High-Order Lifting

[Extended Abstract]

Arne Storjohann
Ontario Research Centre for Computer Algebra
Department of Computer Science
University of Waterloo, Canada
astorjoh@scg.math.uwaterloo.ca

ABSTRACT

The well-known technique of adic-lifting for linear-system solution is studied. Some new methods are developed and applied to get algorithms for the following problems over the ring of univariate polynomials with coefficients from a field: rational system-solving, integrality certification and determinant/Smith-form computation. All algorithms are Las Vegas probabilistic.

1. INTRODUCTION

Let K be a field and $A \in \mathsf{K}[x]^{n\times n}$ and $B \in \mathsf{K}[x]^{n\times m}$. Suppose A is nonsingular and, moreover, that $\det A$ is relatively prime to a given X, $X \in \mathsf{K}[x]$. Then $A^{-1}B \in \mathsf{K}(x)^{n\times m}$ admits a unique X-adic series expansion

$$A^{-1}B = C_0 + C_1X + \cdots + \overbrace{C_hX^h + \cdots + C^{h+k}X^{h+k}}^{HX^h} + \cdots \quad (1)$$

where each $C_* \in \mathsf{K}[x]^{n\times m}$ has $\deg C_* < \deg X$. This paper 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 H for a given h and k as shown in (1). Another variation computes a collection of such segments for a given expansion. The main contribution of this paper is to demonstrate applications for high-order lifting. We get deterministic or Las Vegas solutions of many other computational problems. Let $B \in \mathsf{K}[x]^{n\times m}$ as above and $b \in \mathsf{K}[x]^{n\times 1}$ be given. The three main problems are:

Rational system solving Compute $A^{-1}b$.
Integrality certification Assay if $A^{-1}B$ is integral.
Determinant Compute the determinant/Smith-form of A.

Assuming $\deg b/\deg A$ and $m(1+\deg B/\deg A)$ are $O(n)$, the problems listed above are solved in an expected number of $O\tilde{}(n^\theta \deg A)$ field operations from K. Here, θ is the exponent for matrix multiplication (see below for cost model). These complexity results improve on previous results.

ISSAC 2002, July 7-10, 2002, Lille, France

Previous results

Consider first rational system solving. The currently best deterministic algorithm [10] has running time $O\tilde{}(n^3 \deg A)$. Restricting $\deg b = O(\deg A)$ and allowing randomization, the technique in [8] improves the exponent of n, but not down to θ. Our algorithm is based on adic-lifting [2, 6] and is probabilistic because a small-degree polynomial not dividing the determinant of A is chosen randomly.

Consider the integrality certification problem when $B = I_n$. A is unimodular precisely when A^{-1} is over $\mathsf{K}[x]$. Unimodularity can be tested by computing $\det A \bmod X$ for a randomly-chosen small-degree X; this gives a nearly-optimal $O\tilde{}(n^\theta + n^2 \deg A)$ Monte Carlo probabilistic algorithm. The $O\tilde{}(n^\theta \deg A)$ algorithm we give here is deterministic and nearly matches this running time.

The Hermite-form of A (and hence also the determinant) can be computed deterministically [9] in time $O(n^3(\deg A)^2)$. The Smith-form can be computed in the same time (Las Vegas) using the preconditioning of [4]. The computation of the determinant has been well studied, especially also in the case of integer matrices. We refer to [5] for a survey; the currently best result for integers extends to polynomials, giving an $O\tilde{}(n^{2.698} \deg A)$ Las Vegas algorithm.

Model of computation

By time we mean the number of required field operations from K on an algebraic RAM; the operations $+$, $-$, $\times$ and "divide by a nonzero" are considered as unit step operations. Let $O(d^{1+\epsilon})$ be the time to multiply degree d polynomials. Let $O(n^\theta)$ be the time to multiply two $n \times n$ matrices over a commutative ring with identity. We are going to assume that $2 < \theta \le 3$ and $0 < \epsilon \le 1$. Sometimes we will make the (eminently reasonable) assumption that $\epsilon \le \theta - 2$.

2. OUTLINE

Many of the results in this paper build upon previous results. Although peppered with examples, the rest of this paper necessarily adopts a more concise and technical style. Here we give a global outline and indicate the relationships between the sections. We also give abstract, intuitive descriptions of the key ideas and algorithms.

Adic-lifting for system solving

Sections 3 and 4 define notation and recall some basic facts about X-adic expansions of rational functions, including the recovery of such expansions using X-adic lifting. Consider (1) where B is a single column vector, say b, and both $\deg A, \deg b \leq \deg X$. Suppose our goal is to compute the expansion of $A^{-1}b$ up to order X^k. We can divide the problem into two similar subproblems. The first is to compute the expansion of $A^{-1}b$ up to order $X^{k/2}$.

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

The key idea of X-adic lifting is to replace of the "mod" in the last equivalence by introducing a "residue" $r_{k/2}$.

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

It may easily be verified by substitution that

$$r_{k/2} = (b - A(c_0 + c_1 X + \cdots + c_{k/2-1}X^{k/2-1}))/X^{k/2} \quad (3)$$

and, moreover, that $r_{k/2}$ lives in $\mathsf{K}[x]^{n\times 1}$ and has $\deg r_{k/2} < \deg X$. 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 get around this problem.

High-order components of matrix inverse

Section 5 gives our first high-order lifting algorithm. Consider (1) 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 is currently been computed. Normally, all coefficients of the expansion are computed up to order $X^{\Theta(n)}$ — this costs $O(n \times n^\theta)$ using $O(\log n)$ steps of quadratic X-adic lifting, see Figure 1. After the fourth step of lifting (which dominates the cost)

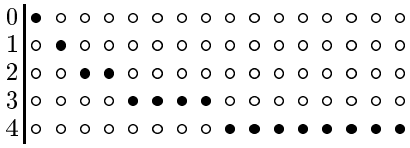
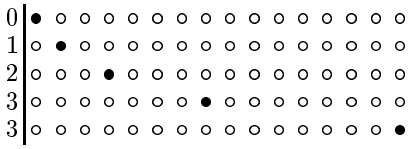


Figure 1: Quadratic lifting for $n = 4$

all initial sixteen 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, see 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 the inverse expansion are not recovered, the computation of the critical



(4)

Figure 2: High-order component lifting for $n = 4$

subset of high-order components has many applications (it's what this paper is about). The algorithm described in this section is used in almost all subsequent sections.

Series solution — small right hand side

Section 6 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. The key difference here is that we compute the residue $r_{k/2}$ shown in (3) without first solving the initial subproblem shown in (2). This is accomplished using $\Theta(1)$ matrix×vector products involving A and a particular high-order component of the inverse (and exploiting the observation that $r_{k/2}$ can be computed from A and $c_{k/2-1}$ alone). We now have

$$A^{-1}b = \left(A^{-1}\left[\begin{array}{c|c} b & r_{k/2} \end{array}\right]\right)\left[\begin{array}{cc} 1 & \\ & X^{k/2} \end{array}\right]$$

where the right hand side $\left[\begin{array}{c|c} b & r_{k/2} \end{array}\right]$ has column dimension two. 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 $O((\log n)n^\theta)$.

Series solution

Section 7 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 $b = b_0 + b_1 X + b_2 X^2 + \cdots + b_{nd-1}X^{nd-1}$. The algorithm encodes the "fat" vector b as an $n \times n$ matrix B with i'th column equal to b_{i-1}. The i-th column of B may be thought to be implicitly multiplied by X^{i-1}. For an $n \times n$ matrix C, a matrix×vector product Cb, $\deg b < nd$, can be now accomplished more efficiently as a matrix×matrix product CB, $\deg B < d$. Suppose our goal is to produce $A^{-1}B$ up to order X^n. Using $\Theta(1)$ matrix products, the algorithm produces a second matrix $\bar{B}$ such that the expansion of $A^{-1}B$ up to order X^n is equal to the expansion of $A^{-1}B$ up to order $X^{n/2}$ added to the expansion of $A^{-1}\bar{B}$ up to order $X^{n/2}$; the key observation is now to compute the desired result as the single expansion of $A^{-1}(B + \bar{B})$. Thus, the single matrix addition $B + \bar{B}$ allows us to recurse on only one instead of two problems. This technique is applied for order $X^{n/2}, X^{n/4}, X^{n/8}, \cdots$ yielding a series of $O(\log n)$ transformations on B using the high-order components of the expansion of A^{-1}. The overall cost in terms of n is $O((\log n)n^\theta)$.

High-order lifting

Section 8 gives a general algorithm for solving the high-order lifting problem: the recovery of H as shown in (1). 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 $\deg A \leq d$, $d = \deg X$. A running time of $O((\log n)n^\theta)$ is achieved for a wide range of the input paramaters m, k and $\deg b$. All that is required is that the paramaters m and $\{(\deg b)/d, k/d\}$ be balanced: both $m \times (\deg b)/d$ and $m \times k/d$ should be $O(n)$.

Further ideas

The discussion so far focused on techniques for rapidly computing adic-expansions. The primary contribution of this paper is to demonstrates applications to solving a variety of other problems.

Section 9 computes a high-order lift H as in (1) in order to solve a generalization of the integrality certification problem. Sections 10, 11 and 12 deal with determinant/Smith-form computation. Section 10 uses rational system-solving to transform A to an almost identical matrix but with potentially much smaller determinant — this is imperative to obtain the complexity result. Section 11 computes a high-order lift H as in (1) where B is chosen to be the trailing m columns of the identity matrix; this is then used to compute the trailing m diagonal entries of the Hermite-form. Section 12 puts all the pieces together and gives the complete algorithm for determinant/Smith-form.

Many results here extend directly to the more difficult case of integer matrices. For space reasons, it will be convenient to sometimes use illustrative examples with decimal expansions of integers instead of X-adic expansions of polynomials. Section 13 concludes and mentions something more about the integer case.

3. ADIC REPRESENTATION

Let l be nonnegative integer and $X \in \mathsf{K}[x]$ have degree greater than zero. By X-adic expansion of $a \in \mathsf{K}[x]$ we mean to write $a = a_0 + a_1X + a_2X^2 + \cdots + a_lX^l$, $\deg a_* < \deg X$. "Degree" will always mean degree in x. In other words, if $\deg X = d$ and a_l is nonzero, then $dl \le \deg a < d(l+1)$. The a_* are called coefficients of the X-adic expansion of a.

The ring $\mathsf{K}[x]$ has the usual arithmetic operations $\{+, -, \times\}$. Here we define three additional operations Left, Trunc and Inverse and gives some of their properties. These functions will implicitly be defined in terms of a proscribed X. Let $a \in \mathsf{K}[x]$ 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 $\text{Trunc}(a,k) = a_0 + a_1X + a_2X^2 + \cdots + a_{k-1}X^{k-1}$. If $a \perp X$, then $\text{Inverse}(a,k)$ denotes the unique $b \in \mathsf{K}[x]$ such that $b = \text{Trunc}(b,k)$ and $\text{Trunc}(ab,k) = \text{Trunc}(ba,k) = 1$.

All the above definitions above extend naturally to matrix polynomials. Just replace $a, q \in \mathsf{K}[x]$ with $A, Q \in \mathsf{K}[x]^{n\times m}$. The operation Inverse takes as input a square matrix A which has $\det A \perp X$.

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

LEMMA 1. *If* $\deg(\gamma) < \deg(X^k)$ *then*

$$\text{Left}(a+\gamma,k) = \text{Left}(a,k).$$

The next lemma observes that Left and Trunc commute.

LEMMA 2. *If* $l \le k$ *then*

$$\text{Left}(\text{Trunc}(a,k),l) = \text{Trunc}(\text{Left}(a,l),k-l).$$

Computation with X-adic polynomials

We are working over $\mathsf{K}[x]$ with the operations $\{+, -, \times, \text{Left}, \text{Trunc}, \text{Inverse}\}$. The cost of these operations will depend essentially on our choice of representation.

For $a \in \mathsf{K}[x]$ let k be minimal such that $a = \text{Trunc}(a,k)$. Then a can be stored as a list comprised of the first k coefficients of the X-adic expansion. Let $a, b \in \mathsf{K}[x]$ be nonzero with $\deg a \ge \deg b$. Then the X-adic expansion of $a+b$ or $a-b$ can be computed in $O(1+\min(\deg a, \deg b))$ field operations, that of ab in $O((1+\deg a)(1+\deg b)^\epsilon)$ field operations, and that of $\text{Inverse}(a,k)$ in $O((k \deg X)^{1+\epsilon})$ field operations. Operations Left, Trunc and multiplication by a power of X are free. For $Y \in \mathsf{K}[x]$, conversion from X-adic to Y-adic representation can be accomplished in $O((1+\deg a)^{1+\epsilon})$ field operations.

4. ADIC-LIFTING

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

$$A = \begin{bmatrix} 1 & 1 \\ x & 1 \end{bmatrix} \quad \text{then} \quad A^{-1} = \begin{bmatrix} \frac{1}{1-x} & \frac{1}{x-1} \\ \frac{x}{x-1} & \frac{1}{1-x} \end{bmatrix} \in \mathsf{K}(x)^{2\times 2}. \quad (5)$$

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 $A \in \mathsf{K}[x]^{n\times n}$ we have:

FACT 3. $\deg(\det A) \le n \deg(A)$.

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

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

Consider again A from (5). Since $\det A \perp x$, we can express each entry of A^{-1} as an infinite x-adic expansion.

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

More generally, let $A \in \mathsf{K}[x]^{n\times n}$ be nonsingular, $\det A \perp X$. Let the X-adic expansion of A^{-1} be

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

Note that each $*$ lives in $\mathsf{K}[x]^{n\times n}$ and has degrees of entries strictly less than $\deg X$. Thus, the indicated C has polynomial entries with degrees $< ld$. For a given $B \in \mathsf{K}[x]^{n\times m}$, let the X-adic expansion of $A^{-1}B$ be

$$\overbrace{* + *X + \cdots + *X^{k-1}}^{D} + \overbrace{*X^k + \cdots + *X^{k+l-1}}^{EX^k} + \cdots.$$

Suppose we have C and D. Then we can recover E using X-adic lifting:

THEOREM 5. $E = \text{Trunc}(C\,\text{Left}(B - AD, k), l)$.

5. HIGH-ORDER COMPONENTS

Let $A \in \mathsf{K}[x]^{n\times n}$ be nonsingular, $\det A \perp X$. In what follows, let $Z^{(i)} = \text{Inverse}(A, 2^i)$. In this section we show how to recover the high order components of the inverse of A: $E^{(i)} = \text{Left}(Z^{(i)}, 2^i - 2)$ for $i = 1, 2, \ldots, k$. To see more clearly what we are computing, write the X-adic expansion of A^{-1} as $C_0 + C_1X + C_2X^2 + \cdots$. Then

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

Starting with $Z^{(0)}$ we can recover $Z^{(1)}, Z^{(2)}, \ldots, Z^{(k)}$ using k steps of quadratic X-adic lifting in time $O(2^k n^\theta d^{1+\epsilon})$, $d = \deg X$. Algorithm `HighOrderComp` recovers only the high order components $E^{(*)}$ as shown above. The cost estimate of $O(kn^\theta d^{1+\epsilon})$ field operations for the algorithm is easy to derive.

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

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

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

$$L^{(j)} = C_{2^j-2} \quad (6)$$
$$H^{(j)} = C_{2^j-1} \quad (7)$$
$$E^{(j)} = C_{2^j-2} + XC_{2^j-1} \quad (8)$$

for $j = 1, 2, \ldots, k$. The base case $j = 1$ has already been established. That (8) follows from (6) and (7) is clear.

For $i > s$, quadratic X-adic lifting (a special case of Theorem 5) gives

$$\text{Left}(Z^{(i)}, 2^{i-1}) = \text{Trunc}(Z^{(i-1)}\text{Left}(I - AZ^{(i-1)}, 2^{i-1}), 2^{i-1})$$

while the loop computes

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

Our goal is to show (6) and (7) hold for $j = i$. It will be sufficient to show that (7) holds since the proof of (6) is analogous. In the proof we will use the following degree estimates, which follow from (7) and (8).

$$\deg(Z^{(j)} - X^{2^j-1}H^{(j)}) < \deg(X^{2^j-1}) \quad (9)$$
$$\deg(Z^{(j)} - X^{2^j-2}E^{(j)}) < \deg(X^{2^j-2}) \quad (10)$$

The next lemma assumes (by induction) that (9) holds for $j = i - 1$.

LEMMA 6. $R = (I - AZ^{(i-1)})/X^{2^{i-1}}$.

PROOF. Let $a = (I - AZ^{(i-1)})/X^{2^{i-1}}$ and choose $\gamma = -A\,\text{Left}(E^{(i-1)}, 1) - Xa$ so that $-A\,\text{Left}(E^{(i-1)}, 1) = Xa + \gamma$. Using (9) for $j = i - 1$ we may derive that $\deg\gamma < \deg A \leq \deg X$. Now use Lemma 1 to conclude that $\text{Left}(Xa + \gamma, 1) = a$. □

At this point we have shown that $S = \text{Left}(E^{(i-1)}R, 1)$ where R is as in Lemma 6. For any nonnegative y we have $S = \text{Left}(X^yE^{(i-1)}R, y + 1)$. Let $y = 2^{i-1} - 2$. The next lemma assumes (by induction) that (10) holds for $j = i - 1$.

LEMMA 7. $S = \text{Left}(Z^{(i-1)}R, y + 1)$.

PROOF. Let $a = Z^{(i-1)}R$ and $\gamma = X^yE^{(i-1)})R - a$ so that $a + \gamma = X^yE^{(i-1)}R$. Using $\deg(R) < d$ and (9) for $j = i - 1$ gives $\deg\gamma < \deg(X^{y+1})$. Now use Lemma 1. □

Lemmas 6, 7 and 2 now give (7) or $j = i$. The proof that (6) holds for $j = i$ is analogous. This ends the inductive proof of correctness of the algorithm. We have shown:

PROPOSITION 8. *Algorithm* `HighOrderComp` *is correct. The cost of the algorithm is* $O(kn^\theta d^{1+\epsilon})$ *field operations.*

Typical applications of the algorithm have $k = O(\log n)$.

6. SERIES SOLUTION — SMALL RHS

Let $A \in \mathsf{K}[x]^{n\times n}$ be nonsingular, $\det A \perp X$. Let $b \in \mathsf{K}[x]^{n\times 1}$. 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 `SeriesSolutionSmallRHS`$[X](A, b, k)$
Input: $A \in \mathsf{K}[x]^{n\times n}$, $b \in \mathsf{K}[x]^{n\times 1}$, $k \geq 2$
Output: $\text{Trunc}(\text{Inverse}(A, 2^k)\,b, 2^k)$
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. $B := [\ b \mid O\ ]$ where 0 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;
   $\bar{B} := \text{Left}(-A \text{ Trunc}(\text{Left}(E^{(i)}\bar{B}, 1), 1), 1)$;
   $\bar{B} := [\ O \mid \bar{B}\ ]$ where O is the $n \times 2^i$ zero matrix;
   $B := B + \bar{B}$;
   od;
   $B := \text{Trunc}(E^{(1)}B, 2)$;

3. # Let $B = [\ d_0 \mid 0 \mid d_2 \mid 0 \mid \cdots \mid d_{2^k-2} \mid 0\ ]$.
   $B := d_0 + d_2 X^2 + \cdots + d_{2^k-2} X^{2^k-2}$;
   return B

We now prove that the algorithm is correct. Let $[X](A, b, k)$ be valid input tuple. Let $Z^{(i)} = \text{Inverse}(A, 2^i)$ for $i > 0$. Phase 1 computes the high order components of $Z^{(k-1)}$ at a cost of $O(kn^\theta d^{1+\epsilon})$ field operations.

Now consider phase 2. The purpose of this phase is to compute all the coefficients of $\text{Trunc}(Z^{(k)}b, 2^k)$. We begin by giving an example when $k = 4$. A formal proof will follow. Let $r_j = (b - A\,\text{Trunc}(\text{Inverse}(A, j)b, j))/X^j$, $j \in \{0, 2, 4, \ldots, 14\}$. Then $r_0 = b$ and we claim that

$$\text{Trunc}(Z^{(4)}b, 16) = \sum_{j=0}^{16/2^i} \text{Trunc}(Z^{(i)} r_{16j/2^i}, 2^i) X^{16j/2^i}$$

for $i = 4, 3, 2, 1$. 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

$$B = [\ r_0 \ \ 0 \ \ 0 \ \ 0 \ \ 0 \ \ 0 \ \ 0 \ \ 0 \ \ 0 \ \ 0 \ \ 0 \ \ 0 \ \ 0 \ \ 0 \ \ 0 \ \ 0\ ].$$

The i'th column of B may be thought to be implicitly multiplied by X^{i-1}. After the loop completes with index i the matrix B looks like:

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

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. The last line of phase 2 does this to compute

$$B = [\ c_0 + c_1 X \ \ 0 \ \ c_2 + c_3 X \ \ 0 \ \ \cdots \ \ c_{14} + c_{15} X \ \ 0\ ]$$

where $c_0 + c_1 X + c_2 X^2 + \cdots$ is the X-adic expansion of $\text{Trunc}(Z^{(4)}b, 16)$.

Now we give a formal proof of the above. At the same time we estimate the complexity in terms of n, k and d. Let $f(i, m)$ be the cost of computing $\text{Trunc}(Z^{(k)}B, 2^i)$ for a given $B \in \mathsf{K}[x]^{n \times m}$, $\deg B \le d$. For $i > 1$, quadratic X-adic lifting gives the identity

$$\text{Trunc}(Z^{(i)}B, 2^i) = \text{Trunc}(Z^{(i-1)} [\ B \mid \bar{B}\ ], 2^{i-1}) \begin{bmatrix} 1 \\ X^{2^{i-1}} \end{bmatrix}$$

where

$$\bar{B} = \text{Left}(B - A\,\text{Trunc}(Z^{(i-1)}B, 2^{i-1}), 2^{i-1}). \qquad (11)$$

It is easy to derive that $\deg \bar{B} < \deg A \le d$. Thus we get

$$f(i, m) \le f(i-1, 2m) + \text{ cost of computing } \bar{B}.$$

Consider equation (11) for $\bar{B}$. We may consider $X^{2^i-2}E^{(i)}$ to be an approximation of $Z^{(i)}$. If we expand formula (11) for $\bar{B}$ we get the new formula

LEMMA 9. $\bar{B} = \text{Left}(-A\,\text{Trunc}(\text{Left}(E^{(i-1)}B, 1), 1), 1)$

When $i = 1$ we may compute $\bar{B} = \text{Trunc}(Z^{(1)}B, 2)$ directly since $E^{(1)} = Z^{(1)}$. Since the number of nonzero column in B is doubling each time, the last iteration of the loop dominates. The cost is $O((2^k/n)n^\theta d^{1+\epsilon})$ field operations if $2^k > n$. If $2^k \le n$ the cost is 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. We have shown:

PROPOSITION 10. *Algorithm* `SeriesSolutionSmallRHS` *is correct. The cost of the algorithm is* $O((k + 2^k/n)n^\theta d^{1+\epsilon})$ *field operations.*

7. SERIES SOLUTION

Let $A \in \mathsf{K}[x]^{n \times n}$ be nonsingular, $\det A \perp X$. Let $b \in \mathsf{K}[x]^{n \times m}$. 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. The algorithm here makes no assumption on the degree of b and allows b to have column dimension m, $m > 1$.

Algorithm `SeriesSolution`$[X](A, b, k)$
Input: $A \in \mathsf{K}[x]^{n \times n}$, $b \in \mathsf{K}[x]^{n \times m}$, $k \ge 2$
Output: $\text{Trunc}(\text{Inverse}(A, 2^k)b, 2^k)$
Condition: $X \perp \det A$ and $d = \deg X \ge \deg A$

1. $E^{(1)}, E^{(2)}, \ldots, E^{(k-1)} :=$ `HighOrderComp`$[X](A, k-1)$;

2. # Let X-adic expansion of b be $b_0 + b_1 X + b_2 X^2 + \cdots$.
   $B := [\ b_0 \mid b_1 \mid \cdots \mid b_{2^k-1}\ ]$;
   for i **from** $k-1$ **by** -1 **to** 1 **do**
   $\bar{B} :=$ the first $m2^k - m2^i$ columns of B;
   $\bar{B} := \text{Left}(-A\,\text{Trunc}(\text{Left}(E^{(i)}\bar{B}), 1), 1), 1)$;
   $\bar{B} := [\ O \mid \bar{B}\ ]$ where O is the $n \times m2^i$ zero matrix;
   $B := B + \bar{B}$;
   od;
   $B := \text{Trunc}(E^{(1)}B, 2)$;

3. # Let $B = [\ d_0 \mid d_1 \mid \cdots \mid d_{2^k-1}\ ]$.
   $B := d_0 + d_1 X + d_2 X^2 + \cdots + d_{2^k-2} X^{2^k-2}$
   $\quad + \text{Trunc}(d_{2^k-1}, 1) X^{2^k-1}$;
   return B

We now prove that the algorithm is correct. At the same time we estimate the complexity in terms of n, m, k and d. Recall that in Algorithm `SeriesSolutionSmallRHS` we assumed $m = 1$ so that m did not appear in the costs estimates there. Let $[X](A, b, k)$ be a valid input tuple. Let $Z^{(i)} = \text{Inverse}(A, 2^i)$ for $i > 0$. Phase 1 computes the high

order components of $Z^{(k-1)}$ at a cost of $O(kn^\theta d^{1+\epsilon})$ field operations.

Phase 2 is identical to the corresponding phase in Algorithm `SeriesSolutionSmallRHS` except that here we solve $m2^k$ systems in parallel. We need only observe that

$$\begin{aligned}\mathrm{Trunc}(Z^{(k)}b,k) &= \sum_{i=0}^{2^k-1}\mathrm{Trunc}(Z^{(k)}b_i\,X^i,k)\\ &= \sum_{i=0}^{2^k-1}X^i\,\mathrm{Trunc}(Z^{(k)}b_i,k-i)\end{aligned}$$

In other words, we use the identity $A^{-1}B+A^{-1}\hat{B}=A^{-1}(B+\hat{B})$. The cost of phase 2 bounded by $O((km2^k/n)n^\theta d^{1+\epsilon})$ field operations if $m2^k > n$. If $m2^k \le n$ the cost is is dominated by that of phase 1.

Phase 3 multiplies each column of B by the appropriate power of X and adds all the columns together. The cost of this phase is dominated by that of phase 2. We have shown:

PROPOSITION 11. *Algorithm* `SeriesSolution` *is correct. The cost of the algorithm is* $O((k(1+m2^k/n))n^\theta d^{1+\epsilon})$ *field operations.*

If we choose k such that $2^k d > 2(n-1)\deg A + \deg B$, then we can recover $A^{-1}b \in \mathsf{K}(x)^{n\times 1}$ using rational reconstruction. The rational reconstruction costs $O(n(nd)^{1+\epsilon})$. If $\theta - 2 \ge \epsilon$ (a reasonable assumption) then the rational reconstruction does not dominate. Noting that we can choose $k = O(\log n)$ if $\deg b = O(nd)$, we get

COROLLARY 12. *Let* $[X](A,b,*)$ *be a valid input tuple to Algorithm* `SeriesSolution`*, with* $b \in \mathsf{K}[x]^{n\times 1}$*. Assuming* $\epsilon \le \theta - 2$ *and* $(\deg b)/d = O(n)$*, the rational system solution* $A^{-1}b \in \mathsf{K}(x)^{n\times 1}$ *can be computed in* $O((\log n)n^\theta d^{1+\epsilon})$ *field operations.*

8. HIGH-ORDER LIFTING

Let $A \in \mathsf{K}[x]^{n\times n}$ be nonsingular, $\det A \perp X$. Let $B \in \mathsf{K}[x]^{n\times m}$. We present an algorithm to recover a contiguous segment $H = \mathrm{Left}(\mathrm{Trunc}(\mathrm{Inverse}(A,h+k)\,b,h+k),h)$ of coefficients from the X-adic expansion of $A^{-1}B$, see (1). If $h = 0$ we can use Algorithm `SeriesSolution`. 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 $h > l$. The point of the algorithm here is that the complexity depends on k and $\deg B$ but not (essentially) on h. This is important because in typical applications $h \gg k$.

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

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

We omit the proof of correctness here. Since the cost estimate depends on many parameters, we only give a special case that interests us.

PROPOSITION 13. *Algorithm* `HighOrderLift` *is correct. If* $\log l = O(\log n)$ *and all of* m*,* mk/d *and* $m(\deg B)/d$ *are* $O(n)$*, then the cost of the algorithm is* $O((\log n)n^\theta d^{1+\epsilon})$ *field operations.*

9. INTEGRALITY CERTIFICATION

Let $A \in \mathsf{K}[x]^{n\times n}$ be nonsingular, $\det A \perp X$. Let $B \in \mathsf{K}[x]^{n\times m}$ and $T \in \mathsf{K}[x]^{m\times m}$. We present an algorithm to assay if $A^{-1}BT$ is over $\mathsf{K}[x]$. The algorithm works by computing a high order lift H of $A^{-1}B$ as shown in (1). Recall that $H = \mathrm{Left}(\mathrm{Trunc}(\mathrm{Inverse}(A,h+k)B,h+k),h)$. Let

$$C = \mathrm{Trunc}(HT,k).$$

PROPOSITION 14. *If* h *and* k *are chosen to satisfy* $hd > (n-1)d + \deg B + \deg T$ *and* $kd > d + \deg T$*, then* C *as computed above has* $\deg C < (k-1)d$ *if and only if* $A^{-1}BT$ *is integral.*

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

$$S = \overbrace{\mathrm{Trunc}(A^{-1}B,h)T}^{\text{degree} < hd+\deg T} + CX^h. \qquad (12)$$

By choice of k we have $hd + \deg T < hd + (k-1)d$. Now we will use the fact that $\mathrm{Trunc}(AS,h+k) = BT$. If $\deg CX^h < hd + (k-1)d$ also, then $AS = \mathrm{Trunc}(AS,h+k)$, whence $S = A^{-1}BT$. This shows the "only if". Now for the "if". The parameter h is chosen so that hd is strictly larger than an *a priori* bound on the degrees of numerators in $A^{-1}BT$. Thus, if $A^{-1}BT$ is integral, then it follows from (12) that $\deg C < \deg T$. □

The next two corollaries will be useful later on. Note that both corollaries assume h and k satisfy the constraints of Proposition 14. The first corollary follows from the proof above.

COROLLARY 15. *If* $A^{-1}BT$ *is integral, then* $\deg C < \deg T$.

The next lemma is obvious. For a fixed h we have:

COROLLARY 16. *If* $A^{-1}BT$ *is integral, then* C *is invariant of the choice of* k.

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

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

1. $l :=$ the smallest integer such that $2^l d > (n-1)d + \deg b + \deg T$;
   $k :=$ the smallest integer such that $2^k d > d + \deg T$;
   $H :=$ `HighOrderLift`$[X](A, B, l, k)$;

2. $C :=$ Trunc$(HT, 2^k)$;
   if $\deg C < \deg T$ **then**
   return (true, C)
   else
   return false
   fi

We get:

PROPOSITION 17. *Algorithm* `IntegralityCertificate` *is correct. If all of* m, $m(\deg B)/d$ *and* $m(\deg T)/d$ *are* $O(n)$, *and assuming* $\epsilon \leq \theta - 2$, *then the cost of the algorithm is* $O((\log n)n^\theta d^{1+\epsilon})$ *field operations.*

Worked example

The integrality certification technique described above can be adapted for integer matrices. Let

$$A = \begin{bmatrix} -28 & -11 & -56 & -39 \\ -5 & 42 & -10 & 37 \\ 22 & -44 & -25 & 44 \\ -32 & 3 & 38 & 46 \end{bmatrix}.$$

Let B to be the last two columns of I_2. Let $T = sI_2$ where $s = 3969$. For convenience, we will work with the usual base-10 decimal expansions. For $h = 90$ (overkill) and $k = 8$ we can use Maple(TM) to compute the H shown in (1) as follows:

```
H := evalm((map(mods,evalm(inverse(A)&*B),10^98) -
     map(mods,evalm(inverse(A)&*B),10^90))/10^90):
C := map(mods,evalm(H*3969),10^k):
```

We get

$$\begin{matrix} H \end{matrix} \begin{bmatrix} -12194507 & -23935500 \\ -24086672 & 42529604 \\ -5946082 & 33232552 \\ 24086672 & -42529604 \end{bmatrix} \text{ and } \begin{matrix} C \end{matrix} \begin{bmatrix} 1717 & 500 \\ -1168 & -1724 \\ 542 & -1112 \\ 1168 & 1724 \end{bmatrix}. \quad (13)$$

We conclude that $A^{-1}BT$ is integral since entries in C have substantially fewer than 8 decimal digits. Indeed, the analogue of Corollary 15 guarantees that $\|C\|_\infty < m\|T\|_\infty$.

10. DETERMINANT REDUCTION

We omit this section except for an example on integer matrices. Consider the matrix A and it's Hermite-form H.

$$\begin{matrix} A \end{matrix} \left[\begin{array}{cccc|c} 15 & 17 & 18 & -9 & 14 \\ -14 & 10 & -9 & -3 & 37 \\ 5 & -35 & 7 & 7 & 35 \\ -14 & 5 & 29 & -37 & -16 \\ 31 & -15 & -25 & -19 & 25 \end{array}\right], \quad \begin{matrix} H \end{matrix} \left[\begin{array}{cccc|c} 1 & 0 & 0 & 0 & 10172267 \\ & 1 & 0 & 0 & 13994003 \\ & & 1 & 1 & 38505091 \\ & & & 3 & 26289760 \\ & & & & 42348292 \end{array}\right]$$

Using rational system solving and extended gcd computation, we can produce a new matrix B, identical to A except for the last column, such that the Hermite form of B is identical to H except for the last column which has pivot entry one. The point is, the determinant is reduced by a factor of 42348292. The analogous construction can be done for polynomial matrices.

PROPOSITION 18. *Let* $A \in \mathsf{K}[x]^{n\times n}$ *and* $X \in \mathsf{K}[x]$ *satisfy* $\det A \perp X$ *and* $\deg X \geq \deg A$. *If* $\epsilon \leq \theta - 2$, *then a* B *as described above can be computed in* $O((\log n)n^\theta d^{1+\epsilon})$ *field operations.*

11. PARTIAL DETERMINANT

Let $A \in \mathsf{K}[x]^{n\times n}$ be nonsingular. Let $X \in \mathsf{K}[x]$ with $X \perp \det A$ and $d = \deg X \geq \deg A$ be given. Let $1 \leq m \leq n$ be given. Throughout this section let $\bar{H}$, $\bar{S}$, $\bar{s}$, and $\bar{L}$ be defined as follows:

- $\bar{H}$ is the trailing $m \times m$ submatrix of the Hermite basis of A.
- $\bar{S}$ is the Smith-form of $\bar{H}$.
- $\bar{s}$ is the largest invariant factor in $\bar{S}$.
- $\bar{L}$ is the last m rows of A^{-1}.

The next two lemmas follow from the uniqueness of $\bar{H}$ and $\bar{s}$. For sundry such facts about lattices, see [12, Chapter 5].

LEMMA 19. *Let* $s \in \mathsf{K}[x]$ *be a multiple of* $\bar{s}$. *Then* $s\bar{L}$ *is integral.*

LEMMA 20. *Let* $T \in \mathsf{K}[x]^{m\times m}$ *be such that* $T\bar{L}$ *is integral. Then* $T = \bar{T}\bar{H}$ *for some* $\bar{T} \in \mathsf{K}[x]^{m\times m}$.

We now describe a novel algorithm to compute $\bar{S}$ from A and a given s as in Lemma 19. Let B be the last m rows of I_n so that $BA^{-1} = \bar{L}$. Lemma 19 attests that sBA^{-1} is integral. We want to use algorithm `IntegralityCertificate` to produce a certificate to this effect. The fact that our situation here is "transposed" is no problem: note that $((A^t)^{-1}B^t(sI_m)^t)^t = sBA^{-1}$. Let

$$C^t := \texttt{IntegralityCertificate}[X](A^t, B^t, sI_m).$$

Then $\deg C < \deg s$ (Lemma 15) and C can be produced in time $O^\sim(n^\theta d)$ if $m(\deg s)/d$ is $O(n)$ (Proposition 17).

PROPOSITION 21. *$\bar{S}$ is the Smith-form of sD^{-1}, where D is the principal $m \times m$ submatrix of the Smith form of $\left[\begin{array}{c|c} C & sI_m \end{array}\right]$.*

PROOF. Let H^t be the high order lift computed in the algorithm. The algorithm computes $C = \text{Trunc}(sH, k)$ for some choice of k, $kd > d + \deg s$. We have $sI_m = \bar{T}\bar{H}$ for some $\bar{T}$ over $\mathsf{K}[x]$ (Lemma 20), so

$$C = \text{Trunc}(\bar{T}\overbrace{\text{Trunc}(\bar{H}H, k)}^{\bar{C}}, k). \tag{14}$$

Recall that $BA^{-1} = \bar{L}$. Since $\bar{H}\bar{L}$ is integral $\deg \bar{C} < \deg \bar{H}$ (Corollary 15). Since $\bar{T}$ and $\bar{C}$ are over $\mathsf{K}[x]$, we may conclude from (14) that $C = \bar{T}\bar{C}$ (Corollary 16).

We have established that

$$\left[\begin{array}{c|c} C & sI_m \end{array}\right] = \bar{T}\left[\begin{array}{c|c} \bar{C} & \bar{H} \end{array}\right].$$

Let G be the column Hermite-form of $\left[\begin{array}{c|c} \bar{C} & \bar{H} \end{array}\right]$. If $G = I_m$ then the D in the statement of the proposition is equal to the Smith-form of $\bar{T}$. Using $s\bar{T}^{-1} = \bar{H}$ it is easy to derive that sD^{-1} is equivalent to $\bar{H}$. Thus, we will be finished if we can show that $G = I_m$.

To arrive at a contradiction, suppose $G \neq I_m$. Both $G^{-1}\bar{C}$ and $G^{-1}\bar{H}$ are necessarily integral with degrees bounded by $\deg \bar{C}$ and $\deg \bar{H}$ respectively. But then the degree of $\text{Trunc}((G^{-1}\bar{H})H, k)$ is $\leq \deg \bar{C}$, implying that $(G^{-1}\bar{H})\bar{L}$ is integral (Proposition 14). Lemma 20 now gives a contradiction. □

The Smith-form in Proposition 21 can be computed in time $O(nm^{\theta-1}(\deg s)^{1+\epsilon})$ using [12, Lemma 7.14].

PROPOSITION 22. *Let A, X and m as described above be given. Let a nonzero multiple s of $\bar{s}$ also be given. If $m(\deg s)/d = O(n)$, and assuming $\epsilon \leq \theta - 2$, then $\bar{S}$ as described above can be computed in $O((\log n)n^{\theta}d^{1+\epsilon})$ field operations.*

Worked example

The ideas described above carry over to the case of integer matrices. The matrix

$$A = \begin{bmatrix} -28 & -5 & 22 & -32 \\ -11 & 42 & -44 & 3 \\ -56 & -10 & -25 & 38 \\ -39 & 37 & 44 & 46 \end{bmatrix}$$

has Hermite-form

$$H = \left[\begin{array}{cc|cc} 1 & 220 & 0 & 379 \\ & 1231 & 2 & 670 \\ \hline & & 3 & 3792 \\ & & & 3969 \end{array}\right]$$

and

$$\bar{H} = \begin{bmatrix} 3 & 3792 \\ 0 & 3969 \end{bmatrix} \quad \text{has Smith form} \quad \bar{S} = \begin{bmatrix} 3 & \\ & 3969 \end{bmatrix}.$$

Let $s = 3969$ and C be the integrality certificate shown in (13), except transposed. Then the principal 2×2 submatrix of the Smith form of

$$\left[\begin{array}{cccc|cc} 1717 & -1168 & 542 & 1168 & 3969 & \\ 500 & -1724 & -1112 & 1724 & & 3969 \end{array}\right]$$

is

$$D = \begin{bmatrix} 1 & \\ & 1323 \end{bmatrix}.$$

Note that the Smith form of sD^{-1} is $\bar{S}$.

12. DETERMINANT COMPUTATION

Suppose $A \in \mathsf{R}^{n \times n}$ is nonsingular with $\det \perp X$ and $\deg X \geq \deg A$. We present an algorithm to compute the determinant of A. We are going to assume that A satisfies some rather strong conditions. First, assume wlog (up to augmentation with an identity matrix) that $n = 2^{k+1} - 1$ for some k. Decompose the Hermite basis H of C as

$$H = \begin{bmatrix} H_k & * & * & * \\ & \ddots & \vdots & \vdots \\ & & H_1 & * \\ & & & H_0 \end{bmatrix}$$

where H_i has dimension $2^i \times 2^i$. Let S_i be the Smith-form of H_i. The algorithm requires that

- (C1) $\text{diag}(S_k, S_{k-1}, \ldots, S_1, S_0)$ is in Smith-form.
- (C2) The Hermite basis of $A[1 \ldots m+1, 1 \ldots m]$ is equal to the Hermite basis of $A[*, 1 \ldots m]$, $m = 2^k + 2^{k-1} + \cdots + 2^i$, $i = k, k-1, \ldots, 1$.

If any of these conditions are not satisfied, the algorithm will detect this and report failure. Given (C2) holds for a given m, we will assume wlog (up to a permutation of the first m rows) that $A[1 \ldots m, 1 \ldots m]$ is nonsingular.

The algorithm proceeds in stages for $i = 0, 1, 2, \ldots, k$. Stage $i = 0$ is to compute S_0. Now fix some i, $i > 0$. Let $m = 2^k + 2^{k-1} + \ldots + 2^i$. Let B be the matrix constructed from $A[1 \ldots m+1, 1 \ldots m+1]$ using the algorithm supporting Proposition 18. Let $C = A[m+2 \ldots n, 1 \ldots m+1]$.

LEMMA 23. *(C2) holds for i iff CB^{-1} is integral.*

In case $\mathsf{R} = \mathsf{K}[x]$, the integrality of CB^{-1} can be assayed using Algorithm `IntegralityCertificate`. Assume henceforth that CB^{-1} is integral. (If it isn't, report failure and terminate.)

At this point we have constructed an $(m+1) \times (m+1)$ matrix B which has Hermite-form equal to

$$\left[\begin{array}{c|c|c} * & * & \\ \hline & H_i & \\ \hline & & 1 \end{array}\right].$$

Our goal now is to recover the Smith-form of H_i. Let s be the smallest invariant factor in S_{i-1}, computed during the (previous) stage $i-1$. Finally, use Proposition 22 to

either determine that $\mathrm{diag}(S_i, S_{i-1})$ is not in Smith-form or to recover S_{i-1}. Since $\mathrm{diag}(S_{i-1}, S_{i-2}, \ldots, S_0)$ is in Smith-form, we must have $\deg s \leq nd/m$. The cost is given by Proposition 22. Summing over all stages gives

PROPOSITION 24. *Let $A \in \mathsf{K}[x]^{n\times n}$ be nonsingular. Let $X \in \mathsf{K}[x]$ with $X \perp \det A$ and $\deg X \geq \deg A$ be given. The algorithm described above will assay if A satisfies conditions $(C1)$ and $(C2)$. If so, the algorithm will produce $\mathrm{diag}(S_k, S_{k-1}, \ldots, S_0)$. Assuming $\epsilon \leq \theta - 2$, the cost of the algorithm is $O((\log n)^2 n^\theta d^{1+\epsilon})$ field operations.*

That $\mathrm{diag}(S_k, S_{k-1}, \ldots, S_0)$ is the Smith-form of A (this does not follow from (C1) directly) can be assayed in the same time using k integrality certifications. We omit the details here.

13. CONCLUSIONS

The algorithms in sections 5—11 are deterministic but require as input a small degree X such that $X \perp \det A$. See [7, Proof of Theorem 29] for a method of finding such an X randomly.

The algorithms in Section 12 requires that A satisfy some conditions. These are easy to achieve using the preconditioning technique as shown in [4]. 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 [4, Algorithm 3.2] and [13, Algorithm REDUCE]). If $\#\mathsf{K}$ is too small, work over an algebraic extension field.

Arguably the most important contribution of this paper is the idea of using high-order lifting to certify integrality. Without this technique, many of the algorithms we propose would be Monte Carlo instead of Las Vegas. The algorithms we have given for determinant/Smith-form computation are of practical interest, especially because they certify the output.

The main task remaining is to extend the results here to the case of integer matrices. The reader may have already noticed that the key ideas Sections 9—12 carry over easily. The main difficulties to be solved are:

- Achieve a suitable preconditioning for the input matrix of the Smith-form computation.
- Get analogous versions of the lifting algorithms in Sections 5, 7 and 8.

To solve the first difficulty the results in [3] and [8] should prove useful. The idea is to modify the determinant algorithm to allow considerably weaker conditions than those outlined at the beginning of Section 12.

The crux of the second difficulty is that the absolute value norm over $\mathbb{Z}$, unlike the degree norm over $\mathsf{K}[x]$, 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-adic number system. We will present this in a future paper.

14. REFERENCES

[1] J. Abbott, M. Bronstein, and T. Mulders. Fast deterministic computation of determinants of dense matrices. In S. Dooley, editor, *Proc. Int'l. Symp. on Symbolic and Algebraic Computation: ISSAC '99*, pages 197–204. ACM Press, 1999.

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

[3] W. Eberly, M. Giesbrecht, and G. Villard. Computing the Smith form of a dense integer matrix. In *Proc. 31st Ann. IEEE Symp. Foundations of Computer Science*, 2000.

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

[5] E. Kaltofen and G. Villard. Computing the sign or the value of the determinant of an integer matrix, a complexity survey. 2002. Submitted to the special issue on Congrès International Algèbre Linéaire et Arithmétique: Calcul Numérique, Symbolique et Parallèle, held in Rabat, Morocco, May 2001, 17 pages.

[6] R. T. Moenck and J. H. Carter. *Approximate algorithms to derive exact solutions to systems of linear equations.*, pages 65–72. Springer-Verlag, Berlin-Heidelberg-New York, 1979.

[7] T. Mulders and A. Storjohann. Diophantine linear system solving. In S. Dooley, editor, *Proc. Int'l. Symp. on Symbolic and Algebraic Computation: ISSAC '99*, pages 281–288. ACM Press, 1999.

[8] T. Mulders and A. Storjohann. Certified diophantine dense linear system solving. Technical Report 355, Departement Informatik, ETH Zürich, Dec. 2000.

[9] T. Mulders and A. Storjohann. On lattice reduction for polynomial matrices. Technical Report 356, Departement Informatik, ETH Zürich, Dec. 2000.

[10] T. Mulders and A. Storjohann. Rational solutions of singular linear systems. In C. Traverso, editor, *Proc. Int'l. Symp. on Symbolic and Algebraic Computation: ISSAC '00*, pages 242–249. ACM Press, 2000.

[11] V. Pan. Computing the determinant and the charactersitic polynomial of a matrix via solving linear systems of equations. *Inf. Proc. Letters*, 28:71–75, 1988.

[12] A. Storjohann. *Algorithms for Matrix Canonical Forms.* PhD thesis, ETH – Swiss Federal Institute of Technology, 2000.

[13] A. Storjohann and G. Labahn. Preconditioning of rectangular polynomial matrices for efficient Hermite normal form computation. In A. H. M. Levelt, editor, *Proc. Int'l. Symp. on Symbolic and Algebraic Computation: ISSAC '95*, pages 119–125. ACM Press, 1995.