High-order lifting and integrality certification

Arne Storjohann

School of Computer Science, University of Waterloo, Waterloo, Ontario, Canada, N2L 3G1

Abstract

Reductions to polynomial matrix multiplication are given for some classical problems involving a nonsingular input matrix over the ring of univariate polynomials with coefficients from a field. High-order lifting is used to compute the determinant, the Smith form, and a rational system solution with about the same number of field operations as required to multiply together two matrices having the same dimension and degree as the input matrix. Integrality certification is used to verify correctness of the output. The algorithms are space efficient.

1 Introduction

The interaction between matrix multiplication and linear algebra problems on matrices over a field K is well understood. The best known algorithms for computing the determinant of a nonsingular matrix $A \in K^{n \times n}$, or for solving a linear system of equations involving A, have cost $O(n^\theta)$ field operations, $2 < \theta \leq 3$ a valid exponent for matrix multiplication. This paper gives similar results for problems on polynomial matrices. We show that a wide variety of problems involving a nonsingular matrix $A \in K[x]^{n \times n}$ can be solved with $O(n^\theta d) \times (\log n + \log d)^{O(1)}$ field operations, d a bound on the degree of A.

Of the problems we consider the most fundamental is linear system solving. Let a vector $b \in K[x]^{n \times 1}$ be given in addition to A. The nonsingular rational system solving problem is to compute the vector $A^{-1}b \in K(x)^{n \times 1}$. Numerators and denominators of entries in $A^{-1}b$ will have degree bounded by nd, where d is a bound on the degree of entries in A and b. The most efficient algorithms for computing $A^{-1}b$ work by computing a truncated X-adic series expansion of $A^{-1}b$ using Hensel lifting, or Newton iteration, and then applying rational function reconstruction. The descriptions in Moenck and Carter (1979)

Email address: astorjoh@scg.uwaterloo.ca (Arne Storjohann).

and Dixon (1982) are for integer matrices but carry over to the case $K[x]$ immediately. The method usually requires knowing a small degree $X \in K[x]$ such that X is relatively prime to $\det A$ (Notation: $X \perp \det A$). The technique has been well studied. Mulders and Storjohann (2000) give a variation that always allows choosing X to be a power of x and is designed to handle efficiently input systems of arbitrary shape and rank. Given a system with n rows, m columns, rank r, and degrees of entries bounded by d, the algorithm either computes a rational solution or proves that the system is inconsistent with $O((n+m)r^2 d^{1+\epsilon})$ field operations from K, $0 < \epsilon \leq 1$ depending on the cost of polynomial multiplication. Thus, the algorithm solves the nonsingular rational system solving problem deterministically with $O(n^3 d^{1+\epsilon})$ field operations. In this paper we reduce the exponent of n from three down to θ, $2 < \theta \leq 3$ a valid exponent for matrix multiplication. Given an $X \in K[x]$ such that $X \perp \det A$, our algorithm computes $A^{-1}b$ with $O(n^\theta (\log n) d^{1+\epsilon})$ field operations, d a bound for $\deg A$, $\deg b$, and $\deg X$. We also give an extension of the algorithm that allows entries in b to have degree substantially larger than those in A without adversely affecting the cost estimate. It suffices that $d \geq (\deg b)/n$ as well as $d \geq \deg A, \deg X$.

The second problem we consider is integrality certification. Let a matrix $B \in K[x]^{n \times m}$ be given in addition to A. The integrality certification problem is to answer the following question: can every column of B be expressed as a $K[x]$–linear combination of columns of A? This question is equivalent to the following: is $A^{-1}B$ over $K[x]$? Given an $X \in K[x]$ such that $X \perp \det A$, our algorithm answers this question with $O(n^\theta (\log n) d^{1+\epsilon})$ field operations, d a bound for $\deg A$ and $\deg X$. This cost estimate holds for any B such that $m(1 + (\deg B)/d)$ is $O(n)$. A special case of the integrality certification problem occurs when B is equal to the identity matrix. The question then becomes: is A a unimodular matrix, that is, is the inverse of A over $K[x]$? Since A is unimodular precisely when $\det A$ has degree zero, we can test for unimodularity by computing $\det A \bmod X$ for a small degree, and randomly chosen X. This gives a nearly optimal Monte Carlo probabilistic algorithm with running time about $O(n^\theta + n^2 d)$ field operations (ignoring logarithmic factors). Here we give a deterministic algorithm for solving this special case of the integrality certification problem that has cost $O(n^\theta (\log n) d^{1+\epsilon})$ field operations.

The third problem we consider is determinant computation. Mulders and Storjohann (2002) show how to compute $\det A$ deterministically with $O(n^3 d^2)$ field operations, $d = \deg A$. The Las Vegas probabilistic algorithm we give here uses an expected number of $O(n^\theta (\log n)^2 d^{1+\epsilon})$ field operations. For fields of small cardinality the cost estimate increases by a poly-logarithmic factor. In the same time the Smith form of A is computed, also Las Vegas. Recall that Smith form of A is the unique diagonal matrix $S = \text{Diagonal}(s_1, s_2, \ldots, s_n)$

such that $s_i | s_{i+1}$ for $1 \leq i < n$, and $S = UAV$ for unimodular matrices $U, V \in K[x]^{n \times n}$.

We mention some recent related work. Giorgi et al. (2003) give algorithms with cost $O(n^\theta d) \times (\log n + \log d)^{O(1)}$ field operations for computing minimal bases and order d matrix approximates, and for computing a column reduced form of an invertible matrix. Giorgi et al. (2003) also consider some reductions in the opposite direction. They show that if there is a straight-line program of length $D(n, d)$ for computing the coefficient of degree d of the determinant, then there is a straight-line program of length no more than $8D(n, d)$ which multiplies two matrices of degree d.

2 Model of computation and cost functions

We analyse our algorithms by bounding the number of required field operations from K on an algebraic random access machine; the operations $+$, $-$, $\times$ and "divide by a nonzero" are considered as unit step operations.

Polynomial multiplication

We use M for polynomial multiplication. Let $\mathsf{M} : \mathbb{Z}_{\geq 0} \longrightarrow \mathbb{R}_{>0}$ be such that polynomials in $K[x]$ of degree bounded by d can be multiplied using at most $\mathsf{M}(d)$ field operations. The classical method has $\mathsf{M}(d) = 2d^2$. The algorithm of Karatsuba and Ofman (1963) allows $\mathsf{M}(d) = O(d^{1.59})$. FFT-based methods allow $\mathsf{M}(d) = O(d(\log d)(\log\log d))$.We refer to (Gathen and Gerhard, 1999, Section 11.1) for more details and references. We assume that $\mathsf{M}(ab) \leq \mathsf{M}(a)\mathsf{M}(b)$ for $a, b \in \mathbb{Z}_{>1}$.

It will be useful to define an additional function B for polynomial gcd–related computations. We assume that $\mathsf{B}(d) = \mathsf{M}(d) \log d$ or $\mathsf{B}(d) = d^2$. Then the extended gcd problem with two polynomials in $K[x]$ of degree bounded by d can be solved with $O(\mathsf{B}(d))$ field operations.

Matrix multiplication

We use MM for matrix multiplication. Let $\mathsf{MM} : \mathbb{Z}_{>0} \longrightarrow \mathbb{R}_{>0}$ be such that two $n \times n$ matrices over a ring (commutative, with 1) can be multiplied with $\mathsf{MM}(n)$ ring operations. The classical method has $\mathsf{MM}(n) = 2n^3 - n^2$. The

algorithm of Strassen (1973) allows $\mathsf{MM}(n) = 42n^{\log_2 7}$. The asymptotically fastest known method allows $\mathsf{MM}(n) = O(n^{2.376})$.

We use MM with two arguments for polynomial matrix multiplication. Let $\mathsf{MM} : \mathbb{Z}_{>0} \times \mathbb{Z}_{\geq 0} \longrightarrow \mathbb{R}_{>0}$ be such that two matrices from $K[x]^{n \times n}$ with degree bounded by d can be multiplied together with at most $\mathsf{MM}(n, d)$ field operations. We can always choose $\mathsf{MM}(n, d) = O(\mathsf{MM}(n)\mathsf{M}(d))$, but better bounds may be possible. For example, if $\#K > 2d$ then we can use an evaluation/interpolation scheme to get $\mathsf{MM}(n, d) = O(\mathsf{MM}(n)d + n^2\mathsf{B}(d))$.

In our algorithms, every time we multiply two polynomial matrices we will need to perform some additional work also, e.g., reduce all entries in the product modulo a given $X \in K[x]$, $\deg X \leq d$. For this reason, we are going to assume that $n^2\mathsf{M}(d) = O(\mathsf{MM}(n, d))$. This is a mild assumption, since an information lower bound gives $n^2(2d + 1) = O(\mathsf{MM}(n, d))$.

Some results will be greatly simplified by making the explicit assumption that $\mathsf{B}(n) = O(\mathsf{MM}(n)/n)$, which stipulates that if fast matrix multiplication techniques are used then fast polynomial multiplication should be used also. For example, $\mathsf{B}(n) = O(\mathsf{MM}(n)/n)$, then $n\,\mathsf{B}(nd) = O(\mathsf{MM}(n)\mathsf{B}(d))$.

Reduction to matrix multiplication

We use $\overline{\mathsf{MM}}$ for some problems (see below) that can be reduced recursively to matrix multiplication. For n a power of two, define

$$\overline{\mathsf{MM}}(n, d) := \left(\sum_{i=0}^{\log_2 n} 4^i \mathsf{MM}(2^{-i}n, d) \right) + n^2(\log n)\mathsf{B}(d). \tag{1}$$

If n is not a power of two, then define $\mathsf{MM}(n, d) := \mathsf{MM}(\bar{n}, d)$, where $\bar{n}$ is the smallest power of two greater than n. We now motivate the definition of $\overline{\mathsf{MM}}$.

Suppose $X \in K[x]$ is nonzero. Then $R := K[x]/(X)$ is a principal ideal ring. R can be taken to be the set of all polynomials in $K[x]$ with degree strictly less than d, $d = \deg X$. Multiplication in R costs $O(\mathsf{M}(d))$ field operations and is accomplished by first multiplying over $K[x]$ and then reducing modulo X. Similarly, matrices in $R^{n \times n}$ can be multiplied with $\mathsf{MM}(n, d)$ field operations. The following operations can be accomplished with $O(\overline{\mathsf{MM}}(n, d))$ field operations:

- Compute a unimodular $U \in R^{n \times n}$ such that UA is upper triangular.
- Compute the inverse of an $A \in R^{n \times n}$ or determine that A is not invertible.
- Compute the Smith canonical form of an $A \in R^{n \times n}$.

An algorithm supporting the running time $O(\overline{\mathsf{MM}}(n,d))$ field operations for the first problem is given by Hafner and McCurley (1991). Now consider the second problem. A will be invertible precisely if all diagonal entries of UA are invertible. If so, the inverse of UA can be found using an additional $O(\mathsf{MM}(n,d) + n\,\mathsf{B}(d))$ field operations: first multiply UA by the diagonal matrix D such that diagonal entries in DUA are equal to one, then apply a standard recipe for triangular matrix inversion. The result for computing the Smith form is given in (Storjohann, 2000, Chapter 7).

If there exists an absolute constant $\gamma > 0$ such that $n^{2+\gamma} = O(\mathsf{MM}(n))$, then we can choose $\overline{\mathsf{MM}}(n,d) = O(\mathsf{MM}(n)\mathsf{B}(d))$.

3 Outline and synopsis

Let K be a field and $A \in K[x]^{n\times n}$ be nonsingular. Let $B \in K[x]^{n\times m}$ be given in addition to A. For any $X \in K[x]$ such that $X \perp \det A$, the matrix of rational functions $A^{-1}B \in K(x)^{n\times m}$ admits a unique, and possibly infinite, X-adic series expansion:

$$A^{-1}B = C_0 + C_1X + C_2X^2 + C_3X^3 + \cdots, \tag{2}$$

where each $C_* \in K[x]^{n\times m}$ has $\deg C_* < \deg X$. The first part of this paper (sections 4—10) presents fast algorithms for computing only parts of the expansion. We call this high-order lifting. There are different variations of high-order lifting. One variation calls for computing a single contiguous segment $[C_h, C_{h+1}, \ldots, C_{h+k-1}]$ of coefficients for a given h and k. Another variation computes a collection of such segments for a given expansion. This section gives intuitive descriptions of the key ideas and algorithms for the various versions of X-adic lifting.

Nonsingular rational system solving using X-adic lifting

Sections 4 and 5 define some notation and recall some basic facts about X-adic expansions of rational functions, including the recovery of such expansions using X-adic lifting.

Consider the problem of computing the X-adic expansion of

$$A^{-1}b = c_0 + c_1X + c_2X^2 + c_3X^3 + \cdots$$

where b is a column vector, and both $\deg A$ and $\deg b$ are $\leq \deg X$. Suppose

our goal is to compute the expansion up to order X^k, k even. We can divide the problem into two similar subproblems. The first is to compute the expansion of $A^{-1}b$ up to order $X^{k/2}$.

$$A^{-1}b \equiv c_0 + c_1 X + \cdots + c_{k/2-1} X^{k/2-1} \bmod X^{k/2}. \tag{3}$$

Multiplying both sides by A and then subtracting the right hand side from the left gives

$$b - A(c_0 + c_1 X + \cdots + c_{k/2-1} X^{k/2-1}) \equiv 0 \bmod X^{k/2}.$$

The left hand side must be divisible by $X^{k/2}$. Set

$$r_{k/2} = (b - A(c_0 + c_1 X + \cdots + c_{k/2-1} X^{k/2-1}))/X^{k/2}. \tag{4}$$

The degree bounds for b and A imply that $\deg r_{k/2} < d$. The key idea of X-adic lifting is to replace of the "mod" in (3) with the "residue" term $r_{k/2}$. Multiply both sides of (4) by $A^{-1}X^{k/2}$, and rearrange to obtain the following:

$$A^{-1}b = \overbrace{c_0 + c_1 X + \cdots + c_{k/2-1} X^{k/2-1}}^{A^{-1}b \bmod X^{k/2}} + A^{-1} r_{k/2} X^{k/2}. \tag{5}$$

Thus, the second subproblem — compute the expansion of $A^{-1}r_{k/2}$ up to order $X^{k/2}$ — has the same form as the first subproblem. The salient point is that we need to solve the first subproblem before we can begin the second subproblem. High–order lifting will be used to compute $r_{k/2}$ directly, allowing us to incorporate recursion into the computation.

High–order components of matrix inverse

Section 6 gives our first high-order lifting algorithm. Consider (2) when $B = I_n$ and $\deg A \leq \deg X$. Let $\circ$ denote the coefficients of the X-adic expansion of A^{-1}, ordered from left to right. Let $\bullet$ denote a coefficient that has currently been computed. Normally, all coefficients of the expansion are computed up to order $X^{\Theta(n)}$ — in terms of n this costs $O(n^\theta \times n)$ field operations using $O(\log n)$ steps of quadratic X-adic lifting, cf. Figure 1. After the fourth step of lifting (which dominates the cost) all initial thirty-two coefficients have been computed. The algorithm we give here computes a critical subset of size $\Theta(\log n)$ from the first $\Theta(n)$ coefficients by using quadratic X-adic lifting combined with short products, cf. Figure 2. The result is that a $\Theta(n)$ factor in the running time is replaced by $\Theta(\log n)$. Although most of the coefficients of

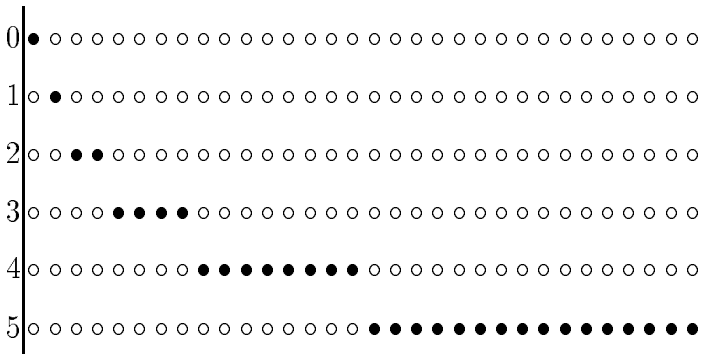
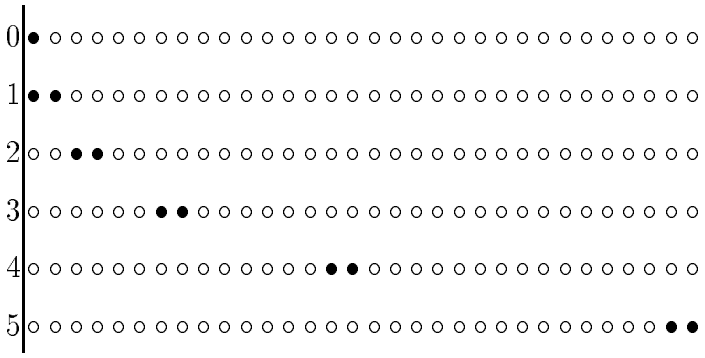


Fig. 1. Quadratic lifting for $n = 5$



(6)

Fig. 2. High-order component lifting for $n = 5$

the inverse expansion are not recovered, the computation of the critical subset of high-order components has many applications. The algorithm described in this section is used in almost all subsequent sections.

Unimodularity certification

Section 7 gives an algorithm to test for unimodularity. Recall that a matrix $A \in K[x]^{n\times n}$ is unimodular precisely when the determinant of A is a nonzero constant polynomial. Another characterization of unimodularity is that the x-adic expansion of A^{-1} exists (i.e., $x \perp \det A$), and is finite.

Suppose that A has $\deg A \leq d$ and $x \perp \det A$. Let $X = x^d$. Let $k > n$ and consider the (possibly infinite) X-adic expansion

$$A^{-1} = \overbrace{C_0 + C_1 X + \cdots + C_{k-2}X^{k-2}}^{C} + C_{k-1}X^{k-1} + C_k X^k + \cdots .$$

If A is unimodular, then $A^{-1} \in K[x]^{n\times n}$ and we have the classical *a priori* bound $\deg A^{-1} \leq (n-1)d$, i.e., A is unimodular precisely if all coefficients

C_i are zero for $i \geq k-1$. Thus, if C_{k-1} is not the zero matrix, then A is not unimodular. A key point to note here is that the particular value of k is not important. We only require that $k > n$. In our case we will choose k to be the smallest possible power of two. Then we can compute C_{k-1} using $O(\log n)$ steps of high–order lifting.

Now suppose that C_{k-1} is the zero matrix. It is not immediately clear that this should imply that all of C_i are zero for $i \geq k-1$, but in Section 7 we show that this is in fact the case.

Thus, we can test if A is unimodular by determining if a single high order coefficient C_{k-1} of A^{-1} is the zero matrix.

Series solution — small degree right hand side

Section 8 gives an algorithm for rational system solving in the case where $\deg b \leq \deg A$. The main idea is to reduce the problem of solving one system up to order X^k to that of solving two systems up to order $X^{k/2}$. We have described such a reduction above, cf. (5). The key difference here is that we compute the residue term $r_{k/2}$ shown in (4) without first solving the initial subproblem shown in (3). In Section 5 we observe that $r_{k/2}$ can be computed using a single matrix×vector involving A and a particular high-order component of the inverse of A. We now have

$$A^{-1}b \bmod X^k = \left(A^{-1}\left[b\middle|r_{k/2}\right] \bmod X^{k/2}\right)\left[\frac{1}{X^{k/2}}\right] \tag{7}$$

where the right hand side $\left[b\middle|r_{k/2}\right]$ has column dimension two, cf. (5). This idea is applied recursively $O(\log k)$ times, each time doubling the column dimension of the right hand side. This allows matrix multiplication to be introduced into the rational system solving problem, effectively reducing the overall complexity in terms of n from $O(n^3)$ to $O(n^\theta \times \log n)$.

Series solution

Section 9 extends the result of the previous section to allow $\deg b = O(n \deg A)$ without increasing the asymptotic cost. Let $d = \deg X$, and consider the case when the right hand side b has degrees bounded by nd, say $b = b_0 + b_1 X + b_2 X^2 + \cdots + b_{n-1}X^{n-1}$. Suppose our goal is to produce $A^{-1}b$ up to order X^n.

Solving this single linear system with large degree right hand side is equivalent to solving n systems with small degree right hand side:

$$A^{-1}b \bmod X^n = \left(\sum_{i=0}^{n-1} (A^{-1}b_i \bmod X^n) X^i \right) \bmod X^n.$$

The algorithm encodes the "fat" vector b as an $n \times n$ matrix B with ith column equal to b_{i-1} and then uses the small degree right hand side series solution method. The ith column of B may be thought to be implicitly multiplied by X^{i-1}. For an $n \times n$ matrix C, a matrix$\times$vector product Cb, $\deg b < nd$, can be now accomplished more efficiently as a matrix$\times$matrix product CB, $\deg B < d$.

Suppose n is even. Let B_i denote the ith column of B. Using $\Theta(1)$ matrix products involving A, B, and the high-order components of the expansion of A^{-1}, the algorithm produces a second matrix R such that

$$\begin{aligned} A^{-1}b \bmod X^n &= \left(\overbrace{\sum_{i=0}^{n-1} (A^{-1}B_i \bmod X^{n/2}) X^i}^{\text{subproblem 1}} + \overbrace{\sum_{i=0}^{n-1} (A^{-1}R_i \bmod X^{n/2}) X^i}^{\text{subproblem 2}} \right) \bmod X^n \\ &= \left(\sum_{i=0}^{n-1} (A^{-1}(B_i + R_i) \bmod X^{n/2}) X^i \right) \bmod X^n. \end{aligned}$$

Thus, a single matrix addition $B + R$ allows us to recurse on only one instead of two problems. Now suppose n is a power of two. Then this technique can be applied for order $X^{n/2}, X^{n/4}, X^{n/8}, \cdots$ yielding a series of $O(\log n)$ transformations using the high-order components of the expansion of A^{-1}. The overall cost in terms of n is $O(n^\theta \times \log n)$.

High-order lifting

Section 10 gives a general algorithm for solving the high-order lifting problem: the recovery, for some h and k, of a contiguous segment of coefficients $H = C_h + C_{h+1}X + C_{h+2}X^2 + \cdots + C_{h+k-1}X^{k-1}$ from the X-adic expansion of $A^{-1}B$ as shown in (2). By general we mean that the column dimension as well as degrees of entries in B are not restricted. The algorithm here is a straightforward combination of the algorithms of previous sections. The key point is the analysis. Let $\deg A \leq d$, $d = \deg X$, and m be the column dimension of B. A running time of $O(n^\theta \times \log n)$ in terms of n is achieved for a wide range of the input parameters m, k and $\deg B$. All that is required is that the

parameters m and $\{(\deg B)/d, k\}$ be balanced: both $m \times (\deg B)/d$ and $m \times k$ should be $O(n)$.

Integrality certification

Many of the techniques we develop in this paper for polynomial matrices are applicable to the integer matrix setting. It will be convenient to give some examples using integers.

There is a natural analogy between X-adic expansions of polynomials and p-adic expansions of integers, e.g.,

$$2691 = 1 + 9(10) + 6(10^2) + 2(10^3).$$

For brevity, we will prefer to use the standard representation on the left.

Fractions also admit 10-adic expansion, e.g., if

$$f := a^{-1}b = \frac{b}{a} = \frac{19669081321110688996}{2691},$$

then

$$f \equiv \underline{-14864362690}44957387929646558644 \bmod 10^{32}.$$

Note that the fraction $a^{-1}b$ is reduced, i.e., $a \perp b$. Suppose we have a number t (say $t = 32292$) which we suspect to be a multiple of the denominator of f, i.e., ft may be an integer. It turns out that we can assay if ft is integral by using only the high-order segment $h := -14864362689$ of $f \bmod 10^{32}$, shown underlined above. Note that

$$(\overbrace{-14864362690}^{h})(\overbrace{32292}^{t}) \equiv \overbrace{14520}^{c} \bmod 10^{12}.$$

There are now two key observations. First, provided the high-order segment is high enough, we will have $c < t$ if and only if t is multiple of the denominator of f. Second, the factor by which t is too large can be computed as $\gcd(t, c)$, which in this case is equal to 12: note that $32292/12 = 2691$. We call c an *integrality certificate* for f and t.

Now consider the above ideas but for polynomial matrices. We are starting with a left fraction $F = A^{-1}B$, and we have a matrix T which is a multiple of

the denominator of F, i.e., FT is over $K[x]$. Then F admits the two fraction descriptions: $F = (A)^{-1}(B)$ and $F = (A^{-1}BT)(T)^{-1}$. (Section 12 recalls some facts about fraction descriptions.) Notice that the second description is a right fraction. The first description may be very compact in the sense that both A and B have degree bounded by d. Unfortunately, the numerator $A^{-1}BT$ in the second description may be much larger, even if T has small degree. Our approach is to compute an integrality certificate C from a high-order lift of $A^{-1}B$. (Section 11 gives an algorithm for computing integrality certificates over $K[x]$.) Then, up to normalization, the matrix fraction CT^{-1} will have the same irreducible denominator as $(A^{-1}BT)(T^{-1})$, and $\deg C < \deg T$. The factor by which T is too large can then be computed via a matrix gcd involving T and C, instead of T and $A^{-1}BT$.

Our application of the above idea is to compute portions of the Hermite form of A, see below. Similar ideas have been used already by Villard (1996), in particular also for computing a column reduced form of A. Roughly speaking, a column reduced form of A is a matrix P with columns of minimal degree, and such that $AU = P$ for a unimodular matrix U. A key observation made in Villard (1996) is that P is a normalized denominator of $(I)(A)^{-1}$. In particular, $(I)(A)^{-1}$ also admits the description $(U)(P)^{-1}$. Here again, although $\deg P \leq \deg A$, the degree of U may be much larger than A. In Giorgi et al. (2003) this difficulty is avoided by using integrality certification as described here: a matrix fraction $(C)(P)^{-1}$ is computed which has the same denominator as $(U)(P)^{-1}$, but with $\deg C < \deg P$.

Smith form computation

Sections 13–17 are about computing the Smith form of a nonsingular $A \in K[x]^{n\times n}$. Recall the the Hermite column basis of A looks like

$$H = \begin{bmatrix} h_1 & & & & \\ * & h_2 & & & \\ * & * & h_3 & & \\ \vdots & \vdots & \vdots & \ddots & \\ * & * & * & \cdots & h_n \end{bmatrix},$$

and that $h_1 h_2 \cdots h_n$ is the monic associate of $\det A$. The Smith form of A looks like Diagonal$(s_1, s_2, \ldots, s_n)$, where s_i divides s_{i+1} for $1 \leq i \leq n-1$. See Section 12 for definitions of the Hermite and Smith form. Let $H(s, e)$ denote the submatrix of H comprised of rows and columns $s, s+1, \ldots, e$. Then $H(s, e)$

is also a Hermite column basis. Our approach for computing the determinant of A is to compute the Smith form $S(s,e)$ of $H(s,e)$ for various choices of s and e.

Section 13 presents an algorithm for computing a trailing submatrix $H(s,n)$ of H. Section 14 modifies the algorithm to compute $S(s,n)$ directly, without first computing $H(s,n)$. It is well known that the singleton matrix $H(n,n) = S(n,n) = [h_n]$ can be computed by solving a single linear system involving A. Our algorithm for $S(s,n)$ in Section 14 compute $S(s,n)$ for $(n-s+1) \leq nd/\deg S(n,k)$ in about the same time, d the degree of A.

Section 15 presents a key subroutine for the algorithm in Section 16, which in turn extends the algorithms of the previous sections to compute $S(s,e)$ for arbitrary e. Finally, Section 17 gives an algorithm for computing the Smith form. If the input matrix has been successfully preconditioned so that Diagonal$(h_1, h_2, \ldots, h_m)$ is equal to the Smith form of A, then the algorithm requires computing a sequence of only $O(\log n)$ blocks: $S(n,n), S(n-2, n-1), S(n-6, n-3), \ldots$, each block having double the dimension of the last but with degree at most half. The algorithm uses integrality certification to verify that Diagonal$(h_1, h_2, \ldots, h_m)$ is indeed equal to the Smith form.

4 X-adic representation

Let $X \in K[x]$ have degree greater than zero. By X-adic expansion of $a \in K[x]$ we mean to write $a = a_0 + a_1X + a_2X^2 + \cdots + a_lX^l$, l nonnegative, $\deg a_* < \deg X$. Throughout this paper, "degree" or "deg a" will always mean degree with respect to x. For example, if $\deg X = d$ and a_l is nonzero, then $dl \leq \deg a < d(l+1)$. The a_* are called coefficients of the X-adic expansion of a.

The ring $K[x]$ has the usual arithmetic operations $\{+, -, \times\}$. Here we define the three additional operations $\{\text{Left}, \text{Trunc}, \text{Inverse}\}$ and give some of their properties. These functions will implicitly be defined in terms of a proscribed X. Let $a \in K[x]$ and k be nonnegative. Suppose the X-adic expansion of a is $a = a_0 + a_1X + a_2X^2 + \cdots$. Then $\text{Left}(a,k) = a_k + a_{k+1}X + a_{k+2}X^2 + \cdots$ and $\text{Trunc}(a,k) = a_0 + a_1X + a_2X^2 + \cdots + a_{k-1}X^{k-1}$.

The Trunc operation truncates an X-adic expansion, e.g.,

$$\begin{aligned} a &= a_0 + a_1X + a_2X^2 + a_3X^3 + a_4X^4 + a_5X^5 + a_6X^6 + \cdots \\ \text{Trunc}(a,4) &= a_0 + a_1X + a_2X^2 + a_3X^3. \end{aligned}$$

The Left operation corresponds to division by a power of X; the name comes

from the fact that all coefficients of the X-adic expansion are shifted left, e.g.,

$$\begin{aligned} a &= a_0 + a_1X + a_2X^2 + a_3X^3 + a_4X^4 + a_5X^5 + a_6X^6 + \cdots \\ \text{Left}(a,3) &= a_3 + a_4X + a_5X^2 + a_6X^3 + a_7X^4 + a_8X^5 + a_9X^6 + \cdots. \end{aligned}$$

If $a \perp X$, then Inverse(a,k) denotes the unique $b \in K[x]$ such that $b = \text{Trunc}(b,k)$ and $\text{Trunc}(ab,k) = \text{Trunc}(ba,k) = 1$.

Let $a, b \in K[x]$ and k be nonnegative. A key property of the Left$(*,k)$ operation is linearity: $\text{Left}(a+b,k) = \text{Left}(a,k) + \text{Left}(b,k)$.

Lemma 1 *If* $\deg b < \deg(X^k)$ *then* $\text{Left}(a+b,k) = \text{Left}(a,k)$.

The next lemma observes that Left and Trunc commute.

Lemma 2 *If* $l \leq k$ *then* $\text{Left}(\text{Trunc}(a,k),l) = \text{Trunc}(\text{Left}(a,l),k-l)$.

X-adic expansions of matrices

Everything discussed above extends naturally to matrix polynomials: replace $a, b \in K[x]$ with $A, B \in K[x]^{n \times m}$. The operation Inverse takes as input a square matrix A, $\det A \perp X$.

The inverse of a nonsingular polynomial-matrix usually has rational function entries. For example, if

$$A = \begin{bmatrix} 1 & 1 \\ x & 1 \end{bmatrix} \quad \text{then} \quad A^{-1} = \begin{bmatrix} \frac{1}{1-x} & \frac{1}{x-1} \\ \frac{x}{x-1} & \frac{1}{1-x} \end{bmatrix} \in K(x)^{2\times 2}. \tag{8}$$

It is well known that denominators of reduced entries in A^{-1} are divisors of the determinant of A. In the above example $\det A = 1 - x$ which has degree bounded by one. In general, for a nonsingular $A \in K[x]^{n\times n}$ we have:

Fact 3 $\deg(\det A) \leq n \deg(A)$.

For a given $B \in K[x]^{n\times m}$, the matrix $A^{-1}B$ usually has rational function entries as opposed to polynomials. But $(\det A)A^{-1}B$ is a polynomial matrix and

Fact 4 $\deg((\det A)A^{-1}B) \leq \deg(B) + (n-1)\deg(A)$.

Consider again A from (8). Since $\det A \perp x$, we can express each entry of A^{-1} as an infinite x-adic expansion.

$$A^{-1} = \left[\begin{array}{c|c} 1 + x + x^2 + x^3 + \cdots & -1 - x - x^2 - x^3 + \cdots \\ \hline -x - x^2 - x^3 + \cdots & 1 + x + x^2 + x^3 + \cdots \end{array}\right]$$
$$= \begin{bmatrix} 1 & -1 \\ 0 & 1 \end{bmatrix} + \begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix} x + \begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix} x^2 + \cdots .$$

The last equation gives the infinite x-adic expansion of A^{-1}. In the rest of the paper, we will use A^{-1} similarly, e.g., A^{-1} denotes a possibly infinite X-adic expansion. In algorithms we will use Trunc(Inverse$(A, k)b, k)$; we will also write Trunc(A^{-1}, k) to mean the same.

Computation with X-adic polynomials

We are working over $K[x]$ with the operations $\{+, -, \times$, Left, Trunc, Inverse$\}$. The cost of these operations will depend essentially on our choice of representation for elements of $K[x]$. Let $d = \deg X$, and for $a \in K[x]$ let k be minimal such that $a = \text{Trunc}(a, k)$. Then $\deg a < kd$, and a can be stored as a list comprised of the first k coefficients of the X-adic expansion.

The conversion between the x-adic representation of a and the X-adic representation (either direction) can be computed with $O(\mathsf{M}(kd) \log k)$ field operations (Gathen and Gerhard, 1999, Theorem 9.15 and Exercise 9.20). In particular, if $k \leq 2$ then the cost of conversion is $O(\mathsf{M}(d))$ field operations; this case occurs often in our algorithms.

Let $b \in K[x]$ be given in addition to a, $\deg b \leq \deg a < kd$. Suppose a and b are represented as X-adic polynomials. Then the X-adic expansion of $a + b$ or $a - b$ can be computed with at most kd field operations. The X-adic expansion of ab can be computed with $O(\mathsf{M}(kd)(\log k))$ field operations by first converting a and b to x-adic representation, computing the product, then converting back to X-adic representation; we may sometimes use the coarser bound $\mathsf{M}(kd)(\log k) = O(\mathsf{B}(kd))$. Similarly, Inverse$(a, k)$ can be computed with $O(\mathsf{B}(kd))$ field operations; the cost of the conversions between X-adic and x-adic representation does not dominate here.

Operations Left, Trunc and multiplication by a power of X are free.

In most of our algorithms, we will make the implicit assumption that the input is given in X-adic representation. The output will also be given in X-adic representation.

5 X-adic lifting

Let $A \in K[x]^{n \times n}$ be nonsingular. Suppose we are given an $X \in K[x]$ such that $X \perp \det A$. In the X-adic expansion

$$A^{-1} = \overbrace{* + *X + \cdots + *X^{l-1}}^{C} + * X^l + *X^{l+1} + \cdots,$$

each $*$ lives in $K[x]^{n \times n}$ and has degree strictly less than $\deg X$. Let $B \in K[x]^{n \times *}$. The next definition and lemma give the key idea of X-adic lifting, cf. (4). Note that the division by X^k is exact.

Definition 5 $\text{Residue}(A, B, k) := (B - A\,\text{Trunc}(A^{-1}B, k))/X^k$.

Lemma 6 $A^{-1}B = \text{Trunc}(A^{-1}B, k) + A^{-1}\,\text{Residue}(A, B, k)X^k$.

The next result follows immediately.

Theorem 7 *Let* $C := \text{Trunc}(A^{-1}, l)$ *and* $R := \text{Residue}(A, B, k)$. *Then*

$$A^{-1}B = \overbrace{* + *X + \cdots + *X^{k-1}}^{\text{Trunc}(A^{-1}B,k)} + \overbrace{*X^k + \cdots + *X^{k+l-1}}^{\text{Trunc}(CR,l)X^k} + \cdots.$$

There are some well known variation of Theorem 7. For $i \geq 0$, define $C^{(i)} := \text{Trunc}(A^{-1}, 2^i)$ and $R^{(i)} := \text{Residue}(A, I, 2^i)$. Then $A^{-1} = C^{(i)} + A^{-1}R^{(i)}X^{2^i}$. Starting with $C^{(0)}$, a Newton iteration (or quadratic X-adic lifting) applies Theorem 7 for $l = k = 2, 4, 8, 16, \ldots$, doubling the number of coefficients of the expansion of A^{-1} at each step, cf. Figure 1:

$$\text{Inverse}(A, 2^k) := \left\lceil \begin{array}{l} C^{(0)} := \text{Inverse}(A, 1); \\ \textbf{for } i \textbf{ to } k \textbf{ do} \\ \quad R^{(i-1)} = (I - AC^{(i-1)})/X^{2^{i-1}}; \\ \quad C^{(i)} := C^{(i-1)} + \text{Trunc}(C^{(i-1)}R^{(i-1)}, 2^i)X^{2^{i-1}} \\ \textbf{od}; \\ \textbf{return } C^{(k)} \end{array} \right.$$

In all of the above, no assumptions are required on the degree of A or B.

X-adic lifting using short products

Let $k > 1$,

$$\mathrm{Trunc}(A^{-1}, k) = \overbrace{* + *X + *X^2 + \cdots + *X^{k-3}}^{C} + \overbrace{*X^{k-2} + *X^{k-1}}^{EX^{k-2}}, \qquad (9)$$

and

$$\mathrm{Trunc}(A^{-1}B, k) = * + *X + *X^2 + \cdots + *X^{k-3} + *X^{k-2} + DX^{k-1}. \qquad (10)$$

Suppose we want to compute only the single high-order coefficient D shown in (10). In general, we need all coefficients of $\mathrm{Trunc}(A^{-1}, k)$ to compute D. The next result shows that it suffices to have only E in case B has small degree. Let $d = \deg X$.

Theorem 8 *Assume* $\deg B \leq d$. *Then* $D = \mathrm{Trunc}(\mathrm{Left}(EB, 1), 1)$.

PROOF. $\mathrm{Trunc}(A^{-1}, k) = C + EX^{k-2}$. This gives $D = \mathrm{Left}(\mathrm{Trunc}(CB + EBX^{k-2}, k), k-1)$. Using Lemma 2 we can interchange the Left and Trunc to get $D = \mathrm{Trunc}(\mathrm{Left}(CB + EBX^{k-2}, k-1), 1)$. The key observation is that $\deg CB \leq \deg C + \deg B < (k-2)d + d \leq (k-1)d$. Using Lemma 1 now gives $D = \mathrm{Trunc}(\mathrm{Left}(EBX^{k-2}, k-1), 1)$. □

Now consider the computation of $R := \mathrm{Residue}(A, B, k)$, cf. Lemma 6. In general, we need all coefficients of $\mathrm{Trunc}(A^{-1}B, k)$ to compute R. The next result shows it suffices to have only D in case $\deg A$ and $\deg B$ are small enough.

Theorem 9 *Assume* $\deg A \leq d$ *and* $\deg B < kd$. *Then* $R = \mathrm{Left}(-AD, 1)$.

PROOF. By definition, $R = \mathrm{Left}(B - A\,\mathrm{Trunc}(A^{-1}B, k), k)$. Lemma 1 gives $R = \mathrm{Left}(-A\,\mathrm{Trunc}(A^{-1}B, k), k)$. Now substitute $\mathrm{Trunc}(A^{-1}B, k-1) + DX^{k-1}$ for $\mathrm{Trunc}(A^{-1}B, k)$, and apply Lemma 1 to see that the term $A\,\mathrm{Trunc}(A^{-1}B, k-1)$, which has degree strictly less than kd, vanishes. □

Recall Lemma 6: $A^{-1}B = \mathrm{Trunc}(A^{-1}B, k) + A^{-1}RX^k$. Thus, the problem of computing $A^{-1}B$ up to a certain order can be divided into two parts. The first is to compute $\mathrm{Trunc}(A^{-1}B, k)$. The second is to continue by computing the

expansion of $A^{-1}R$. The following corollary of Theorem 9 states that R may have small degree even if B has large degree.

Corollary 10 *Assume* $\deg A \le d$ *and* $\deg B < kd$. *Then* $\deg R < d$.

The next corollary is obtained by applying Theorems 8 and 9 in succession.

Corollary 11 *Assume* $\deg A \le d$ *and* $\deg B \le d$. *Then*

$$R = \mathrm{Left}(-A\,\mathrm{Trunc}(\mathrm{Left}(EB, 1), 1), 1).$$

6 High-order components of matrix inverse

Let $A \in K[x]^{n\times n}$ be nonsingular, $\det A \perp X$. In what follows, let $C^{(i)} = \mathrm{Trunc}(A^{-1}, 2^i)$. In this section we show how to recover the high order components of the inverse of A: $E^{(i)} = \mathrm{Left}(C^{(i)}, 2^i - 2)$ for $i = 1, 2, \ldots, k$. To see more clearly what we are computing, write the X-adic expansion of A^{-1} as

$$A^{-1} = C_0 + C_1X + C_2X^2 + \cdots.$$

Then

$$\begin{aligned}
C^{(1)} &= \overbrace{C_0 + C_1X}^{E^{(1)}} \\
C^{(2)} &= C_0 + C_1X + \overbrace{C_2X^2 + C_3X^3}^{E^{(2)}X^2} \\
C^{(3)} &= C_0 + C_1X + \cdots + C_5X^5 + \overbrace{C_6X^6 + C_7X^7}^{E^{(3)}X^6} \\
&\vdots
\end{aligned}$$

Algorithm 1 (`HighOrderComp`) recovers only the high order components $E^{(*)}$ as shown above.

Algorithm 1 `HighOrderComp`$[X](A, k)$
Input: $A \in K[x]^{n\times n}$ and $k \ge 2$.
Output: $(E^{(1)}, E^{(2)}, \ldots, E^{(k)})$ as shown above.
Condition: $X \perp \det A$ and $d = \deg X \ge \deg A$.

(1) $L := \mathrm{Inverse}(A, 1)$;
$H := \mathrm{Trunc}(L\,\mathrm{Left}(I - AL, 1), 1)$;
$E^{(1)} := L + XH$;

(2) **for** i **from** 2 **to** k **do**
$L := \text{Trunc}(\text{Left}(E^{(i-1)}\,\text{Left}(-AL, 1), 1), 1)$;
$H := \text{Trunc}(\text{Left}(E^{(i-1)}\,\text{Left}(-AH, 1), 1), 1)$;
$E^{(i)} := L + XH$
od;
return $(E^{(1)}, E^{(2)}, \ldots, E^{(k)})$

We now prove that the algorithm is correct. Let $[X](A, k)$ be a valid input tuple. Let $(L^{(i)}, H^{(i)})$ be equal to (L, H) as computed during the loop in phase 2 with index i. Phase 1 computes $(L^{(1)}, H^{(1)}) = (C_0, C_1)$ and $E^{(1)} = C_0 + XC_1$. Using induction on j we now prove that

$$L^{(j)} = C_{2^j-2} \tag{11}$$
$$H^{(j)} = C_{2^j-1} \tag{12}$$
$$E^{(j)} = C_{2^j-2} + XC_{2^j-1} \tag{13}$$

for $j = 1, 2, \ldots, k$. The base case $j = 1$ has already been established. That (13) follows from (11) and (12) is clear. Let $i > j$. Our goal is to show (11) and (12) hold for $j = i$. It will be sufficient to show that (12) holds since the proof of (11) is analogous.

The algorithm computes

$$H^{(i)} := \text{Trunc}(\text{Left}(E^{(i-1)}\,\overbrace{\text{Left}(-AH^{(i-1)}, 1)}^{R}, 1), 1).$$

By Theorem 9, $R = \text{Residue}(A, I, 2^{i-1})$. Theorem 8 now gives that $H^{(i)}$ is equal to the coefficient of $X^{2^{i-1}-1}$ in the X-adic expansion of $A^{-1}R$. Since $A^{-1} = C^{(i-1)} + A^{-1}RX^{2^{i-1}}$, this coefficient is equal to C_{2^i-1}. This shows that (12) holds. The proof that (11) holds for $j = i$ is analogous. This ends the inductive proof of correctness of the algorithm.

Inverse$(A, 1)$ costs $\overline{\mathsf{MM}}(n, d)$ field operations. The remaining steps cost $O(k\,\mathsf{MM}(n, d))$ field operations.

Proposition 12 *Algorithm 1* (`HighOrderComp`) *is correct. The cost of the algorithm is* $O(k\,\mathsf{MM}(n, d) + \overline{\mathsf{MM}}(n, d))$ *field operations.*

7 Unimodularity certification

We present an algorithm to assay if a given $A \in K[x]^{n\times n}$ is unimodular. Our approach is to assay if the x-adic expansion of A^{-1} is finite.

Algorithm 2 `UnimodularityCert`(A)
Input: $A \in K[x]^{n \times n}$.
Output: True in case A is unimodular, otherwise false.

(1) **if** $\det(A \bmod x) = 0$ **then return** false **fi**;
$d := \deg A$;
$X := x^d$;
(2) $k := \lceil \log_2(n+3) \rceil$;
$(*, *, \ldots, *, E) :=$ `HighOrderComp`$[X](A, k)$;
(3) **if** E is the zero matrix **then**
return true
else
return false
fi

We now prove correctness. Let k and E be as computed in phase 2. Then $\mathrm{Trunc}(A^{-1}, 2^k) = \mathrm{Trunc}(A^{-1}, 2^k - 2) + EX^{2^k-2}$. Let $R := \mathrm{Residue}(A, B, 2^k)$.

On the one hand, suppose E is the zero matrix. Then Theorem 9 gives $R = \mathrm{Left}(-A\,\mathrm{Left}(E, 1), 1)$, i.e., R is the zero matrix. Since $A^{-1} = \mathrm{Trunc}(A^{-1}, 2^k) + A^{-1}RX^{2^k}$, the expansion of A^{-1} is finite. This shows that a return value of true is always correct.

On the other hand, the parameter k is chosen so that $\deg(X^{2^k-2})$ is strictly greater than degrees of numerators in $A^{-1} \in K(x)^{n \times n}$. Thus, if A^{-1} is over $K[x]$ then E will be the zero matrix.

Proposition 13 *Algorithm 2* (`UnimodularityCert`) *is correct. The cost of the algorithm is* $O((\log n)\mathsf{MM}(n, \deg A) + \overline{\mathsf{MM}}(n, \deg A))$ *field operations.*

8 Series solution — small degree right hand side

Let $A \in K[x]^{n \times n}$ be nonsingular, $\det A \perp X$. Let $b \in K[x]^{n \times 1}$. We present an algorithm for computing the X-adic expansion of $A^{-1}b$ up to a given order. The algorithm requires both $\deg b$ as well as $\deg A$ to be bounded by d, $d = \deg X$.

Algorithm 3 `SeriesSolSmall`$[X](A, b, k)$
Input: $A \in K[x]^{n \times n}$, $b \in K[x]^{n \times 1}$, and $k \geq 2$.
Output: $\mathrm{Trunc}(A^{-1}b, 2^k)$.
Condition: $X \perp \det A$ and $d = \deg X \geq \max(\deg A, \deg b)$.

(1) $E^{(1)}, E^{(2)}, \ldots, E^{(k-1)} :=$ `HighOrderComp`$[X](A, k-1)$;
(2) $B := \left[\, b \,\middle|\, O \,\right]$ where O is the $n \times (2^k - 1)$ zero matrix;

for i **from** $k-1$ **by** -1 **to** 1 **do**
$\bar{B} :=$ the first $2^k - 2^i$ columns of B;
$\bar{R} := \mathrm{Left}(-A\,\mathrm{Trunc}(\mathrm{Left}(E^{(i)}\bar{B}, 1), 1), 1)$;
$R := \left[\, O \,\middle|\, \bar{R} \,\right]$ where O is the $n \times 2^i$ zero matrix;
$B := B + R$;
od;
$B := \mathrm{Trunc}(E^{(1)}B, 2)$;
(3) # Let $B = \left[\, d_0 \middle| 0 \middle| d_2 \middle| 0 \middle| \cdots \middle| d_{2^k-2} \middle| 0 \,\right]$.
$B := d_0 + d_2\,X^2 + \cdots + d_{2^k-2}\,X^{2^k-2}$;
return B

We now prove correctness. Let $[X](A, b, k)$ be valid input tuple.

The purpose of phase 2 is to compute all the coefficients of $\mathrm{Trunc}(A^{-1}b, 2^k)$. The idea is most clearly explained with an example: consider the case $k = 4$. Let $r_j = \mathrm{Residue}(A, b, j)$, $j \in \{0, 2, 4, \ldots, 14\}$. Then $r_0 = b$, and repeated application of Lemma 6 gives

$$\begin{aligned}
\mathrm{Trunc}(A^{-1}b, 16) &= \mathrm{Trunc}(A^{-1}r_0, 16) \\
&= \mathrm{Trunc}(A^{-1}r_0, 8) + \mathrm{Trunc}(A^{-1}r_8, 8)X^8 \\
&\;\;\vdots \\
&= \mathrm{Trunc}(A^{-1}r_0, 2) + \cdots + \mathrm{Trunc}(A^{-1}r_{14}, 2)X^{14}
\end{aligned}$$

Our initial problem is to compute the solution to a single linear system up to order 16. At the start of the loop we have

$$B = \left[\, r_0 \;0\;0\;0\;0\;0\;0\;0\;0\;0\;0\;0\;0\;0\;0 \,\right].$$

The j'th column of B may be thought to be implicitly multiplied by X^{j-1}, cf. (7). After the loop completes with index i, the matrix B is as follows:

$$\begin{array}{l|cccccccc}
i = 3 & r_0 & & & & r_8 & & & \\
i = 2 & r_0 & & r_4 & & r_8 & & r_{12} & \\
i = 1 & r_0 & r_2 & r_4 & r_6 & r_8 & r_{10} & r_{12} & r_{14}
\end{array}\,.$$

Thus, each pass through the loop doubles the number of systems we need to solve, but halves the order of precision to which we need the solutions. After the loop completes we need to solve 8 systems up to order X^2; this is done by the last line of phase 2.

We now give a formal proof of correctness for phase 2. We will prove by induction on s, $s = k, k-1, k-2, \ldots, 1$, that after the loop completes with index $i = s$, we have

$$A^{-1}b \equiv \sum_{j=1}^{2^k} \mathrm{Trunc}(A^{-1}\,\mathrm{Column}(B,j), 2^s)X^{j-1} \pmod{X^{2k}}, \tag{14}$$

and $\deg B \leq d$. The base case $s = k$ corresponds to the state of B before the first iteration of the loop: (14) holds.

Now assume (14) holds with $s = i+1$, some $i \geq 1$. Let $c_j = \mathrm{Column}(B,j)$, where B is at the start of the loop with index i. Then (14) with $s = i+1$ gives

$$A^{-1}b \equiv \sum_{j=1}^{2^k} \mathrm{Trunc}(A^{-1}c_j, 2^{i+1})X^{j-1} \pmod{X^{2k}}. \tag{15}$$

We need to show that (14) holds with $s = i$ after the loop completes with index i. Let $\bar{c}_j := \mathrm{Residue}(A, c_j, 2^i)$. Then Lemma 6 gives

$$\mathrm{Trunc}(A^{-1}c_j, 2^{i+1}) = \mathrm{Trunc}(A^{-1}c_j, 2^i) + \mathrm{Trunc}(A^{-1}\bar{c}_j, 2^i)X^{2^i}.$$

Substituting into (15) gives

$$A^{-1}b \equiv \sum_{j=1}^{2^k} \mathrm{Trunc}(A^{-1}c_j, 2^i)X^{j-1} + \sum_{j=1}^{2^k} \mathrm{Trunc}(A^{-1}\bar{c}_j, 2^i)X^{2^i+j-1} \pmod{X^{2k}}. \tag{16}$$

Let $\bar{R}$ and R be as computed in the loop. By Corollary 11, $\mathrm{Column}(\bar{R}, j) = \bar{c}_j$ for $1 \leq j \leq 2^k - 2^i$. Substitute $\bar{c}_j = \mathrm{Column}(R, 2^i + j - 1)$ for $1 \leq j \leq 2^k - 2^i$ into (16), and use the observation that $\mathrm{Trunc}(A^{-1}\bar{c}_j, 2^i)X^{2^i+j-1} \equiv 0 \bmod X^{2k}$ in case $j > 2^k - 2^i$, to get

$$A^{-1}b \equiv \sum_{j=1}^{2^k} \mathrm{Trunc}(A^{-1}(c_j + \mathrm{Column}(R,j)), 2^i)X^{j-1} \pmod{X^{2k}}.$$

Thus, after the update $B := B + R$, B will satisfy (14) with $s = i$. Corollary 10 gives $\deg R < d$. Thus $\deg B \leq \max(\deg B, \deg R) \leq d$. This completes the inductive proof.

Now we estimate the cost. The cost of phase 1 is given by Proposition 12. In phase 2, the number of nonzero columns in B is doubling each time through the loop. The last iteration of the loop dominates. The cost is $O((2^k/n)\mathsf{MM}(n,d))$

field operations if $2^k > kn$. If $2^k \leq kn$ the cost is dominated by that of phase 1. Finally, phase 3 multiplies each column of B by the appropriate power of X and adds all the columns together. Under our cost model this is free.

Proposition 14 *Algorithm 3* (`SeriesSolSmall`) *is correct. The cost of the algorithm is* $O((k + 2^k/n)\mathsf{MM}(n,d) + \overline{\mathsf{MM}}(n,d))$ *field operations.*

9 Series solution

Let $A \in K[x]^{n \times n}$ be nonsingular, $\det A \perp X$. Let $b \in K[x]^{n \times m}$. We present an algorithm for computing the X-adic expansion of $A^{-1}b$ up to a given order. The algorithm here extends the algorithm given in the previous section: no assumption is required on the degree of b, and b may have column dimension m, $m > 1$.

Algorithm 4 `SeriesSol`$[X](A, b, k)$
Input: $A \in K[x]^{n \times n}$, $b \in K[x]^{n \times m}$, and $k \geq 2$.
Output: $\mathrm{Trunc}(A^{-1}b, 2^k)$.
Condition: $X \perp \det A$ and $d = \deg X \geq \deg A$.

(1) $E^{(1)}, E^{(2)}, \ldots, E^{(k-1)} :=$ `HighOrderComp`$[X](A, k-1)$;
(2) # Let X-adic expansion of b be $b_0 + b_1 X + b_2 X^2 + \cdots$.
$B := \left[b_0 \middle| b_1 \middle| \cdots \middle| b_{2^k-1} \right]$;
for i **from** $k-1$ **by** -1 **to** 1 **do**
$\bar{B} :=$ the first $m2^k - m2^i$ columns of B;
$\bar{R} := \mathrm{Left}(-A\,\mathrm{Trunc}(\mathrm{Left}(E^{(i)}\bar{B}), 1), 1), 1)$;
$R := \left[O \middle| \bar{R} \right]$ where O is the $n \times m2^i$ zero matrix;
$B := \bar{B} + R$;
od;
$B := \mathrm{Trunc}(E^{(1)}B, 2)$;
(3) # Let $B = \left[d_0 \middle| d_1 \middle| \cdots \middle| d_{2^k-1} \right]$.
$B := d_0 + d_1 X + d_2 X^2 + \cdots + d_{2^k-2} X^{2^k-2} + \mathrm{Trunc}(d_{2^k-1}, 1) X^{2^k-1}$;
return B

We now prove correctness. Let $[X](A, b, k)$ be a valid input tuple.

Suppose $m = 1$. Then Algorithm 4 (`SeriesSol`) is identical to Algorithm 3 (`SeriesSolSmall`), except that b_i is not necessarily zero for $i > 0$. The formal proof of correctness for phase 2 carries over directly. There are some minor differences in phase 3. Here, d_i may not necessarily be zero for odd i, and in particular we need to truncate the expansion of d_{2^k-1}.

Now we estimate the cost. The analysis for phase 2 is slightly different than for Algorithm 3 (`SeriesSolSmall`). Here, the number of nonzero columns in B is bounded by $O(m2^k)$ in each iteration of the loop. This gives the cost estimate of $O(k\lceil m2^k/n\rceil \mathsf{MM}(n,d))$ field operations for phase 2. Phase 3 multiplies each column of B by the appropriate power of X and adds all the columns together. Unlike the corresponding phase in Algorithm 3 (`SeriesSolSmall`), we may have to perform some additions here, but the cost of this phase is dominated by that of phase 2.

Proposition 15 *Algorithm 4* (`SeriesSol`) *is correct. The cost of the algorithm is* $O(k\lceil m2^k/n\rceil \mathsf{MM}(n,d) + \overline{\mathsf{MM}}(n,d))$ *field operations.*

Let $(A, b, *)$ be a valid input tuple to Algorithm 4 (`SeriesSol`), b a column vector. Based on Facts 3 and 4, Algorithm 5 (`RationalSol`) computes the minimal degree monic factor g of $\det A$ such that $gA^{-1}b$ is over $K[x]$.

Algorithm 5 `RationalSol`$[X](A, b)$
Input: $A \in K[x]^{n\times n}$ and $b \in K[x]^{n\times 1}$.
Output: $(gA^{-1}b, g) \in (K[x]^{n\times 1}, K[x])$ with g monic of minimal degree.
Condition: $X \perp \det A$ and $d = \deg X \geq \deg A$.

(1) $N := (n-1)\deg A$;
$k :=$ the smallest integer ≥ 2 such that $2^k > N + n\deg A$;
$v :=$ `SeriesSol`$[X](A, b, k)$;
(2) $g := 1$;
for i **to** n **do**
$h :=$ minimal deg monic polynomial with $\deg \mathrm{Trunc}(h(gv[i]), 2^k) \leq N$;
$g := hg$
od;
return (gv, g)

Each computation of h in phase 2 costs $O(\mathsf{B}(2^k d))$ field operations using rational function reconstruction, see (Gathen and Gerhard, 1999, Sections 5.7 and 11.1). This bounds the cost of converting between X-adic and x-adic representations.

Corollary 16 *Algorithm 5* (`RationalSol`) *is correct. If* $(\deg b)/d = O(n)$, *then the cost of the algorithm is:*

- $O((\log n)\mathsf{MM}(n,d) + \overline{\mathsf{MM}}(n,d) + n\,\mathsf{B}(nd))$ *field operations, or*
- $O((\log n)\mathsf{MM}(n)\mathsf{B}(d))$ *field operations, assuming* $\mathsf{B}(n) = O(\mathsf{MM}(n)/n)$ *and* $n^{2+\gamma} = O(\mathsf{MM}(n))$ *for some positive* γ.

10 High-order lifting

Let $A \in K[x]^{n\times n}$ be nonsingular, $\det A \perp X$. Let $B \in K[x]^{n\times m}$. We present an algorithm to compute a segment $H = \mathrm{Left}(\mathrm{Trunc}(A^{-1}B, h+k), h)$ of coefficients from the X-adic expansion of $A^{-1}B$. Note that

$$A^{-1}B = * + *X + \cdots + \overbrace{*X^h + \cdots + *X^{h+k-1}}^{HX^h} + *X^{h+k} + \cdots \tag{17}$$

If $h = 0$ we can use Algorithm 4 (`SeriesSol`) to compute H. In high-order lifting, what is important is that h be larger than some specified bound, say $h > l$ for a given l. The particular value of h is not important, only that $h > l$. Given l, the algorithm here chooses $h := 2^{\bar{l}} + 2^{\bar{k}}$, where $\bar{k}$ is chosen to be the smallest integer that satisfies $2^{\bar{k}}d > \deg B$, and $\bar{l}$ is then chosen to be the smallest integer that satisfies $2^{\bar{l}} + 2^{\bar{k}} > l$.

The point of the algorithm here is that the cost depends linearly on $\log l$, not on l. This is important because in typical applications $l \gg k$.

Algorithm 6 `HighOrderLift`$[X](A, B, l, k)$
Input: $A \in K[x]^{n\times n}$, $B \in K[x]^{n\times m}$, $l \geq 2$, and k a power of two.
Output: $\mathrm{Left}(\mathrm{Trunc}(A^{-1}B, h+k), h)$ for some $h > l$.
Condition: $X \perp \det A$ and $d = \deg X \geq \deg A$.

(1) $\bar{k} :=$ the smallest integer ≥ 2 such that $2^{\bar{k}}d > \deg B$;
$D := \mathrm{Left}(\texttt{SeriesSol}[X](A, B, \bar{k}), 2^{\bar{k}} - 1)$;
$\bar{R} := \mathrm{Left}(-AD, 1)$;
(2) $\bar{l} :=$ the smallest integer ≥ 2 such that $2^{\bar{l}} + 2^{\bar{k}} > l$;
$(*, *, \ldots, *, E^{(\bar{l})}) := \texttt{HighOrderComp}[X](A, \bar{l})$;
$R := \mathrm{Left}(-A\,\mathrm{Trunc}(\mathrm{Left}(E^{(\bar{l})}\bar{R}, 1), 1), 1)$;
(3) $H := \texttt{SeriesSol}[X](A, R, \log_2 k)$;
return H

The purpose of phase 1 is to reduce a possible large degree right hand side B to a small degree residue $\bar{R}$. After phase 1 finishes, $\bar{R} = \mathrm{Residue}(A, B, 2^{\bar{k}})$ (Theorem 9), $\deg \bar{R} < d$ (Corollary 10), and Lemma 6 gives

$$A^{-1}B = \mathrm{Trunc}(A^{-1}B, 2^{\bar{k}}) + A^{-1}\bar{R}X^{2^{\bar{k}}}.$$

After phase 2 finishes, $R = \mathrm{Residue}(A, \bar{R}, 2^{\bar{l}})$ (Corollary 11), and

$$A^{-1}B = \mathrm{Trunc}(A^{-1}B, h) + A^{-1}RX^h,$$

where $h = 2^{\bar{l}} + 2^{\bar{k}}$.

Phase 1 costs $O((\log \deg B)\lceil m(\deg B)/(nd)\rceil \mathsf{MM}(n,d) + \overline{\mathsf{MM}}(n,d))$ field operations (Proposition 15), phase 2 costs $O((\log l)\mathsf{MM}(n,d) + \overline{\mathsf{MM}}(n,d))$ field operations (Proposition 12), and phase 3 costs $O((\log k)\lceil mk/n\rceil \mathsf{MM}(n,d) + \overline{\mathsf{MM}}(n,d))$.

Proposition 17 *Algorithm 6* (`HighOrderLift`) *is correct. If* $\log l = O(\log n)$ *and both* $m \times k$ *and* $m \times (\deg B)/d$ *are* $O(n)$*, then the cost of the algorithm is* $O((\log n)\mathsf{MM}(n,d) + \overline{\mathsf{MM}}(n,d))$ *field operations.*

11 Integrality certification

Let $A \in K[x]^{n\times n}$ be nonsingular, $\det A \perp X$. Let $B \in K[x]^{n\times m}$ and $T \in K[x]^{m\times m}$. This section presents an algorithm to assay if $A^{-1}BT$ is integral, i.e., if $A^{-1}BT$ is over $K[x]$. Let

$$S = \mathrm{Trunc}(A^{-1}BT, h+k).$$

We will specify h and k below. For now, note that $AS \equiv BT \bmod X^{h+k}$. Thus, if $\deg AS$ and $\deg BT$ are $< (h+k)d$, then $AS = BT$, i.e., $S = A^{-1}BT$.

Lemma 18 *If* $\deg AS, \deg BT < (h+k)d$*, then* $A^{-1}BT$ *is integral.*

Let $H = \mathrm{Left}(\mathrm{Trunc}(A^{-1}B, h+k), h)$, cf. (17). Assume that k satisfies $\deg T < kd$. Then

$$\overbrace{\mathrm{Trunc}(A^{-1}BT, h+k)}^{S} = \overbrace{\mathrm{Trunc}(A^{-1}B, h)T}^{\text{degree} < hd + \deg T} + \overbrace{\mathrm{Trunc}(HT, k)}^{C} X^h. \qquad (18)$$

Theorem 19 *Assume* h *satisfies* $(n-1)\deg A + \deg B + \deg T < hd$ *and* k *satisfies* $\deg T + \deg A < kd$*. Then* $A^{-1}BT$ *is integral if and only if* $\deg C < \deg T$*.*

PROOF. (If:) Assume $\deg C < \deg T$. Then $\deg S < hd + \deg T$ (cf. (18)). Now apply Lemma 18, noting that $\deg AS \le \deg A + \deg S$. **(Only if:)** Assume $A^{-1}BT$ is integral. Then Fact 4 gives $\deg A^{-1}BT \le (n-1)\deg A + \deg B + \deg T$, which is $< hd$. Considering (18), we must have $\mathrm{Left}(S,h)$ equal to the zero matrix, which implies $C = -\mathrm{Left}(\mathrm{Trunc}(A^{-1}B,h)T), h)$. □

The next corollary will be useful later on. The corollary observes that C will be invariant of the choice of k. Of course, h and k are still required to satisfy the assumptions of Theorem 19.

Corollary 20 *If $A^{-1}BT$ is integral, then $A^{-1}BT = \mathrm{Trunc}(A^{-1}B, h)T + C$.*

In case of integrality, the algorithm returns C, the *integrality certificate*.

Algorithm 7 `IntegralityCert`$[X](A, B, T)$
Input: $A \in K[x]^{n\times n}$, $B \in K[x]^{n\times m}$, and $T \in K[x]^{m\times m}$.
Output: An integrality certificate if $A^{-1}BT$ is over $K[x]$, otherwise fail.
Condition: $X \perp \det A$ and $d = \deg X \geq \deg A$.

(1) $h :=$ the smallest integer such that $hd > (n-1)d + \deg B + \deg T$;
$k :=$ the smallest power of two such that $kd > \deg T + d$;
$H :=$ `HighOrderLift`$[X](A, B, h, k)$;
(2) $C := \mathrm{Trunc}(HT, k)$;
if $\deg C < \deg T$ **then**
return C
else
return fail
fi

The cost estimate for phase 1 is given by Proposition 17. For the multiplication of HT in phase 2 we need to take care to include the cost of conversion between X-adic and x-adic representation.

Proposition 21 *Algorithm 7* (`IntegralityCert`) *is correct. If all of m, $m \times (\deg B)/d$ and $m \times (\deg T)/d$ are $O(n)$, then the cost of the algorithm is:*

- $O((\log n)\mathsf{MM}(n,d) + \overline{\mathsf{MM}}(n,d) + (n/m)\mathsf{MM}(m, nd/m) + nm\,\mathsf{B}(nd/m))$ *field operations, or*
- $O((\log n)\mathsf{MM}(n)\mathsf{B}(d))$ *field operations, assuming* $\mathsf{B}(n) = O(\mathsf{MM}(n)/n)$ *and* $n^{2+\gamma} = O(\mathsf{MM}(n))$ *for some positive* γ.

Extension to integer matrices

We show how the idea of integrality certification described above for polynomial matrices can be adapted to integer matrices. For convenience, we are going to work modulo powers of 10 in the symmetric range. For $a \in \mathbb{Z}$ and k nonnegative, let $\mathrm{Trunc}(a,k)$ and $\mathrm{Left}(a,k)$ be the unique integers that satisfy the following:

$$a = \mathrm{Left}(a,k)10^k + \mathrm{Trunc}(a,k), \quad -\frac{10^k}{2} < \mathrm{Trunc}(a,k) \leq \frac{10^k}{2}. \tag{19}$$

In Maple(TM) we could define these operators as follows:

```
Trunc(a,k) := proc(a,k) mods(a,10^k) end:
 Left(a,k) := proc(a,k) (a-Trunc(a,k))/10^k end:
```

The computation with integer is considerably complicated because of the presence of carries. We will need the following lemmata, which follow from the definition of Left and Trunc.

Lemma 22 $|\mathrm{Left}(a,k)| \le (|a| + |\mathrm{Trunc}(a,k)|)/10^k \le |a|/10^k + 1/2$.

Lemma 23 *If* $|a| < 10^k/2$, *then* $\mathrm{Trunc}(a,k) = a$.

We now develop the analogue of Theorem 19 for the integer setting. Suppose $\det A \perp 10$. Let

$$
\begin{aligned}
S &= \mathrm{Trunc}(A^{-1}BT, h+k),\\
H &= \mathrm{Left}(\mathrm{Trunc}(A^{-1}B, h+k), h), \text{ and}\\
C &= \mathrm{Trunc}(HT, k).
\end{aligned}
$$

Note that $\mathrm{Trunc}(AS, h+k) = \mathrm{Trunc}(BT, h+k)$. Thus, if $\|AS\|_\infty < 10^k/2$ and $\|BT\|_\infty < 10^k/2$, then $AS = BT$ (Lemma 23).

Lemma 24 *If* $\|AS\|_\infty, \|BT\|_\infty < 10^k/2$, *then* $A^{-1}BT$ *is integral.*

Before stating the main result, we give two more lemmas. The fact that the absolute value norm over $\mathbb{Z}$ is Archimedian accounts for the first lemma. The second lemma follows from the first, Cramer's rule, and Hadamard's inequality.

Lemma 25 *If* $P \in \mathbb{Z}^{*\times m}$, *and* $T \in \mathbb{Z}^{m\times *}$, *then* $\|PT\|_\infty \le m\|P\|_\infty\|T\|_\infty$.

Lemma 26 $\|\det(A)A^{-1}BT\|_\infty \le mn^{n/2}(\|A\|_\infty)^{n-1}\|B\|_\infty\|T\|_\infty$.

The analogue of (18) is

$$
S = \mathrm{Trunc}(\overbrace{\mathrm{Trunc}(A^{-1}B, h)T}^{|\cdot| \le (m/2)\|T\|_\infty 10^h} + C10^h), h+k). \tag{20}
$$

The magnitude bound in (20) follows from (19) and Lemma 25. The outermost Trunc operation on the right hand side of (20) is required because the Trunc operation over $\mathbb{Z}$ is not linear, e.g., $\mathrm{Trunc}(5+1,1) \ne \mathrm{Trunc}(5,1) + \mathrm{Trunc}(1,1)$.

Theorem 27 *Assume* h *satisfies* $mn^{n/2}(\|A\|_\infty)^{n-1}\|B\|_\infty\|T\|_\infty < 10^h/2$ *and* k *satisfies* $nm\|A\|_\infty\|T\|_\infty < 10^k/2$. *Then* $A^{-1}BT$ *is integral if and only if* $\|C\|_\infty \le (m/2)\|T\|_\infty$.

PROOF. (If:) Assume $\|C\| \leq (m/2)\|T\|_\infty$. Then $S \leq m\|T\|_\infty 10^h$ (cf. (20)). Now apply Lemma 24, noting that $\|AS\| \leq nm\|A\|_\infty\|T\|_\infty 10^h$ (Lemma 25). **(Only if:)** Assume $A^{-1}BT$ is integral. Then $\|A^{-1}BT\|_\infty < 10^h/2$ (Lemma 26). Lemma 23 applied to both sides of (20) gives $A^{-1}BT = \mathrm{Trunc}(A^{-1}B,h)T + C10^h$. Now note that $\mathrm{Trunc}(\mathrm{Trunc}(A^{-1}B,h)T,h) = A^{-1}BT$ to deduce that $C = \mathrm{Left}(\mathrm{Trunc}(A^{-1}B,h)T,h)$. The magnitude bound in (20), together with Lemma 22, gives $\|C\|_\infty \leq (m/2)\|T\|_\infty + \|A^{-1}BT\|_\infty/10^h < (m/2)\|T\|_\infty + 1/2$. Finally, note that $\|C\|_\infty \in \mathbb{Z}$, yielding the required bound: $\|C\|_\infty \leq \lceil (m/2)\|T\|_\infty + 1/2 - 1 \rceil$. $\Box$

Worked example

We will assay if $A^{-1}BT$ is integral, where

$$A := \begin{bmatrix} -28 & -11 & -56 & -39 \\ -5 & 42 & -10 & 37 \\ 22 & -44 & -25 & 44 \\ -32 & 3 & 38 & 46 \end{bmatrix}, \quad B := \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 1 & 0 \\ 0 & 1 \end{bmatrix}, \quad T := \begin{bmatrix} 3969 & 0 \\ 0 & 3969 \end{bmatrix}.$$

Let $h = 90$ and $k = 8$. Then the assumptions of Theorem 27 are satisfied. Let $H := \mathrm{Left}(\mathrm{Trunc}(A^{-1}B,98),90)$ and $C := \mathrm{Trunc}(HT,8)$:

$$H = \begin{bmatrix} -12194507 & -23935500 \\ -24086672 & 42529604 \\ -5946082 & 33232552 \\ 24086672 & -42529604 \end{bmatrix} \text{ and } C = \begin{bmatrix} 1717 & 500 \\ -1168 & -1724 \\ 542 & -1112 \\ 1168 & 1724 \end{bmatrix}. \tag{21}$$

Since $\|C\|_\infty = 1724 < (m/2)\|T\|_\infty = 3969$, we conclude that $A^{-1}BT$ is integral. Note that we have not described how to efficiently compute the high-order lift H. This requires some new techniques and will be the subject of a future paper.

12 The Hermite basis and fraction descriptions

This section introduces some notation and recalls some definitions and facts that we will need in subsequent sections. The notation StackMatrix(A_1, A_2) is defined by

$$\text{StackMatrix}(A_1, A_2) = \left[\begin{array}{c} A_1 \\ \hline A_2 \end{array}\right].$$

A matrix A is a *left multiple* of B if $A = *B$ for a matrix $*$ over $K[x]$. Analogously, A is a *right multiple* of B if $A = B*$. A matrix $G \in K[x]^{m\times m}$ is a *row basis* for a full column rank $A \in K[x]^{n\times m}$ if A and G are left multiples of each other. *Column basis* is defined analogously.

Corresponding to every full column rank $A \in K[x]^{n\times m}$ is a unimodular matrix $U \in K[x]^{n\times n}$ such that

$$UA = \text{StackMatrix}(H, 0) = \left[\begin{array}{cccc} h_1 & h_{12} & \cdots & h_{1m} \\ & h_2 & \cdots & h_{2m} \\ & & \ddots & \vdots \\ & & & h_m \\ \hline & & & \end{array}\right] \in K[x]^{n\times m},$$

with off-diagonal entries h_{*j} in H of degree strictly less than the monic diagonal entry h_j in the same column. The principal nonsingular submatrix H is the unique *Hermite row basis* of A. In particular, AH^{-1} is over $K[x]$ and has Hermite row basis equal to I_m. If A is square as well as nonsingular, then $U := HA^{-1}$ is the unique unimodular transforming matrix such that $UA = H$.

Hermite column basis is defined analogously: for $A \in K[x]^{m\times n}$ with full row rank m, the Hermite column basis of A is the transpose of the Hermite row basis of Transpose(A).

$$AU = \left[\begin{array}{c|c} H & \end{array}\right] = \left[\begin{array}{ccc|c} h_1 & & & \\ h_{21} & h_2 & & \\ \vdots & \vdots & \ddots & \\ h_{m1} & h_{m2} & \cdots & h_m \end{array}\right] \in K[x]^{m\times n}.$$

Fact 28 *Full row rank matrices A and B over $K[x]$ are right multiples of each other if and only A and B have the same Hermite column basis.*

We now recall some facts about matrix gcds and fractions. (See for example Kailath (1980) for a detailed study.) Suppose A_1 and A_2 are over $K[x]$, with same column dimension, and A_2 is nonsingular. Then a right matrix gcd of A_1 and A_2 is any row basis for StackMatrix(A_1, A_2). Now let $F \in K(x)^{n \times m}$ have rank m.

Definition 29 *A nonsingular matrix $D \in K[x]^{m \times m}$ is an* irreducible right denominator *of F if FD is over $K[x]$, and I_m is a right gcd of* StackMatrix(FD, D).

Irreducible right denominators of F are right equivalent (equal up to post-multiplication by a unimodular matrix on the right) in $K[x]^{m \times m}$. In particular, we will use the following two results.

Fact 30 *If D_1 and D_2 are irreducible right denominators of F, then the Hermite column basis of D_1 equals the Hermite column basis of D_2.*

Fact 31 *Let D be an irreducible right denominator of F, and $M \in K[x]^{m \times m}$. Then FM is over $K[x]$ if and only if M is a right multiple of D.*

Suppose we have a nonsingular right multiple $M \in K[x]^{m \times m}$ of an irreducible right denominator of F. Then F admits the right fraction description $F = (FM)(M)^{-1}$. An irreducible right denominator of F can be computed from FM and M as follows. Let G be a right gcd (e.g., the Hermite row basis) of StackMatrix$(FM, M) \in K[x]^{n \times m}$. Then StackMatrix$(FM, M)G^{-1} =$ StackMatrix(FMG^{-1}, MG^{-1}) is also over $K[x]$, and has Hermite row basis I_m. Then $F = (FMG^{-1})(MG^{-1})^{-1}$, and MG^{-1} is by definition an irreducible right denominator of F. This gives the following well known recipe.

Fact 32 *Let $M \in K[x]^{m \times m}$ be nonsingular and such that FM is over $K[x]$. Then MG^{-1} is an irreducible right denominator of F, where G is any row basis of* StackMatrix(FM, M).

13 Trailing Hermite basis

Let m satisfy $1 \leq m \leq n$, and throughout this section, let

- $A \in K[x]^{n \times n}$ be nonsingular,
- $B \in K[x]^{n \times m}$ the the last m columns of I_n,
- $T \in K[x]^{m \times m}$ be the trailing submatrix of the Hermite column basis of A.

This section presents Algorithm 8 (`TrailingHermite`) for computing T . The algorithm is based on the observation that the following matrix is unimodular: $A^{-1}H = [\,* \mid A^{-1}BT\,]$. It follows that $A^{-1}BT$ is over $K[x]$ and that I_m is a left multiple of $A^{-1}BT$. This gives the following.

Lemma 33 *T is an irreducible right denominator of $A^{-1}B$.*

In particular, T is the Hermite column basis of any other irreducible right denominator of $A^{-1}B$ (Fact 30). Suppose we are given a nonsingular $M \in K[x]^{m\times m}$ such that $A^{-1}BM$ is over $K[x]$. Then Fact 32 gives a method to compute an irreducible right denominator of $A^{-1}B$ from $A^{-1}BM$ and M. Unfortunately, $A^{-1}BM \in K[x]^{n\times m}$ may have large degree (i.e., $\deg A^{-1}BM \leq (n-1)\deg A + \deg B + \deg M$) compared to M and T, leading to a bad complexity for the row basis computation. Our algorithm avoids this by using high-order lifting to computing a matrix $C \in K[x]^{n\times m}$, with $\deg C < \deg M$, and such that $(C)(M^{-1})$ and $(A^{-1}BM)(M^{-1})$ have the same irreducible right denominators.

Algorithm 8 `TrailingHermite`$[X](A, M, m)$
Input: $A \in K[x]^{n\times n}$ and a nonsingular $M \in K[x]^{m\times m}$.
Output: The trailing $m \times m$ submatrix T of the Hermite column basis of A in case M is a right multiple of T, otherwise fail.
Condition: $X \perp \det A$ and $d = \deg X \geq \deg A$.

(1) $B :=$ the last m columns of I_n;
$C :=$ `IntegralityCert`$[X](A, B, M)$;
if $C =$ fail **then return** fail **fi**;
(2) $E :=$ HermiteRowBasis(StackMatrix(C, M));
$D :=$ HermiteColumnBasis(ME^{-1});
return D

We now prove correctness. By the specification of Algorithm 7 (`IntegralityCert`), phase 1 will not return fail if and only if $A^{-1}BM$ is integral. By Fact 31 and Lemma 33, $(A^{-1}B)M$ is integral if and only if M is right multiple of T.

Suppose that the algorithm does not return fail. Let G be the Hermite row basis of StackMatrix$(A^{-1}BM, M)$. Then $A^{-1}BMG^{-1}$ and MG^{-1} are over $K[x]$. Let E be as computed in phase 2. Then CE^{-1} and ME^{-1} are over $K[x]$. By Fact 32, MG^{-1} is an irreducible right denominator of $A^{-1}B$, while ME^{-1} is an irreducible right denominator of CM^{-1}. Thus, we will be done if we show that $A^{-1}B$ and ME^{-1} are right multiples of each other (Fact 28).

For some h, Corollary 20 gives that

$$A^{-1}BM = \mathrm{Trunc}(A^{-1}B, h)M + CX^h. \tag{22}$$

On the one hand, both ME^{-1} and CE^{-1} are over $K[x]$. Post-multiplying both sides of (22) by E^{-1} shows that $A^{-1}BME^{-1}$ must be over $K[x]$ also. But then ME^{-1} is a right multiple of MG^{-1} (Fact 31). On the other hand, both $A^{-1}BMG^{-1}$ and MG^{-1} are over $K[x]$. Post-multiplying (22) by G^{-1} shows that CG^{-1} must be over $K[x]$ also. But then MG^{-1} is a right multiple of ME^{-1} (Fact 31).

Theorem 34 *Algorithm 8* (`TrailingHermite`) *is correct.*

We will not estimate the complexity of Algorithm 8 (`TrailingHermite`). A potential problem is that the Hermite row and column basis computations in phase 2 may have too high complexity, even if M has small degree. (The known algorithms for reducing Hermite form computation to matrix multiplication work modulo the determinant and have a complexity which depends on $\deg \det M$ rather than $\deg M$.) Instead, the next section presents a modification of the algorithm which computes directly the Smith form of T, avoiding any explicit Hermite basis computations.

14 Smith of trailing Hermite basis

Recall the definition of the Smith form: corresponding to any full column rank matrix $A \in K[x]^{n\times m}$ are unimodular matrices $U \in K[x]^{n\times n}$ and $V \in K[x]^{m\times m}$ such that $UAV = \text{Smith}(A) = \text{StackMatrix}(\text{PrincipalSmith}(A), 0)$, with $\text{PrincipalSmith}(A) = \text{Diagonal}(s_1, s_2, \ldots, s_m)$, each s_i monic, and s_i dividing s_{i+1} for $1 \le i \le m-1$.

Let m satisfy $1 \le m \le n$, and throughout this section, let

- $A \in K[x]^{n\times n}$ be nonsingular,
- $B \in K[x]^{n\times m}$ be the last m columns of I_n,
- $T \in K[x]^{m\times m}$ be trailing submatrix of the Hermite column basis of A, and
- $S \in K[x]^{m\times m}$ be the Smith form of T.

Algorithm 9 (`SmithOfTrailingHermite`) is a simple modification of Algorithm 8 (`TrailingHermite`).

Algorithm 9 `SmithOfTrailingHermite`$[X](A, s, m)$
Input: $A \in K[x]^{n\times n}$ and a nonzero $s \in K[x]$.
Output: The Smith form S of the trailing $m \times m$ submatrix T of the Hermite column basis of A in case sI_m is a right multiple of T, otherwise fail.
Condition: $X \perp \det A$ and $d = \deg X \ge \deg A$.

(1) $B :=$ the last m columns of I_n;
$C :=$ `IntegralityCert`$[X](A, B, sI_m)$;

if $C = \text{fail}$ **then return** fail **fi**;
(2) $\bar{E} := \text{PrincipalSmith}(\text{StackMatrix}(C, sI_m))$;
$\bar{D} := \text{Smith}((sI_m)\bar{E}^{-1})$;
return D

We now prove correctness. Phase 1 is identical to Algorithm 8 (`TrailingHermite`): fail will not be returned if and only if sI_m is a right multiple of T. Assume phase 1 does not fail, and let

$$E := \text{HermiteRowBasis}(\text{StackMatrix}(C, sI_m)).$$

Then $S = \text{Smith}((sI_m)E^{-1})$, since the Hermite column basis of $(sI_m)E^{-1}$ is equal to T. The key idea of phase 2 is to note that the Smith and inverse computation commute. This allows us to avoid the computation of the Hermite basis E. Let U and V be unimodular matrices such that UEV is in Smith form.

$$\begin{aligned} S &= \text{Smith}((sI_m)E^{-1}) \\ &= \text{Smith}(V^{-1}((sI_m)E^{-1})U^{-1}) \\ &= \text{Smith}((sI_m)V^{-1}E^{-1}U^{-1}) \\ &= \text{Smith}((sI_m)(\text{Smith}(E))^{-1}) \\ &= \text{Smith}((sI_m)\bar{E}^{-1}). \end{aligned}$$

We have shown that the algorithm is correct.

The cost of phase 1 is bounded by Proposition 21. Note that $\deg C < \deg s$ (Theorem 19). The initial Smith form in phase 2 can be computed with $O((n/m)\overline{\mathsf{MM}}(m, \deg s))$ field operations by working modulo s, i.e., over the principal ideal ring $R = K[x]/(s)$. First embed $\text{StackMatrix}(C, sI_m)$ into R, then compute an upper echelon form, and finally transform an $m \times m$ matrix to Smith form over R. The resulting Smith form over R, considered as a matrix over $K[x]$, will be as desired after replacing zero diagonal entries by s. For details and algorithm we refer to (Storjohann, 2000, Chapters 3 and 7).

Proposition 35 *Algorithm 9* (`SmithOfTrailingHermite`) *is correct. If m and $m \times (\deg s)/d$ are $O(n)$, then the cost of the algorithm is:*

- $O((\log n)\mathsf{MM}(n,d) + \overline{\mathsf{MM}}(n,d) + (n/m)\overline{\mathsf{MM}}(m,nd/m))$ *field operations, or*
- $O((\log n)\mathsf{MM}(n)\mathsf{B}(d))$ *field operations, assuming* $\mathsf{B}(n) = O(\mathsf{MM}(n)/n)$ *and* $n^{2+\gamma} = O(\mathsf{MM}(n))$ *for some positive* γ.

Worked example

The essential idea used in the last two sections carries over to the case of integer matrices with no modification. Specifically, let

- $A \in \mathbb{Z}^{n \times n}$ be nonsingular,
- $B \in \mathbb{Z}^{n \times m}$,
- $M \in \mathbb{Z}^{m \times m}$ be nonsingular.
- C be an integrality certificate for $A^{-1}BM$ (cf. Theorem 27).

Then the right matrix fractions $(A^{-1}BM)(M^{-1})$ and $(C)(M^{-1})$ have irreducible denominators which are right multiples of each other. The key point is that $\|C\|_\infty \leq (m/2)\|M\|_\infty$ (Theorem 27) even though $\|A^{-1}BM\|_\infty$ may be large.

For example, the matrix

$$A = \begin{bmatrix} -28 & -11 & -56 & -39 \\ -5 & 42 & -10 & 37 \\ 22 & -44 & -25 & 44 \\ -32 & 3 & 38 & 46 \end{bmatrix} \quad \text{has Hermite basis} \quad H = \begin{bmatrix} 1 & & & \\ 220 & 1231 & & \\ 0 & 2 & 3 & \\ 379 & 670 & 3792 & 3969 \end{bmatrix},$$

and the trailing 2×2 submatrix

$$T = \begin{bmatrix} 3 & \\ 3792 & 3969 \end{bmatrix} \quad \text{of } H \text{ has Smith form} \quad S = \begin{bmatrix} 3 & \\ & 3969 \end{bmatrix}.$$

Let $s = 3969$, and let C be the integrality certificate shown in (21). Then

$$\left[\begin{array}{cc} 1717 & 500 \\ -1168 & -1724 \\ 542 & -1112 \\ 1168 & 1724 \\ \hline 3969 & \\ & 3969 \end{array}\right] \quad \text{has principal Smith form} \quad \bar{D} = \begin{bmatrix} 1 & \\ & 1323 \end{bmatrix}.$$

Note that the Smith form of $(sI_2)\bar{D}^{-1}$ is S.

15 Determinant reduction

Let $A \in K[x]^{n\times n}$ be nonsingular. Recall that the Hermite row basis of A has the shape

$$H = \begin{bmatrix} h_1 & h_{12} & \cdots & h_{1n} \\ & h_2 & \cdots & h_{2n} \\ & & \ddots & \vdots \\ & & & h_n \end{bmatrix} \in K[x]^{n\times n},$$

and that $\det A = c \det H$ for a nonzero constant polynomial c.

Algorithm 10 (`DetReduction`) computes a matrix B, obtained from A by replacing the last column, such that the last diagonal entry in the Hermite row basis of B is one. The algorithm is thus named because $\det B = (\det A)/h_n$, where h_n is the trailing diagonal entry in the Hermite row basis of A.

A key step in the algorithm is to solve an instance of the extended gcd problem. For this we use the following result.

Lemma 36 *Given a row vector $w \in K[x]^{n\times 1}$, a column vector $b \in K[x]^{1\times n}$ such that $\deg b \leq \deg w$, and $wb = \gcd(w[1], w[2], \ldots, w[n])$, can be computed with $O(n\mathsf{B}(\deg w))$ field operations.*

An algorithm supporting the running time estimate of Lemma 36 is given in (Storjohann, 2000, Corollary 6.5).

Algorithm 10 `DetReduction`$[X](A)$
Input: $A \in K[x]^{n\times n}$.
Output: $B \in K[x]^{n\times n}$, with B equal to A except for possibly the last column, $\deg B \leq \deg A$, and last diagonal entry in the Hermite row basis of B equal to one.
Condition: $X \perp \det A$ and $d = \deg X \geq \deg A$.

(1) $(\bar{w}, h) :=$ `RationalSol`$[X](\text{Transpose}(A), \text{Column}(I_n, n))$;
$w := \text{Transpose}(\bar{w})$;
$P :=$ a permutation such that the last entry of wP has maximal degree;
$b :=$ an element of $K[x]^{n\times 1}$ such that $wPb = 1$, $\deg b \leq \deg w$;

(2) $v := \text{Column}(I_n - P^{-1}A, n)$;
$s := h + wPv$;
$(y, g) := \texttt{RationalSol}[X](A, P(sb - v))$;
$q := \text{Left}[sg](y, 1)$;
$q[n] := 0$;
return a copy of A except with last column replaced by $Pb - Aq$

We now explain the algorithm and prove correctness. Let w, P, and b, be as computed in phase 1. Let us assume, without loss of generality, that $P = I_n$. Then $\{w\}$ is a basis for the left kernel (over $K[x]$) of the first $n-1$ columns of A. The next fact follows.

Fact 37 *Assume B is nonsingular and equal to A except for possibly the last column. Then the unimodular transforming matrix which transforms B to Hermite row basis has last row equal to a scalar multiple of w.*

By construction of b in phase 1, the matrix obtained from A by replacing the last column with b (cf. the matrix on the left of (23)) will have Hermite row basis with trailing diagonal entry one. The problem is that $\deg b$ may be as large as $\deg w$, and $\deg w \leq (n-1)\deg A$. Phase 2 applies lattice reduction: the first $n-1$ columns of A are used to reduce the degree of b.

Let $s, g \in K[x]$ and $v, y, q \in K[x]^{n\times 1}$ be as computed in phase 2. Then $(I_n + v(w/h))A$ is equal to A with the last column replaced by $\text{Column}(I_n, n)$, and $((I_n + v(w/h))A)^{-1} = A^{-1}(I_n - (1/s)vw)$. The vector y is the unique solution to $(I_n - v(w/h))Ay = sgb$. Let $\bar{b}, \bar{y}, \bar{q} \in K[x]^{(n-1)\times 1}$ be the principal subvectors of b, y, q, and let $\bar{A} \in K[x]^{(n-1)\times(n-1)}$ be the principal submatrix of A. Because the last column of $(I - v(w/h))A$ is equal to $\text{Column}(I_n, n)$, we also have $\bar{A}\bar{y} = sg\bar{b}$. The vector q is a polynomial approximation to the rational vector $y/(sg)$ in the following sense: $y/(sg) = q + r/(sg)$ for some $r \in K[x]$ with $\deg r < \deg sg$.

$$\left[\begin{array}{c|c} \bar{A} & \bar{b} \\ \hline \bar{a} & b_n \end{array}\right] \left[\begin{array}{c|c} I_{n-1} & -\bar{q} \\ \hline & 1 \end{array}\right] = \left[\begin{array}{c|c} \bar{A} & \bar{b} - A\bar{q} \\ \hline \bar{a} & b_n - \bar{a}\bar{q} \end{array}\right] \tag{23}$$

It follows that $\bar{b} - \bar{A}\bar{q}$, which is equal to $\bar{A}\bar{r}/(sg)$, has degree strictly less than $\deg \bar{A}$. Since w is a vector in the left kernel of the first $n-1$ columns of A, and $q[n] = 0$, we have $w(b - Aq) = wb$. Since $wb = 1$, we have $w[n]\,(b - Aq)[n] = 1 - \sum_{i=1}^{n-1} w[i]\,(b - Aq)[i]$. By assumption, $\deg w[n] \geq \deg w[i]$ for $1 \leq i \leq n-1$. It follows that $\deg(b - Aq) \leq \max((\deg A) - 1, 0)$.

Corollary 16 bounds the cost of the two calls to Algorithm 5 (`RationalSol`). The cost of converting among X-adic, x-adic and (sg)-adic representations is bounded by $O(n\,\mathsf{B}(nd))$ field operations.

Proposition 38 *Algorithm 10* (`DetReduction`) *is correct. The cost of the algorithm is:*

- $O((\log n)\mathsf{MM}(n,d) + \overline{\mathsf{MM}}(n,d) + n\,\mathsf{B}(nd))$ *field operations, or*
- $O((\log n)\mathsf{MM}(n)\mathsf{B}(d))$ *field operations, assuming* $\mathsf{B}(n) = O(\mathsf{MM}(n)/n)$ *and* $n^{2+\gamma} = O(\mathsf{MM}(n))$ *for some positive* γ.

Worked example

The same determinant reduction method is applicable to the case of integer matrices. Consider the matrix A with Hermite row basis H.

$$A = \begin{bmatrix} -66 & -65 & 20 & -90 & 30 \\ 55 & 5 & -7 & -21 & 62 \\ 68 & 66 & 16 & -56 & -79 \\ 13 & -41 & -62 & -50 & 28 \\ 26 & -36 & -34 & -8 & -71 \end{bmatrix}, \quad H = \begin{bmatrix} 1 & 0 & 0 & 10 & 260246748 \\ & 1 & 0 & 2 & 292062707 \\ & & 1 & 7 & 244095302 \\ & & & 14 & 342954195 \\ & & & & 344319363 \end{bmatrix}.$$

An extended gcd computation gives $b = \begin{bmatrix} 779244 & 46649 & 46649 & 0 & 0 \end{bmatrix}$ such that $\mathrm{Row}(HA^{-1}, n)b = 1$. In the integer case, we compute q to be the integer vector such that each entry of $\bar{A}^{-1}\bar{b} - \bar{q}$ has magnitude < 1. The matrix obtained from A by replacing the last column with $b - Aq$ is

$$\begin{bmatrix} -66 & -65 & 20 & -90 & 3 \\ 55 & 5 & -7 & -21 & 46 \\ 68 & 66 & 16 & -56 & 79 \\ 13 & -41 & -62 & -50 & -15 \\ 26 & -36 & -34 & -8 & 2 \end{bmatrix}, \text{ with Hermite row basis } \begin{bmatrix} 1 & 0 & 0 & 10 & 0 \\ & 1 & 0 & 2 & 0 \\ & & 1 & 7 & 0 \\ & & & 14 & 0 \\ & & & & 1 \end{bmatrix}.$$

16 Partial Smith form

Let $A \in K[x]^{n \times n}$ be nonsingular. Let k and m be given, $1 \leq m \leq k \leq n-1$, and throughout this section, let

- $A = \left[\begin{array}{c|c|c} A_{11} & A_{12} & A_{13} \\ \hline A_{21} & A_{22} & A_{23} \\ \hline A_{31} & A_{32} & A_{33} \end{array}\right]$ where A_{11} is $k \times k$, and A_{22} is 1×1,
- H be the Hermite column basis of $[\, A_{11} \,|\, A_{12} \,]$,
- T be the trailing $m \times m$ submatrix of H,
- S be the Smith form of T, and
- the Hermite column basis of A be $\left[\begin{array}{c|c} H_{11} & \\ \hline H_{21} & H_{22} \end{array}\right]$ where H_{11} is $k \times k$.

This section presents an algorithm to compute S. Our eventual goal is to compute the entire Smith form of A. The algorithm in the next section will accomplish this by repeatedly applying the algorithm of this section to compute S as defined above for various choices of k and m. Note that S is not necessarily a submatrix of the Smith form of A. What is sufficient for the algorithm of the next section is that the following conditions (C1) and (C2) are satisfied:

- (C1) $H = H_{11}$.
- (C2) $\text{Smith}(A) = \text{Smith}(\text{Diagonal}(H_{11}, H_{22}))$.
- (C3) $\text{Smith}(A) = \text{Diagonal}(\text{Smith}(H_{11}), \text{Smith}(H_{22}))$.

Lemma 39 *(C3) implies (C2).*

Lemma 39 follows from the definition and uniqueness of the Smith form. Normally, these conditions may not hold. However, preconditioning techniques exist for transforming a nonsingular input matrix in $K[x]^{n \times n}$ to new matrix A which has the same Smith form, and which satisfies these conditions with high probability for all $1 \leq m \leq k \leq n-1$, see §18. For a given m and k, the algorithm here will fail if conditions (C1) and (C2) do not hold, and will not fail if (C1) and (C3) (and some additional conditions) do hold.

Define B, C, and D with the following conformal block decomposition:

$$\left[\begin{array}{c|c} B & C \\ \hline D & * \end{array}\right] = \left[\begin{array}{cc|c} A_{11} & A_{12} & A_{13} \\ A_{21} & A_{22} & A_{23} \\ \hline A_{21} & A_{22} & A_{23} \\ \hline A_{31} & A_{32} & A_{33} \end{array}\right] \in K[x]^{(n+1)\times n}, \tag{24}$$

so that B is $(k+1) \times (k+1)$, and the last row of C is zero. Note that the matrix in (24) is obtained from A by repeating row $k+1$.

Algorithm 11 `PartialSmith`$[X](A, s, k, m)$
Input: $A \in K[x]^{n\times n}$, nonzero $s \in K[x]$, $1 \le m \le k < n$.
Note: Let T, S, H_{11}, H_{21}, B, C, and D be as defined above.
Output: S or fail. Fail will be returned if (C1) and (C2) do not hold. Fail will not be returned if (C2) and (C3) hold, $X \perp \det B$, and sI_m is a right multiple of T.
Condition: $d = \deg X \ge \deg A$.

(1) **if** $X \not\perp \det B$ **then return** fail **fi**;
$R :=$ Transpose(`DetReduction`$[X]$(Transpose(B)));
(2) **if** `IntegralityCert`$[X](R, C, I) =$ fail **then return** fail **fi**;
(3) **if** `IntegralityCert`$[X]$(Transpose(R), Transpose(D), $I) =$ fail **then return** fail
fi;
(4) $\bar{S} :=$ `SmithOfTrailingHermite`$[X](R, s, m+1)$;
if $\bar{S} =$ fail **then return** fail **fi**;
$S :=$ the trailing $m \times m$ submatrix of $\bar{S}$;
return S

We now prove correctness. Assume phase 1 does not fail. Then R is identical to B except for possibly the last row (row $k+1$).

Phase 2 assays if $R^{-1}C$ is integral. Let $V \in K[x]^{(k+1)\times(k+1)}$ be the unimodular matrix such that RV is the Hermite column basis of R. Then

$$\overset{R}{\left[\begin{array}{c|c} A_{11} & A_{12} \\ \hline R_{11} & R_{12} \end{array}\right]} \overset{V}{\left[\begin{array}{c|c} V_{11} & V_{12} \\ \hline V_{21} & V_{22} \end{array}\right]} = \left[\begin{array}{c|c} H & \\ \hline & I_1 \end{array}\right].$$

$R^{-1}C$ is integral $\Leftrightarrow$ $V^{-1}R^{-1}C$ is integral $\Leftrightarrow$ $H^{-1}A_{13}$ is integral $\Leftrightarrow$ $H = H_{11}$. This shows that Phase 2 does not return fail if and only if $H = H_{11}$.

So far, we have established that

$$\left[\begin{array}{c|c|c} A_{11} & A_{12} & A_{13} \\ \hline R_{21} & R_{12} & A_{23} \\ \hline A_{21} & A_{12} & A_{23} \\ \hline A_{31} & A_{32} & A_{33} \end{array}\right] \left[\begin{array}{c|c|c} V_{11} & V_{12} & \\ \hline V_{21} & V_{12} & \\ \hline & & I \end{array}\right] \left[\begin{array}{c|c|c} I_k & & -H_{11}^{-1}A_{13} \\ \hline & I_1 & \\ \hline & & I \end{array}\right] = \left[\begin{array}{c|c|c} H_{11} & & \\ \hline & I_1 & Q_{13} \\ \hline Q_{11} & Q_{12} & Q_{13} \\ \hline Q_{21} & Q_{22} & Q_{23} \end{array}\right], \quad (25)$$

where the Q_{**} are new labels. Removing row $k+1$ gives

$$\left[\begin{array}{c|c|c} A_{11} & A_{12} & A_{13} \\ \hline A_{21} & A_{12} & A_{23} \\ \hline A_{31} & A_{32} & A_{33} \end{array}\right] \left[\begin{array}{c|c|c} V_{11} & V_{12} & \\ \hline V_{21} & V_{12} & \\ \hline & & I \end{array}\right] \left[\begin{array}{c|c|c} I_k & & -H^{-1}A_{13} \\ \hline & I_1 & \\ \hline & & I \end{array}\right] = \left[\begin{array}{c|c|c} H_{11} & & \\ \hline Q_{11} & Q_{12} & Q_{13} \\ \hline Q_{21} & Q_{22} & Q_{23} \end{array}\right]. \quad (26)$$

Considering (26) shows that H_{21} is equal to the Hermite column basis of

$$\left[\begin{array}{c|c} Q_{12} & Q_{13} \\ \hline Q_{22} & Q_{23} \end{array}\right].$$

Phase 3 does not return fail if and only if DR^{-1} is integral. Note that DR^{-1} is integral if and only if $(DV)(RV)^{-1}$ is integral. Considering (25) now shows that DR^{-1} is integral if and only if $\mathrm{StackMatrix}(Q_{11}, Q_{21})H_{11}^{-1}$ is integral, in which case $\mathrm{Smith}(A) = \mathrm{Smith}(\mathrm{Diagonal}(H_{11}, H_{12}))$. At this point the argument splits. On the one hand, we have just shown shows that if phase 3 does not return fail, then $\mathrm{Smith}(A) = \mathrm{Smith}(\mathrm{Diagonal}(H_{11}, H_{12}))$. On the other hand, suppose $\mathrm{Smith}(A) = \mathrm{Diagonal}(\mathrm{Smith}(H_{11}), \mathrm{Smith}(H_{12}))$. Then the definition and uniqueness of the Smith form imply that $\mathrm{StackMatrix}(Q_{11}, Q_{21})H_{11}^{-1}$ is integral, in which case phase 3 does not return fail.

Finally, consider phase 4. By construction, the trailing $(m+1) \times (m+1)$ submatrix of the Hermite column basis of R is equal to $\mathrm{Diagonal}(T, I_1)$. Now note that $\mathrm{Smith}(\mathrm{Diagonal}(T, I_1)) = \mathrm{Diagonal}(I_1, \mathrm{Smith}(T))$. By the specification of Algorithm 9 (`SmithOfTrailingHermite`), phase 4 does not return fail if and only if sI_m is a multiple of T.

Proposition 40 *Algorithm 11* (`PartialSmith`) *is correct. If m and $m \times (\deg s)/d$ are $O(n)$, then the cost of the algorithm is:*

- $O((\log n)\mathsf{MM}(n,d) + \overline{\mathsf{MM}}(n,d) + (n/m)\overline{\mathsf{MM}}(m, nd/m) + n\,\mathsf{B}(nd))$ *field operations, or*

- $O((\log n)\mathsf{MM}(n)\mathsf{B}(d))$ *field operations, assuming* $\mathsf{B}(n) = O(\mathsf{MM}(n)/n)$ *and* $n^{2+\gamma} = O(\mathsf{MM}(n))$ *for some positive* γ.

17 Smith form computation

Let $A \in K[x]^{n\times n}$ be nonsingular. We present an algorithm to compute the Smith form of A. Write the Hermite column basis H of A using a block decomposition as

$$H = \begin{bmatrix} H_{i-1} & & \\ \vdots & \ddots & \\ * & \cdots & H_1 \\ * & \cdots & * \quad H_0 \end{bmatrix},$$

where H_j is $2^j \times 2^j$ for $j = 0, 1, \ldots, i-2$, and the dimension of H_{i-1} is $\leq 2^{i-1}$.

Algorithm 12 $\mathtt{Smith}[X](A)$
Input: $A \in K[x]^{n\times n}$.
Output: The Smith form of A or fail. Fail will not be returned if and only if $\mathrm{Smith}(A) = \mathrm{Diagonal}(\mathrm{Smith}(H_{i-1}), \ldots, \mathrm{Smith}(H_0))$, and

- the Hermite column basis of the principal $k \times (k+1)$ submatrix of A is equal to the the Hermite column basis of the first k rows of A, and
- the principal $k \times k$ minor of A is $\perp X$,

for $k \in \{n-1, n-(1+2), n-(1+2+4), \ldots, n-(1+2+\cdots+2^{i-2}))\}$.
Condition: $X \perp \det A$ and $d = \deg X \geq \deg A$.

(1) $(*, h) := \mathtt{RationalSol}[X](A, \mathrm{Column}(I_n, n))$;
$S_0 := [h]$;
(2) $i := 0$;
$k := n-1$;
$m := \min(2, k)$;
for i **while** $k > 0$ **do**
$S_i := \mathtt{PartialSmith}[X](A, S_{i-1}[1,1], k, m)$;
if $S_i =$ fail **then return** fail **fi**;
$k := k - m$;
$m := \min(2m, k)$
od;
return $\mathrm{Diagonal}(\mathrm{S_{i-1}}, \mathrm{S_{i-2}}, \ldots, \mathrm{S_0})$

We now prove that if the algorithm does not fail, the result will be correct. Phase 1 computes $S_0 = \text{Smith}(H_0)$. Suppose phase 2 does not fail. Then $S_j = \text{Smith}(H_j)$ for $0 \leq j \leq i-1$. Since condition (C2) was satisfied for each call to Algorithm 11 (`PartialSmith`), we may conclude that $\text{Smith}(\text{Diagonal}(S_{i-1}, S_{i-2}, \ldots, S_0))$ is the Smith form of A. Finally, since $S_{j-1}[1,1]I$ is a right multiple of S_j for $1 \leq j \leq i-1$, we have that $\text{Diagonal}(S_{i-1}, S_{i-2}, \ldots, S_0)$ is already in Smith form.

Proposition 41 *Algorithm 12* (`Smith`) *is correct. Assuming n is a power of two, the cost of the algorithm is:*

- $O((\log n)^2 \mathsf{MM}(n,d) + \sum_{i=0}^{\log_2 n} 2^i \overline{\mathsf{MM}}(2^{-i}n, 2^i d))$ *field operations, or*
- $O((\log n)^2 \mathsf{MM}(n)\mathsf{B}(d))$ *field operations, assuming* $\mathsf{B}(n) = O(\mathsf{MM}(n)/n)$ *and* $n^{2+\gamma} = O(\mathsf{MM}(n))$ *for some positive* γ.

18 Conclusions

Most of our algorithms require as input a small degree X such that $X \perp \det A$. If $\#K$ is large enough, then X can be chosen to be $(x-a)^d$, for a randomly chosen $a \in K$, $d = \deg A$. Otherwise, X can be chosen to be the power of a small degree irreducible, see for example Shoup (1994). See (Mulders and Storjohann, 1999, Proof of Theorem 29) for more complete details of a method for choosing X randomly.

Algorithm 12 (`Smith`) requires that A satisfy some conditions. These are easy to achieve using the preconditioning technique as shown in Kaltofen et al. (1990). Choose nonsingular matrices U and V uniformly and randomly from $S^{n \times n}$, S a subset of K with $\#S \geq 4dn^4$. Then UAV will satisfy all required conditions with probability at least $1/2$ (see (Kaltofen et al., 1990, Algorithm 3.2) and (Storjohann and Labahn, 1995, Algorithm REDUCE)). If $\#K$ is too small, we can work over an algebraic extension field, but this will cause cost estimates to increase by a polylogarithmic factor.

A key idea in this paper is the use of high order lifting to efficiently certify integrality. Without this technique, many of the algorithms we propose would be Monte Carlo instead of Las Vegas.

The main task remaining is to extend the results here to the case of integer matrices. The key ideas of Sections 11—17 carry over easily. The main difficulties to be solved are to achieve a suitable preconditioning for the input matrix of the Smith form computation, and to get analogous versions of the lifting algorithms in Sections 6, 9 and 10. To solve the first difficulty the results in Eberly et al. (2000) and Mulders and Storjohann (2003) should prove useful.

The crux of the second difficulty is that the absolute value norm over $\mathbb{Z}$, unlike the degree norm over $K[x]$, is Archimedean; because integer addition has carries, the analogue of Lemma 1 does not hold. One solution to this is to do computation in a shifted number system. We will present this in a future paper.

References

Dixon, J. D., 1982. Exact solution of linear equations using p-adic expansions. Numer. Math. 40, 137–141.

Eberly, W., Giesbrecht, M., Villard, G., 2000. Computing the Smith form of a dense integer matrix. In: Proc. 31st Ann. IEEE Symp. Foundations of Computer Science. pp. 675–685.

Gathen, J. v. z., Gerhard, J., 1999. Modern Computer Algebra. Cambridge University Press.

Giorgi, P., Jeannerod, C.-P., Villard, G., 2003. On the complexity of polynomial matrix computations. Research Report 2003-2. Laboratoire LIP, ENS Lyon, France.

Hafner, J. L., McCurley, K. S., Dec. 1991. Asymptotically fast triangularization of matrices over rings. SIAM Journal of Computing 20 (6), 1068–1083.

Kailath, T., 1980. Linear Systems. Prentice Hall, Englewood Cliffs, N.J.

Kaltofen, E., Krishnamoorthy, M. S., Saunders, B. D., 1990. Parallel algorithms for matrix normal forms. Linear Algebra and its Applications 136, 189–208.

Karatsuba, A., Ofman, Y., 1963. Multiplication of multidigit numbers on automata. Soviet Physics-Doklady 7, 595–596.

Moenck, R. T., Carter, J. H., 1979. Approximate algorithms to derive exact solutions to systems of linear equations. In: Proc.EUROSAM '79, volume 72 of *Lecture Notes in Compute Science*. Springer-Verlag, Berlin-Heidelberg-New York, pp. 65–72.

Mulders, T., Storjohann, A., 1999. Diophantine linear system solving. In: Dooley, S. (Ed.), Proc. Int'l. Symp. on Symbolic and Algebraic Computation: ISSAC '99. ACM Press, New York, pp. 281–288.

Mulders, T., Storjohann, A., 2000. Rational solutions of singular linear systems. In: Traverso, C. (Ed.), Proc. Int'l. Symp. on Symbolic and Algebraic Computation: ISSAC '00. ACM Press, New York, pp. 242–249.

Mulders, T., Storjohann, A., 2002. On lattice reduction for polynomial matrices. Journal of Symbolic Computation 35 (4), 377–401.

Mulders, T., Storjohann, A., 2003. Certified diophantine dense linear system solving. Journal of Symbolic Computation To appear.

Shoup, V., 1994. Fast construction of irreducible polynomials over finite fields. Journal of Symbolic Computation 17, 371–391.

Storjohann, A., 2000. Algorithms for matrix canonical forms. Ph.D. thesis,

Swiss Federal Institute of Technology, ETH–Zurich.

Storjohann, A., Labahn, G., 1995. Preconditioning of rectangular polynomial matrices for efficient Hermite normal form computation. In: Levelt, A. H. M. (Ed.), Proc. Int'l. Symp. on Symbolic and Algebraic Computation: ISSAC '95. ACM Press, New York, pp. 119–125.

Strassen, V., 1973. Vermeidung von Divisionen. J. reine angew. Math. 264, 182—202.

Villard, G., 1996. Computing Popov and Hermite forms of polynomial matrices. In: Lakshman, Y. N. (Ed.), Proc. Int'l. Symp. on Symbolic and Algebraic Computation: ISSAC '96. ACM Press, New York, pp. 251–258.