O\left(d^{(\omega+1)/2}\right)$$

operations $(+, \times)$ in $\mathbb{A}$.

Other cases could be considered along these lines, such as taking $\mathbb{B}$ of the form $\mathbb{A}[X_1, X_2]/\langle T_1, T_2\rangle$, with $\langle T_1, T_2\rangle$ a triangular set of degree d, but we do not need this here.

2.2 Outline of the algorithm

In essence, Leverrier's algorithm relies on two facts: for A in $\mathbb{B}$, *(i)* the traces of the powers of A are the Newton sums of χ_A (A's characteristic polynomial) and *(ii)* Cayley-Hamilton's theorem, which says that A cancels χ_A.

Fact *(i)* above is made explicit in the following folklore lemma; see e.g. [**?**] for essentially the same arguments, in the case where $\mathbb{B} = \mathscr{M}_d(\mathbb{A})$.

Lemma 1. *Let* $\mathrm{rev}(\chi_A) = X^d \chi_A(1/X)$ *be the reverse polynomial of* χ_A*. Then the following holds in* $\mathbb{A}[[X]]$*:*

$$\frac{\mathrm{rev}(\chi_A)'}{\mathrm{rev}(\chi_A)} = -\sum_{i\geq 0} \mathrm{tr}(A^{i+1})X^i. \tag{1}$$

Proof. This equality is well-known when $\mathbb{A}$ is a field and when $\mathbb{B} = \mathscr{M}_d(\mathbb{A})$. We use this fact to prove the lemma in our slightly more general setting.

Let $\mu_{1,1}, \ldots, \mu_{d,d}$ be d^2 indeterminates over $\mathbb{Z}$, and let μ be the $(d \times d)$ matrix with entries $(\mu_{i,j})$. It is sufficient to prove we have

$$\frac{\mathrm{rev}(\chi_\mu)'}{\mathrm{rev}(\chi_\mu)} = -\sum_{i\geq 0} \mathrm{tr}(\mu^{i+1})X^i, \tag{2}$$

where $\mathrm{tr}(\mu)$, χ_μ and $\mathrm{rev}(\chi_\mu)$ are defined as previously. Indeed, starting from the equality for μ, we can deduce it for $A \in \mathbb{B}$ by applying the evaluation morphism $\mu_{i,j} \mapsto \phi(A)_{i,j}$, where $\phi(A)_{i,j}$ is the (i,j)-th entry of the matrix $\phi(A) \in \mathscr{M}_d(\mathbb{A})$.

To prove our equality for μ, we can see the variables $\mu_{i,j}$ over $\mathbb{Q}$, so that we are left to prove (2) over the field $\mathbb{L} = \mathbb{Q}(\mu_{1,1}, \ldots, \mu_{d,d})$. Since $\mathbb{L}$ is a field, it is sensible to introduce the roots $\gamma_1, \ldots, \gamma_d$ of χ_μ in $\overline{\mathbb{L}}$, which are thus the eigenvalues of μ. Then, (2) is a well-known restatement of the Newton-Girard identities (see for instance Lemma 2 in [**?**]). □

Let us write

$$\chi_A = X^d - a_1 X^{d-1} - \cdots - a_d.$$

Then, extracting coefficients in (1) shows that knowing the values $s_k = \mathrm{tr}(A^k)$, for $k = 1, \ldots, d$, enables us to obtain the coefficients a_k in a successive manner using the formula

$$a_k = \frac{1}{k}\left(s_k - \sum_{i=1}^{k-1} s_{k-i} a_i\right). \tag{3}$$

(Note our assumption that $1, \ldots, d$ are units in $\mathbb{A}$ makes this identity well-defined.) Computing all a_k in this manner takes a quadratic number of operations in $\mathbb{A}$. Using Newton iteration to solve the differential equation (1), which

essentially boils down to computing a power series exponential, one can compute $a_1, \ldots, a_d$ from $s_1, \ldots, s_d$ in $O(\mathsf{M}(d))$ operations $(+, \times)$ in $\mathbb{A}$ [**?**,**?**].

Thus, we now assume we know the characteristic polynomial χ_A of A. Fact *(ii)* above then amounts to the following. Cayley-Hamilton's theorem implies that $\chi_A(\phi(A)) = 0$ in $\mathscr{M}_d(\mathbb{A})$, and thus that $\chi_A(A) = 0$ in $\mathbb{B}$; in other words,

$$A^d - a_1 A^{d-1} - \cdots - a_{d-1} A - a_d = 0.$$

This can be rewritten as

$$A(A^{d-1} - a_1 A^{d-2} - \cdots - a_{d-1}) = a_d.$$

Thus, if $a_d = \det(A)$ is invertible in $\mathbb{A}$, A is invertible in $\mathbb{B}$, with inverse

$$A^{-1} = a_d^{-1}(A^{d-1} - a_1 A^{d-2} - \cdots - a_{d-1}); \tag{4}$$

conversely, if A is invertible in $\mathbb{B}$, $\phi(A)$ is invertible in $\mathscr{M}_d(\mathbb{A})$, and thus a_d is invertible in $\mathbb{A}$.

To summarize this outline, Leverrier's algorithm can decide if A is invertible (and if so compute its inverse) by means of the following steps:

1. compute the traces $s_1, \ldots, s_d$ of the powers of A
2. deduce χ_A using (1), using $O(\mathsf{M}(d))$ operations $(+, \times)$ in $\mathbb{A}$
3. A is invertible in $\mathbb{B}$ if and only if a_d is invertible in $\mathbb{A}$; if so, we deduce A^{-1} by means of (4).

2.3 Baby-steps / giant steps techniques

The direct implementation of Step 1 of Leverrier's algorithm consists of computing the powers $A^1, \ldots, A^d$, then taking their traces; this requires $O(d)$ multiplications in $\mathbb{B}$. Similarly, the direct approach to Step 3 by means of Horner's scheme requires $O(d)$ multiplications in $\mathbb{B}$. As is well-known, the baby steps / giant steps techniques allows for the reduction of the number of multiplications for both steps, from $O(d)$ to $O(\sqrt{d})$. We review this idea here, and analyze it in our general setup.

The dual of $\mathbb{B}$. As a preliminary, we say a few words about linear forms over $\mathbb{B}$. Let $\mathbb{B}^* = \mathrm{Hom}_{\mathbb{A}}(\mathbb{B}, \mathbb{A})$ be the dual of $\mathbb{B}$, that is, the set of $\mathbb{A}$-linear forms $\mathbb{B} \to \mathbb{A}$. For instance, the trace $\mathrm{tr} : \mathbb{B} \to \mathbb{A}$ is in $\mathbb{B}^*$.

Since we assume we have an $\mathbb{A}$-basis B of $\mathbb{B}$, it is natural to represent elements of $\mathbb{B}^*$ by means of their values on the basis B. Since we assume that $\mathbb{B}$ has rank e, its elements can be seen as column-vectors of size e, and the elements of $\mathbb{B}^*$ as row-vectors of size e. Then, applying a linear form to an element takes $O(e)$ operations $(+, \times)$ in $\mathbb{A}$.

There exists a useful operation on $\mathbb{B}^*$, the *transposed product*. The $\mathbb{A}$-module $\mathbb{B}^*$ can be turned into a $\mathbb{B}$-module: to any $A \in \mathbb{B}$, and to any $\lambda \in \mathbb{B}^*$, we can associate the linear form $A \circ \lambda : \mathbb{B} \to \mathbb{A}$ defined by $(A \circ \lambda)(B) = \lambda(AB)$. A general algorithmic theorem, the transposition principle [**?**, Th. 13.20], states that given

A and λ, one can compute the linear form $A \circ \lambda$ using $\mathsf{M}(\mathbb{B})$ operations in $\mathbb{A}$ (that is, for the same cost as multiplication in $\mathbb{B}$).

Step 1. Using transposed products, we now explain how to implement the first step of Leverrier's algorithm. As a preliminary, we "compute the trace", that is, its values on the basis B. As per our convention, this takes $\mathsf{T}(\mathbb{B})$ operations $(+, \times)$ in $\mathbb{A}$.

Let $m = \lfloor \sqrt{d} \rfloor$ and $m' = \lceil (d+1)/m \rceil$, so that both m and m' are $O(\sqrt{d})$. The baby steps / giant steps version of Step 1 first computes the sequence of "baby steps"

$$M_0, M_1, M_2, \ldots, M_m = A^0, A^1, A^2, \ldots, A^m,$$

by means of repeated multiplications by A. Then, by repeated transposed multiplications by M_m, we compute the "giant steps" (which are here linear forms)

$$\lambda_0, \lambda_1, \lambda_2, \ldots, \lambda_{m'} = \mathrm{tr},\ M_m \circ \mathrm{tr},\ M_m^2 \circ \mathrm{tr},\ \ldots,\ M_m^{m'} \circ \mathrm{tr}.$$

Computing all M_i and λ_j takes $O(\sqrt{d})$ multiplications and transposed multiplications in $\mathbb{B}$, for a total of $O(\sqrt{d}\,\mathsf{M}(\mathbb{B}))$ operations $(+, \times)$ in $\mathbb{A}$.

Knowing the M_i and λ_j, we can compute the required traces as $\lambda_j(M_i)$, for $0 \leq i < m$ and $0 \leq j < m'$, since they are given by

$$\lambda_j(M_i) = \mathrm{tr}(M_i M_m^j) = \mathrm{tr}(A^i A^{mj}) = \mathrm{tr}(A^{i+mj}),$$

and the exponent $i + mj$ cover all of $0, \ldots, d$. As we saw above, computing each $\lambda_j(M_i)$ amounts to doing a dot-product in size e, so a direct approach would give a cost of $O(de)$ operations in $\mathbb{A}$.

Better can be done, though. Consider the $(e \times m)$ matrix Γ whose columns give the coefficients of $M_0, \ldots, M_{m-1}$ on the basis B, and the $(m' \times e)$ matrix Λ whose rows give the coefficients of $\lambda_0, \ldots, \lambda_{m'-1}$ on the dual basis of B. Then, the (j, i)-th entry of $\Lambda\Gamma$ is precisely the value $\lambda_j(M_i)$. Since m and m' are both equivalent to $\sqrt{d}$, a naive matrix multiplication algorithm computes the product $\Lambda\Gamma$ in $O(de)$ operations in $\mathbb{A}$, as above. However, by doing a block product, with $O(e/\sqrt{d})$ blocks of size $O(\sqrt{d})$, we obtain $\Lambda\Gamma$ using $O(d^{(\omega-1)/2}e)$ operations $(+, \times)$ in $\mathbb{A}$.

Step 3. In order to perform Step 3, we have to evaluate $A^{d-1} - a_1 A^{d-2} - \cdots - a_{d-1}\mathbf{I}$, then divide by a_d if possible. Let us write $a_0 = -1$, and define $\alpha_i = -a_{d-1-i}$ for $i = 0, \ldots, d-1$; then, the quantity to compute is

$$p(A) = \sum_{i=0}^{d-1} \alpha_i A^i.$$

We extend the sum, by adding dummy coefficients α_i set to zero, to write

$$p(A) = \sum_{i=0}^{mm'-1} \alpha_i A^i;$$

this is valid, since by construction $mm' - 1 \geq d$. For $k \geq 0$, let us then define

$$\sigma_k = \sum_{i=km}^{(k+1)m-1} \alpha_i M_{i-km} = \sum_{i=0}^{m-1} \alpha_{i+km} M_i;$$

then, we see that we have

$$p(A) = (\cdots(\sigma_{m'-1} M_m + \sigma_{m'-2})\, M_m + \cdots)\, M_m + \sigma_0. \tag{5}$$

Using this formula, the algorithm to compute $p(A)$ first requires the computation of all M_i, for $i = 0, \ldots, m$, using $O(\sqrt{d}\,\mathsf{M}(\mathbb{B}))$ operations $(+, \times)$ in $\mathbb{A}$.

Next, we have to compute $\sigma_0, \ldots, \sigma_{m'-1}$. As for Step 1, let Γ denote the $(e \times m)$ matrix whose columns give the coefficients of $A^0 = M_0, \ldots, A^{m-1} = M_m$. Then, σ_k is obtained by right-multiplying the matrix Γ by the size m column vector $[\alpha_{km} \cdots \alpha_{(k+1)m-1}]^t$. Joining all these column vectors in a $(m \times m')$ matrix Δ, we obtain all σ_k by computing the product $\Gamma\Delta$. As for Step 1, the cost is $O(d^{(\omega-1)/2}e)$ operations $(+, \times)$ in $\mathbb{A}$.

Finally, once all σ_k are known, we obtain $p(A)$ by means of m' products and additions in $\mathbb{B}$; the cost is $O(\sqrt{d}\,\mathsf{M}(\mathbb{B}))$ operations $(+, \times)$ in $\mathbb{A}$. Putting all costs seen before together, we obtain the cost announced in Proposition 1.

3 Proof of the main theorems

Using Proposition 1, it becomes straightforward to prove Theorems 1 and 2. Let $\mathbf{T} = (T_1, \ldots, T_n)$ be a triangular set of multidegree $(d_1, \ldots, d_n)$ in $k[X_1, \ldots, X_n]$. First, we deal with invertibility test and inversion in $R_{\mathbf{T}}$, assuming that all integers from 1 to $s_{\mathbf{T}} = \max(d_1, \ldots, d_n)$ are units in k.

Let A be in $R_{\mathbf{T}}$. As in the introduction, we view $R_{\mathbf{T}}$ as $R_{\mathbf{T}'}[X_n]/\langle T_n \rangle$, where $\mathbf{T}'$ is the triangular set $(T_1, \ldots, T_{n-1})$ in $k[X_1, \ldots, X_{n-1}]$. Applying Proposition 1, and referring to the discussion just after it, we see that we can decide whether A is invertible in $R_{\mathbf{T}}$, and if so compute its inverse, using

1. $O\left(d_n^{(\omega+1)/2}\right)$ operations $(+, \times)$ in $R_{\mathbf{T}'}$; and
2. one invertibility test / inversion in $R_{\mathbf{T}'}$.

As recalled in the introduction, multiplications in $R_{\mathbf{T}'}$ can be done for the cost of $K 4^{n-1} \delta_{\mathbf{T}'} \lg(\delta_{\mathbf{T}'}) \lg\lg(\delta_{\mathbf{T}'})$ operations in k, for some constant K. The same holds for additions in $R_{\mathbf{T}'}$, since additions can be done in optimal time $\delta_{\mathbf{T}'}$. Let K' be a constant such that the big-Oh estimate in the first item above is bounded by $K' d_n^{(\omega+1)/2}$.

Notice $\delta_{\mathbf{T}'} = d_1 \cdots d_{n-1}$, and that it admits the obvious upper bound: $\delta_{\mathbf{T}'} \leq \delta_{\mathbf{T}}$. Then, the total running time $\mathsf{I}(d_1, \ldots, d_n)$ of the invertibility test / inversion algorithm follows the recurrence

$$\mathsf{I}(d_1, \ldots, d_n) \leq K K' 4^{n-1} d_1 \cdots d_{n-1} d_n^{(\omega+1)/2} \lg(\delta_{\mathbf{T}}) \lg\lg(\delta_{\mathbf{T}}) + \mathsf{I}(d_1, \ldots, d_{n-1}),$$

which can be simplified as

$$\mathsf{I}(d_1, \ldots, d_n) \leq C4^n \delta_{\mathbf{T}} d_n^{(\omega-1)/2} \lg(\delta_{\mathbf{T}}) \lg\lg(\delta_{\mathbf{T}}) + \mathsf{I}(d_1, \ldots, d_{n-1}),$$

with $C = KK'/4$. Unrolling the recurrence, we obtain

$$\mathsf{I}(d_1, \ldots, d_n) \leq C4^n \delta_{\mathbf{T}} \left(d_1^{(\omega-1)/2} + \cdots + d_n^{(\omega-1)/2} \right) \lg(\delta_{\mathbf{T}}) \lg\lg(\delta_{\mathbf{T}}).$$

With $s_{\mathbf{T}} = \max(d_1, \cdots, d_n)$, this admits the upper bound

$$\mathsf{I}(d_1, \ldots, d_n) \leq C4^n n\, \delta_{\mathbf{T}}\, s_{\mathbf{T}}^{(\omega-1)/2} \lg(\delta_{\mathbf{T}}) \lg\lg(\delta_{\mathbf{T}}),$$

which proves Theorem 1.

Theorem 2 then follows from the combination of Proposition 1 and Theorem 1. To invert a $(d \times d)$ matrix A with entries in $R_{\mathbf{T}}$, we apply Leverrier's algorithm in Proposition 1, over the ring $\mathbb{A} = R_{\mathbf{T}}$. As explained after Proposition 1, the cost is $O(d^{\omega+1/2})$ operations $(+, \times)$ in $R_{\mathbf{T}}$, followed by the invertibility test / inversion of the determinant of A in $R_{\mathbf{T}}$. The cost reported in Theorem 2 then follows easily from the bounds on the cost of multiplication and invertibility test in $R_{\mathbf{T}}$.

4 Experimental Results

In this section, we compare Maple implementations of two approaches: our own recursive Leverrier algorithm and the existing (Gauss-Bareiss based) method from the `RegularChains` Maple library [**?**]. Our implementation uses the `RegularChains` library for normal forms, multiplication, etc, so we believe that this is a fair comparison.

Letting $p = 962592769$, we choose a random dense regular chain $\mathbf{T}$ in $\mathbb{F}_p[X_1, \ldots, X_n]$, with varying n, with and multidegree $(d, \ldots, d)$ for some varying d. We invert a random (and thus invertible) $m \times m$ matrix A with random entries in $R_{\mathbf{T}}$. We compare our results to the `MatrixInverse` function from `RegularChains`.

Table 4 gives the results of our experiments on a AMD Athlon running Linux, using Maple 15. For our algorithm, we detail the timings for trace computation (Step 1 of the algorithm), reconstituting the characteristic polynomial χ_A (Step 2), the inverse of the determinant of A, and the computation of the inverse of A itself (Step 3). As was to be expected, Step 1 and Step 3 take comparable times. For $n = 1$, our algorithm behaves very similarly to the built-in `MatrixInverse`. Already for $n = 3$, our implementation usually gives better results.

				Leverrier					**MatrixInverse**
n	d	$\delta_{\mathbf{T}}$	m	Traces	CharPoly	Inverse	Horner	Total	Time
1	2	2	3	0.03	0.00	0.00	0.01	0.04	0.2
1	2	2	6	0.03	0.00	0.00	0.02	0.05	0.07
1	2	2	9	0.07	0.00	0.00	0.06	0.13	0.15
1	2	2	12	0.19	0.00	0.00	0.12	0.31	0.34
1	2	2	15	0.26	0.00	0.00	0.23	0.49	0.54
1	2	2	18	0.47	0.00	0.00	0.45	0.92	0.72
1	10	10	3	0.02	0.00	0.00	0.01	0.03	0.12
1	10	10	6	0.10	0.01	0.00	0.09	0.20	0.39
1	10	10	9	0.43	0.01	0.00	0.21	0.65	1.09
1	10	10	12	0.96	0.01	0.00	0.63	1.60	2.26
1	10	10	15	1.67	0.02	0.00	1.29	2.98	4.09
1	10	10	18	3.17	0.02	0.00	2.09	5.28	6.67
1	18	18	3	0.02	0.01	0.00	0.03	0.06	0.22
1	18	18	6	0.33	0.01	0.00	0.20	0.54	0.87
1	18	18	9	0.93	0.02	0.00	0.50	1.45	2.28
1	18	18	12	2.30	0.02	0.00	1.51	3.83	4.60
1	18	18	15	4.22	0.03	0.00	3.36	7.61	8.02
1	18	18	18	8.07	0.05	0.00	5.43	13.56	13.14
3	3	27	3	0.14	0.02	0.08	0.22	0.46	7.7
3	3	27	6	1.75	0.07	0.07	1.46	3.35	10.4
3	3	27	9	5.68	0.11	0.08	3.58	9.45	15.5
3	3	27	12	13.47	0.16	0.07	9.18	22.8	24
3	3	27	15	22.9	0.27	0.08	19.4	42.8	35.7
3	3	27	18	42.67	0.27	0.07	30	73	52.2
3	4	64	3	0.88	0.22	0.58	1.6	3.28	54.5
3	4	64	6	10.6	0.43	0.63	9.80	21.4	100
3	4	64	9	32.8	0.77	0.62	22.5	56.7	184
3	4	64	12	74.9	1.07	0.63	55.1	132	324
3	4	64	15	121	1.38	0.65	111	233	524
3	4	64	18	213	1.67	0.58	163	379	840
3	5	125	3	0.75	0.08	0.63	0.74	2.20	159
3	5	125	6	7.07	0.22	0.63	5.07	14	299
3	5	125	9	22.5	0.38	0.55	12.6	36.0	548
3	5	125	12	53.7	0.65	0.54	33.2	88.1	960
3	5	125	15	94.1	0.84	0.54	72.1	167	1582
3	5	125	18	175.08	1.08	0.57	112	288	2462

Table 1. Experimental results (in seconds).