# On the complexity of inverting integer and polynomial matrices 

Arne Storjohann<br>David R. Cheriton School of Computer Science<br>University of Waterloo, Ontario, Canada N2L 3G1

July 24, 2008


#### Abstract

An algorithm is presented that probabilistically computes the exact inverse of a nonsingular $n \times n$ integer matrix $A$ using $O^{\sim}\left(n^{3}(\log \|A\|+\log \kappa(A))\right)$ bit operations. Here, $\|A\|=\max _{i j}\left|A_{i j}\right|$ denotes the largest entry in absolute value, $\kappa(A):=\left\|A^{-1}\right\|\|A\|$ is the condition number of the input matrix, and the soft-O notation $O^{\sim}$ indicates some missing $\log n$ and $\log \log \|A\|$ factors. A variation of the algorithm is presented for polynomial matrices. The inverse of any nonsingular $n \times n$ matrix whose entries are polynomials of degree $d$ over a field can be computed using an expected number of $O^{\sim}\left(n^{3} d\right)$ field operations. Both algorithms are randomized of the Las Vegas type: fail may be returned with probability at most $1 / 2$, and if fail is not returned the output is certified to be correct in the same running time bound.


## 1 Introduction

Let $A \in \mathbb{Z}^{n \times n}$ be nonsingular. We denote by $\|A\|:=\max \left|A_{i j}\right|$ the maximum magnitude of entries in $A$, and by $\kappa(A):=\|A\|\left\|A^{-1}\right\|$ the condition number of the matrix with respect to the max norm. We give a Las Vegas probabilistic algorithm that has expected running time $O^{\sim}\left(n^{3}(\log \|A\|+\log \kappa(A))\right)$ bit operations to compute the exact inverse of $A$. Thus, for a well conditioned $A$, with $\kappa(A)$ bounded by a polynomial function of $n \log \|A\|$, this cost estimate becomes $O^{\sim}\left(n^{3} \log \|A\|\right)$. For comparison, the sum of the bitlengths of all entries in the inverse of a nonsingular $A \in \mathbb{Z}^{n \times n}$ may be more than $n^{3} \log _{2}\|A\|$ bits. To the best of our knowledge, the best previously known complexity estimate for integer matrix inversion is $O^{\sim}\left(n^{\omega+1} \log \|A\|\right)$ bit operations, supported by any of the classical approaches such as homomorphic imaging and Chinese remaindering [6, Section 5.5], quadratic lifting via Newton iteration, or a recursive version of fraction-free Gaussian elimination [23, Section 2]. Here, $\omega$ is the exponent for matrix multiplication over a ring [2, Chapter 1] and the soft-O notation $O^{\sim}$ indicates some missing $\log n$ and $\log \log \|A\|$ factors.

Now consider the case of polynomial matrices. Let K be a field, and let $A \in \mathrm{~K}[x]^{n \times n}$ be nonsingular with entries bounded in degree by $d>0$. The inverse of $A$ may require on the order of $n^{3} d$ field elements to represent. Similar to the integer case, a variety of classical approaches for polynomial matrix inversion exist, all with a cost estimate of $O^{\sim}\left(n^{\omega+1} d\right)$ field operations from K. We refer to [10, Section 1] for a brief survey of previous methods, and for a discussion of some of the progress made in obtaining faster algorithms for problems on polynomial matrices. Jeannerod and Villard [10] propose a new approach that, for a generic $A$ with $n$ a power of two, will compute $A^{-1}$ in $O^{\sim}\left(n^{3} d\right)$ field operations. In this paper we adapt our approach for integer matrix inversion to the polynomial case. By incorporating some algebraic preconditioning techniques that are particular to polynomials, we obtain a Las Vegas probabilistic algorithm that has expected running time $O^{\sim}\left(n^{3} d\right)$ for any nonsingular input.

The discovery of the essentially optimal inversion algorithm for generic polynomial matrices in [10], and the recent progress made in reducing the complexity of many basic linear algebra problems on integer matrices, motivates us to develop an alternative algorithm for inversion that is applicable to integer matrices. Assuming $\omega=3$, we recall in the next two paragraphs some key results for two problems that are particularly relevant for this paper: nonsingular rational linear system solving and determinant/Smith-form computation. For a survey of work that has been done on computing these and other integer matrix invariants, as well as incorporating fast matrix multiplication techniques, we refer to [12, 24].

It was shown already more than a quarter of a century ago that a rational system solution $A^{-1} b$ can be computed in $O^{\sim}\left(n^{3} \log \|A\|\right)$ bit operations using linear $p$-adic lifting [15, 3]. Taking $b$ to be a column of the identity matrix shows that any single column of the inverse can be computed in $O^{\sim}\left(n^{3} \log \|A\|\right)$ bit operations. Linear $p$-adic lifting is a key subroutine of the algorithm in this paper. We show that lifting can be used to compute all $n$ columns of the inverse of a well conditioned matrix in the same time as a single column.

Now consider the computation of the determinant, assuming $\omega=3$. Classical methods such as homomorphic imaging [6, Section 5.5] require $O^{\sim}\left(n^{4} \log \|A\|\right)$ bit operations. A breakthrough was the Krylov approach of Kaltofen [11] which gives an $O^{\sim}\left(n^{3.5} \log \|A\|\right)$ Las Vegas algorithm. More recently, a Monte Carlo algorithm with the same running time to compute the Smith form of $A$ is given by Eberly et al. [5]. The Smith form $\operatorname{Diag}\left(s_{1}, s_{2}, \ldots, s_{n}\right)$ of an $A \in \mathbb{Z}^{n \times n}$ is a canonical diagonalization under unimodular pre- and post-multiplication, see Section 2. The invariant factors $s_{i}$ satisfy $|\operatorname{det} A|=s_{1} s_{2} \cdots s_{n}$, so the determinant of $A$ is easily recovered once the form is known. The approach in [5] is based on computing the $k \in$ $O(\sqrt{n})$ distinct invariant factors of the Smith form of $A$, together with their multiplicities, by computing $O^{\sim}(k)$ rational linear system solutions, each at a cost of $O^{\sim}\left(n^{3} \log \|A\|\right)$ using linear $p$-adic lifting. The overall cost of the algorithm in [5] is thus sensitive to $k$, the number of distinct invariant factors. On the one hand, the algorithm we present here avoids this sensitivity to $k$ by being able exploit the fact that $\sum_{i=1}^{n} \operatorname{bitlength}\left(s_{i}\right) \leq n+\operatorname{bitlength}(\operatorname{det} A)$, independent of the invariant structure of $A$. On the other hand, our algorithm is sensitive to the condition number $\kappa(A)$. We compute all $n$ invariant factors in succession in time proportional to about $n^{2} \sum_{i=1}^{n}\left(\log s_{i}+\log \kappa(A)\right)$ bit operations.

Next, we motivate our approach for integer matrix inversion by recalling a fact about the adjoint of an $n \times n$ matrix $A$ over a field that has rank $n-1$, and then observe how this gives a simple formula for the adjoint of a class of matrices over $\mathbb{Z}$.

Since $A$ has rank $n-1, A$ has at least one nonzero minor dimension $n-1$. Assume without loss of generality (up to a row and column permutation) that the leading $(n-1) \times(n-1)$ submatrix $\bar{A}$ of $A$ is nonsingular, and partition $A$ as

$$
A=\left[\begin{array}{l|l}
\bar{A} & y \\
\hline x & a
\end{array}\right]
$$

If we set $u:=-x \bar{A}^{-1}$, a row vector, and $v:=-\bar{A}^{-1} y$, a column vector, then

$$
\left[\begin{array}{c|c}
I_{n-1} &  \tag{1}\\
\hline u & 1
\end{array}\right]\left[\begin{array}{c|c}
\bar{A} & y \\
\hline x & a
\end{array}\right]\left[\begin{array}{l|l}
I_{n-1} & v \\
\hline & 1
\end{array}\right]=\left[\begin{array}{l|l}
\bar{A} & \\
\hline & 0
\end{array}\right]
$$

The adjoint of the matrix on the right hand side of (1) will have all entries zero except for the entry in the last row and last column, which will be equal to det $\bar{A}$. Replacing both sides of (1) with the adjoint of that side and solving for $A^{\text {adj }}$ gives

$$
\begin{equation*}
A^{\mathrm{adj}}=\left[\frac{(\operatorname{det} \bar{A}) v}{\operatorname{det} \bar{A}}\right][u \mid 1] \tag{2}
\end{equation*}
$$

Note that the formula for $A^{\text {adj }}$ in (2) is valid also when the entries of $A$ are coming from a principal ideal ring, even a ring with zero divisors such as a residue class ring of the integers, provided that $\operatorname{det} A=0$ and $\operatorname{det} \bar{A}$ is a unit from the ring (i.e., $\bar{A}$ is invertible over the ring). Consider in particular a nonsingular matrix $A \in \mathbb{Z}^{n \times n}$ for which the leading $(n-1) \times(n-1)$ submatrix $\bar{A}$ satisfies $\operatorname{det} \bar{A} \perp \operatorname{det} A$. Then over the residue class ring $\mathbb{Z} /\langle\operatorname{det} A\rangle$, we have $\operatorname{det} A \equiv 0(\bmod \operatorname{det} A)$ and $\operatorname{det} \bar{A}$ a unit, so equation (2) holds modulo $\operatorname{det} A$. In other words, the adjoint of $A$ over $\mathbb{Z}$ is element-wise congruent to the rank 1 matrix in (2). Moreover, if $\operatorname{det} A$ is large enough, the exact adjoint of $A$ over $\mathbb{Z}$ can be obtained by multiplying out the outer product in (2) and reducing all entries modulo $\operatorname{det} A$ in the symmetric range $[-\lfloor(|\operatorname{det} A|-1) / 2\rfloor,\lfloor|\operatorname{det} A| / 2\rfloor]$.

For example, the matrix

$$
A=\left[\begin{array}{cccc}
-93 & -32 & 8 & 44 \\
-76 & -74 & 69 & 92 \\
-72 & -4 & 99 & -31 \\
-2 & 27 & 29 & 67
\end{array}\right]
$$

has $\operatorname{det} A=64334045$, which is relatively prime to $\operatorname{det} \bar{A}=533666$. Formula (2) gives

$$
\begin{align*}
A^{\text {adj }} & \equiv\left[\begin{array}{c}
-28408 \\
861667 \\
181262 \\
533666
\end{array}\right]\left[\begin{array}{cccc}
-27259506 & 11404741 & -8995165 & 1
\end{array}\right] \quad(\bmod 64334045) \\
& \equiv\left[\begin{array}{cccc}
-853217 & 368292 & -179420 & -28408 \\
409178 & -744548 & 233455 & 861667 \\
-584392 & 295157 & 438250 & 181262 \\
62584 & 183281 & -289125 & 533666
\end{array}\right] \quad(\bmod 64334045) . \tag{3}
\end{align*}
$$

For this example, $\operatorname{det} A$ is large enough to capture all entries in $A^{\text {adj } ; ~ t h e ~ m a t r i x ~ i n ~(3), ~}$ obtained by multiplying out the outer product and reducing entries in the symmetric range modulo 64334045 , is the adjoint of $A$ over $\mathbb{Z}$.

To adapt the approach just described to compute the inverse of an arbitrary input matrix requires handling the case when all minor of dimension $n-1$ in $A$ have a common factor with $\operatorname{det} A$ (i.e., the Smith form of $A$ is nontrivial). Our solution is to extend formula (2) to an $A$ with nontrivial invariant structure by giving an expression for $A^{\text {adj }} \bmod \operatorname{det} A$ as the sum of scaled outer products.

$$
A^{\text {adj }} \equiv \frac{\operatorname{det} A}{s_{n}} v_{n} u_{n}+\frac{\operatorname{det} A}{s_{n-1}} v_{n-1} u_{n-1}+\cdots+\frac{\operatorname{det} A}{s_{1}} v_{1} u_{1} \quad(\bmod \operatorname{det} A) .
$$

Each $v_{i}$ is a column vector and each $u_{i}$ a row vector. We call this construction an outer product adjoint formula for $A$. Actually, since all entries in $A^{\text {adj }}$ are divisible by $(\operatorname{det} A) / s_{n}$, our definition of the formula in Section 4 is as shown above but with $\operatorname{det} A$ replaced by $s_{n}$, and the left hand side replaced with $s_{n} A^{-1}$.

We remark that the existence of an outer product adjoint formula for any nonsingular $A \in \mathrm{R}^{n \times n}$ over any principal ideal domain R follows directly from the existence of the Smith form over R. Our contribution in this paper is twofold. First, we identify and exploit some useful properties of the formula: (1) when $\mathrm{R}=\mathbb{Z}$ or $\mathrm{R}=\mathrm{K}[x]$, an outer product adjoint formula can be represented using about the same space to represent $A ;(2)$ an outer product adjoint formula gives an essentially optimal method to compute $\operatorname{Rem}\left(A^{\operatorname{adj}} v, \operatorname{det} A\right)$ for a given vector $v$ that has entries reduced modulo $\operatorname{det} A$. Second, we present fast probabilistic algorithms to compute an outer product adjoint formula.

When applying the outer product adjoint formula to compute $A^{\text {adj }}$, a problem occurs if $|\operatorname{det} A|$ is too small to capture the entries of $A^{\text {adj }}$ in the symmetric range modulo $\operatorname{det} A$. In this case we divide the computation of $A^{\text {adj }}$ into two parts. Consider the decomposition $s_{n} A^{-1}=$ $\operatorname{Rem}\left(s_{n} A^{-1}, s_{n}\right)+s_{n} R$ where Rem denotes the remainder modulo $s_{n}$. First we compute $\operatorname{Rem}\left(s_{n} A^{-1}, s_{n}\right)$ via an outer product adjoint formula. Then we compute the remaining part $R:=\left(1 / s_{n}\right)\left(s_{n} A^{-1}-\operatorname{Rem}\left(s_{n} A^{-1}, s_{n}\right)\right)$ using linear $p$-adic lifting. Unfortunately, the cost of computing both of these parts is sensitive to $\log \kappa(A)$.

The rest of this paper is organized as follows. In Section 2 we fix some notation and recall some definitions, including that of the Smith canonical form. Section 3 considers the problem of computing, for a nonsingular $A \in \mathrm{R}^{n \times n}$, a Smith decomposition: $A=* \operatorname{snf}(A) *$ where $*$ are $n \times n$ matrices over R . Section 4 defines, and gives an algorithm for computing, the outer product adjoint formula, which is essentially a Smith decomposition $s_{n} A^{-1}=$ $* \operatorname{snf}\left(s_{n} A^{-1}\right) *$. Sections 3 and 4 develop results generically so they apply both over $\mathrm{R}=$ $\mathbb{Z}$ and $\mathrm{R}=\mathrm{K}[x]$. Section 5 gives our integer matrix inversion algorithm, and Section 6 the variation for polynomial matrices. Fortuitously, we can apply some simple algebraic preconditioning techniques to transform an original input matrix $A \in \mathrm{~K}[x]^{n \times n}$ of degree $d$ into a new matrix $B$ that will have determinant of degree $n d$. Once the inverse of $B$ has been determined via an outer product adjoint formula, the preconditioning can be reversed in the allotted time to recover the inverse of $A$.

## Cost functions

Let $\mathrm{M}: \mathbb{Z}_{>0} \longrightarrow \mathbb{R}_{>0}$ be such that integers bounded in magnitude by $2^{t}$ can be multiplied using at most $\mathrm{M}(t)$ bit operations. The Schönhage-Strassen algorithm [21] allows $\mathrm{M}(t)=$ $O(t(\log t)(\log \log t))$. We assume that $\mathrm{M}(a)+\mathrm{M}(b) \leq \mathrm{M}(a+b)$ and $\mathrm{M}(a b) \leq \mathrm{M}(a) \mathrm{M}(b)$ for $a, b \in \mathbb{N}_{\geq 2}$. We refer to [6, Section 8.3] for further references and discussion about integer multiplication.

It will be useful to define an additional function B for bounding the cost of integer gcdrelated computations. We can take $\mathrm{B}(t)=\mathrm{M}(t) \log t$. Then the extended gcd problem with two integers bounded in magnitude by $2^{t}$, and the rational number reconstruction problem [6, Section 5.10] with modulus bounded by $2^{t}$, can be solved with $O(\mathrm{~B}(t))$ field operations [20] (compare with [19]).

We will overload notation slightly and use $\mathrm{M}: \mathbb{Z}_{\geq 0} \longrightarrow \mathbb{R}_{>0}$ as a cost functions for polynomial multiplication: two polynomials in $\mathrm{K}[x]$ of degree bounded by $d$ can be multiplied using at most $\mathrm{M}(d)$ field operations. Similarly, B will be used as a cost function for gcdrelated problems like rational function reconstruction and extended gcd. Similar to the integer case, we can take $\mathrm{B}(d)=\mathrm{M}(d) \log d$. We refer to [6, Section 11.1] for more details and references.

## 2 Definitions, notation and preliminaries

Let R be a principal ideal ring, a commutative ring with identity in which every ideal is principal. In this paper our focus is on the integral domains $\mathrm{R}=\mathbb{Z}$ and $\mathrm{R}=\mathrm{K}[x]$. Following [17], we prescribe a complete set of non-associates $\mathcal{A}(\mathrm{R})$ and, for every nonzero $s \in \mathrm{R}$, a complete set of residues $\mathcal{R}(\mathrm{R}, s)$, as follows.

$$
\begin{array}{l|l}
\mathrm{R}=\mathbb{Z} & \mathrm{R}=\mathrm{K}[x]  \tag{4}\\
\hline \mathcal{A}(\mathbb{Z})=\{0,1,2, \ldots\} & \mathcal{A}(\mathrm{K}[x])=\{0\} \cup\{f \in \mathrm{~K}[x] \mid f \text { is monic }\} \\
\mathcal{R}(\mathbb{Z}, s)=\left[-\left\lfloor\frac{|s|-1}{2}\right\rfloor,\left\lfloor\frac{|s|}{2}\right\rfloor\right] & \mathcal{R}(\mathrm{K}[x], s)=\{f \in \mathrm{~K}[x] \mid \operatorname{deg} f<\operatorname{deg} s\}
\end{array}
$$

Note that our choice for $\mathcal{R}(\mathbb{Z}, s)$ corresponds to the usual "symmetric range" modulo $s$.
For nonzero $N$, the function $\operatorname{Rem}(a, N)$ returns the element of $\mathcal{R}(\mathrm{R}, N)$ that is congruent to $a$ modulo $N$. The next lemma follows as a consequence of our choices for $\mathcal{R}(\mathrm{R}, N)$.

Lemma 1. Let $a, N \in \mathrm{R}$ with $N$ nonzero. Then $a=\operatorname{Rem}(a, N)$ if

$$
\begin{array}{ll}
\mathrm{R}=\mathbb{Z}: & |N| \geq 2|a|+2 \\
\mathrm{R}=\mathrm{K}[x]: & \operatorname{deg} N \geq \operatorname{deg} a+1
\end{array}
$$

For $a, s \in \mathrm{R}$, we denote by $\operatorname{gcd}(a, s)$ the unique principal generator in $\mathcal{A}(\mathrm{R})$ of the ideal generated by $a$ and $s$. We allow gcd to take an arbitrary number of arguments, including matrices and vectors as well as individual elements of R . For example, if $B$ is a matrix over R and $s$ is an element of R , then $\operatorname{gcd}(B, s)$ denotes the $\operatorname{gcd}$ of $s$ and all entries in $B$.

We can use the definition of gcd over R to induce definitions of $\mathcal{A}$ and $\mathcal{R}$ for a residue class ring $\mathrm{R} /\langle s\rangle$ from the definitions of $\mathcal{A}$ and $\mathcal{R}$ over R . This will be useful below where we recall how the Smith form over $\mathbf{R}$ can be computed by working over $\mathrm{R} /\langle s\rangle$ for a well chosen $s$. For nonzero $s \in \mathrm{R}$, we identify the residue class ring $\mathrm{R} /\langle s\rangle$ with the set of elements $\mathcal{R}(\mathrm{R}, s)$, and define

$$
\mathcal{A}(\mathrm{R} /\langle s\rangle)=\{\operatorname{gcd}(a, s) \mid a \in \mathcal{R}(\mathrm{R}, s)\} \quad \text { and } \mathcal{R}(\mathrm{R} /\langle s\rangle, b)=\mathcal{R}(\mathrm{R}, \operatorname{gcd}(b, s)) .
$$

These choices for $\mathcal{A}$ and $\mathcal{R}$ allow us to easily obtain algorithms for basic operations over $\mathrm{R} /\langle s\rangle$ in terms of algorithms for basic operations over R, see [23, Section 1].

## The Smith canonical form

Corresponding to every $A \in \mathrm{R}^{n \times n}$ there exist unimodular (invertible over R ) matrices $P, Q \in$ $\mathrm{R}^{n \times n}$ such that $\operatorname{snf}(A)=S=U A V=\operatorname{Diag}\left(s_{1}, s_{2}, \ldots, s_{n}\right)$ with $S$ in Smith canonical form [17, Chapter II], that is, with $s_{i} \mid s_{i+1}$ for $1 \leq i \leq n-1$ and $s_{i} \in \mathcal{A}(\mathrm{R})$ for $1 \leq i \leq n$. When R is a principal ideal domain, $s_{n}$ is an associate of $(\operatorname{det} A) / \operatorname{gcd}\left(A^{\text {adj }}\right)$. Thus, the largest invariant factor $s_{n}$ is the smallest nonzero element of R (minimal degree over $\mathrm{K}[x]$ and minimal magnitude over $\mathbb{Z}$ ) such that $s_{n} A^{-1}$ is over R .

The classical approach to compute the Smith form over $\mathrm{R}=\mathbb{Z}$ or $\mathrm{R}=\mathrm{K}[x]$ is to apply a sequence of elementary row and column operations. A well known problem is that entries in the work matrix can grow excessively large. To avoid this phenomenon, many authors (e.g., $[4,9,8]$ ) have used the idea of the following lemma in conjunction with that of Lemma 1. The lemma holds independently of of the choices for $\mathcal{A}$ and $\mathcal{R}$, and follows from existence and uniqueness of the Smith form over any principal ideal ring [13], in particular over $\mathbf{R} /\langle s\rangle$.

Lemma 2. Let $A \in \mathrm{R}^{n \times n}$ be nonsingular, and let $s \in \mathrm{R}$ be a nonzero multiple of $s_{n}$, the largest invariant factor of $A$. If $S$ is the Smith form of $A$ over R , and $\bar{S}$ is the Smith form of $\bar{A}=\operatorname{Rem}(A, s)$ over $\mathrm{R} /\langle s\rangle$, then $\bar{S}=\operatorname{Rem}(S, s)$.

To help clarify the algorithms developed in the subsequent sections, we sketch here an approach of [7] to transform a nonsingular $A \in \mathbb{Z}^{n \times n}$ to Smith form. Suppose we have precomputed $s$, a nonzero multiple of $s_{n}$ as specified in Lemma 2. Compute $c_{2}, c_{3}, \ldots, c_{n} \in \mathbb{Z}$ such that the gcd of $s$ with all entries in the column vector $A\left[1 \left\lvert\, \begin{array}{cccc}c_{2} & c_{3} & \cdots & c_{n}\end{array}\right.\right]^{T}$ equals $\operatorname{gcd}(s, A)$. Now compute $x_{1}, x_{2}, \ldots, x_{n}$ with $x_{1} \perp s$ such that

$$
\left[\begin{array}{c|cccc}
x_{1} & x_{2} & x_{3} & \cdots & x_{n} \\
\hline & 1 & & & \\
& & 1 & & \\
& & & \ddots & \\
& & & & 1
\end{array}\right] A\left[\begin{array}{c|ccc}
1 & & & \\
\hline c_{2} & 1 & & \\
c_{3} & & 1 & \\
\vdots & & \ddots & \\
c_{n} & & & \\
\hline
\end{array}\right]=\left[\begin{array}{c|c}
s_{1} & u \\
\hline & \\
v & * \\
&
\end{array}\right]
$$

with $s_{1}$ equal to the gcd of $s$ and all entries in $A$. Next, apply some more elementary row and column operations to zero out entries below and to the right of $s_{1}$.

$$
\left[\begin{array}{c|c}
1 & -u / s_{1} \\
\hline & I_{n-1}
\end{array}\right]\left[\begin{array}{c|c}
s_{1} & u \\
\hline v & *
\end{array}\right]\left[\begin{array}{c|c}
1 & \\
\hline-v / s_{1} & I_{n-1}
\end{array}\right]=\left[\begin{array}{l|l}
s_{1} & \\
\hline & A
\end{array}\right] .
$$

The entries of $\bar{A}$ may be reduced modulo $s$ by applying the idea of Lemma 2. The remaining invariant factors are computed in a similar fashion by recursing on $\bar{A}$. Actually, since $s_{1}$ divides all entries in $\operatorname{Diag}\left(s_{1}, \bar{A}\right)$, we can work with $(1 / s) \bar{A}$ and modulus $s / s_{1}$ instead.

## 3 The Smith decomposition

Let $B \in \mathrm{R}^{n \times n}$ be nonsingular with $\operatorname{snf}(B)=\operatorname{Diag}\left(g_{1}, g_{2}, \ldots, g_{n}\right)$. In this section we adapt the approach for Smith form computation described in the previous section to compute not only the Smith form of $B$, but to also construct column vectors $v_{1}, v_{2}, \ldots, v_{n} \in \mathrm{R}^{n \times 1}$ and row vectors $u_{1}, u_{2}, \ldots, u_{n} \in \mathrm{R}^{1 \times n}$ such that $B$ can be expressed as the sum of scaled outer products:

$$
\begin{equation*}
B=T_{n+1}=g_{1} v_{n} u_{n}+g_{2} v_{n-1} u_{n-1}+\cdots+g_{n} v_{1} u_{1} . \tag{5}
\end{equation*}
$$

Note that initially we will work directly over R. After, we will apply the idea of Lemma 2 to avoid the problem of expression swell.

For convenience define $g_{0}=1$. Define $e_{j}=g_{j} / g_{j-1}$ so that $g_{j}=e_{1} e_{2} \cdots e_{j}, 1 \leq j \leq n$. We will construct matrices $B_{1}, B_{2}, \ldots, B_{n+1} \in \mathrm{R}^{n \times n}$ such that

$$
\begin{equation*}
B_{j}=\frac{1}{g_{j-1}}(B-\overbrace{\left(g_{1} v_{n} u_{n}+g_{2} v_{n-1} u_{n-1}+\cdots+g_{j-1} v_{n-j+2} u_{n-j+2}\right)}^{T_{j}}) \tag{6}
\end{equation*}
$$

with

$$
\begin{equation*}
\operatorname{snf}\left(B_{j}\right)=\operatorname{Diag}(\overbrace{e_{j}}^{g_{j} / g_{j-1}}, \overbrace{e_{j} e_{j+1}}^{g_{j+1} / g_{j-1}}, \ldots, \overbrace{e_{j} e_{j+1} \cdots e_{n}}^{g_{n} / g_{j-1}}, 0,0, \ldots, 0) . \tag{7}
\end{equation*}
$$

Considering (7), $\operatorname{snf}\left(B_{n+1}\right)$ will be the zero matrix, showing that (5) will follow from (6) for $j=n+1$.

We now explain how to construct the $B_{j}$ by induction on $j, 1 \leq j \leq n+1$, with base case $j=1$. Initialize $B_{1}=B$. Then $B_{1}$ satisfies (6) and (7). Suppose $B_{j}$ satisfying (6) and (7) have been computed for $1 \leq j \leq i$, for some $i$ with $1 \leq j \leq n$. We show how to compute $B_{i+1}$ satisfying (6) and (7). Because R is a principal ideal ring, there exist vectors $x \in \mathrm{R}^{1 \times n}$ and $y \in \mathrm{R}^{n \times 1}$ such that $e_{i}:=x B_{i} y$, the gcd of all entries of $B_{i}$. (We defer until Sections 5 and 6 to describe how to compute $x$ and $y$.) The nonzero invariant structure of a matrix does not change if we embed into a larger zero matrix. Let $v:=B_{i} y$ and $u:=x B_{i}$, and consider the unimodular transformation of the matrix obtained from $B_{i}$ by augmenting with an initial row and column of zeroes.

$$
\left[\begin{array}{l|l}
1 & x  \tag{8}\\
\hline & I_{n}
\end{array}\right]\left[\begin{array}{l|l}
0 & \\
\hline & B_{i}
\end{array}\right]\left[\begin{array}{l|l}
1 & \\
\hline y & I_{n}
\end{array}\right]=\left[\begin{array}{c|c}
e_{i} & u \\
\hline v & B_{i}
\end{array}\right]
$$

Let $v_{n-i+1}:=v / e_{i}$ and $u_{n-i+1}:=u / e_{i}$, and apply another unimodular transformation to the matrix on the right hand side of (8) to zero out entries below and to the right of $e_{i}$.

$$
\left[\begin{array}{c|c}
1 & -u_{n-i+1}  \tag{9}\\
\hline & I_{n}
\end{array}\right]\left[\begin{array}{c|c}
e_{i} & u \\
\hline v & B_{i}
\end{array}\right]\left[\begin{array}{c|c}
1 & \\
\hline-v_{n-i+1} & I_{n}
\end{array}\right]=\left[\begin{array}{c|c}
e_{i} & \\
\hline & B_{i}-e_{i} v_{n-i+1} u_{n-i+1}
\end{array}\right]
$$

All entries of the matrix on the right hand side of (9) are divisible by $e_{i}$. Let $B_{i+1}=$ $\left(1 / e_{i}\right)\left(B_{i}-e_{i} v_{n-i+1} u_{n-i+1}\right) \in \mathrm{R}^{n \times n}$, matching (6) for $j=i+1$. Since the matrix on the right of (9) has the same nonzero invariant factors as $B_{i}$, we conclude that (7) also holds for $j=i+1$. The procedure just described is summarized by the following code fragment, correctness of which follows by induction.

```
Input: • nonsingular \(B \in \mathrm{R}^{n \times n}\)
Let \(B_{1}=B\).
for \(i\) from 1 to \(n\) do
    Let \(x \in \mathrm{R}^{1 \times n}\) and \(y \in \mathrm{R}^{n \times 1}\) be such that \(x B_{i} y=\operatorname{gcd}\left(B_{i}\right)\).
    \(v:=B_{i} y ; \quad u:=x B_{i} ; \quad e_{i}:=x v ;\)
    \(g_{i}:=e_{1} e_{2} \ldots e_{i} ; \quad v_{n-i+1}:=v / e_{i} ; \quad u_{n-i+1}:=u / e_{i} ;\)
    Let \(B_{i+1}=\frac{1}{e_{i}}\left(B_{i}-e_{i} v_{n-i+1} u_{n-i+1}\right) \in \mathrm{R}^{n \times n}\).
    assert: Equations (6) and (7) hold over R for \(j=i+1\).
od
assert: \(B=g_{1} v_{n} u_{n}+g_{2} v_{n-1} u_{n-1}+\cdots+g_{n} v_{1} u_{1}\).
```

Figure 1: Computing a Smith decomposition
Figure 2 adjust the algorithm in Figure 1 to compute a decomposition in (5) that only holds modulo $s$, where $s$ is a nonzero multiple of $g_{n}$. All operations are performed explicitly

```
Input: • nonsingular \(B \in \mathrm{R}^{n \times n}\)
    - \(s \in \mathrm{R}\), a nonzero multiple of the largest invariant factor of \(B\)
Let \(B_{1}=B\).
for \(i\) from 1 to \(n\) do
    Let \(x \in \mathrm{R}^{1 \times n}\) and \(y \in \mathrm{R}^{n \times 1}\) be such that \(\operatorname{Rem}\left(x B_{i} y, s / g_{i-1}\right)=\operatorname{Rem}\left(\operatorname{gcd}\left(B_{i}, s / g_{i-1}\right), s / g_{i-1}\right)\).
    \(v:=\operatorname{Rem}\left(B_{i} y, s / g_{i-1}\right) ; \quad u:=\operatorname{Rem}\left(x B_{i}, s / g_{i-1}\right) ; \quad e_{i}:=\operatorname{Rem}\left(x v, s / g_{i-1}\right)\);
    if \(e_{i}=0\) then break fi;
    \(g_{i}:=e_{1} e_{2} \ldots e_{i} ; \quad v_{n-i+1}:=v / e_{i} ; \quad u_{n-i+1}:=u / e_{i} ;\)
    Let \(B_{i+1}=\frac{1}{e_{i}}\left(B_{i}-e_{i} v_{n-i+1} u_{n-i+1}\right) \in \mathrm{R}^{n \times n}\).
    assert: Equations (6) and (7) hold modulo \(s / g_{i}\) for \(j=i+1\).
od
assert: \(B \equiv g_{1} v_{n} u_{n}+g_{2} v_{n-1} u_{n-1}+\cdots+g_{i} v_{n-i+1} u_{n-i+1}(\bmod s)\).
```

Figure 2: Computing a Smith decomposition modulo $s$
over R, but Lemma 2 is applied at iteration $i$ to compute modulo $s / g_{i-1}$. There are two subtleties to be aware of.

First, at iteration $i$ the matrix $B_{i}$ is only correct modulo $s / g_{i-1}$. By Lemma 2, we need to compute $e_{i}$ as the gcd of all entries of $B_{i}$ over $\mathrm{R} /\left\langle s / g_{i-1}\right\rangle$ : over R we compute the gcd of $s / g_{i-1}$ with all entries in $B_{i}$, and then reduce modulo $s / g_{i-1}$.

Second, if $s$ is an associate of $g_{n}$, then there exists a minimal $k \leq n$ such that $g_{k}=g_{k+1}=$ $\cdots=g_{n}$, in which case $B_{k}$ is congruent the zero matrix modulo $s / g_{k-1}$ : this is handled by exiting the loop early since the remaining part of the sum $g_{k} v_{n-k+1}+g_{k+1} v_{n-k} u_{n-k}+\cdots+$ $g_{n} v_{1} u_{1}$ in (5) is known to congruent to zero modulo $s$ in this case.

## 4 The outer product adjoint formula

Let R be a principal ideal domain and let $A \in \mathrm{R}^{n \times n}$ be nonsingular with Smith form $S=$ $U A V=\operatorname{Diag}\left(s_{1}, s_{2}, \ldots, s_{n}\right)$. Let $v_{i}$ and $u_{i}$ be column $i$ and row $i$ of the unimodular matrices $V$ and $U$ respectively, $1 \leq i \leq n$. Inverting both sides of the equation $S=U A V$, multiplying by $s_{n}$, and solving for $s_{n} A^{-1}$ gives

$$
\begin{equation*}
s_{n} A^{-1}=V\left(s_{n} S^{-1}\right) U=\frac{s_{n}}{s_{n}} v_{n} u_{n}+\frac{s_{n}}{s_{n-1}} v_{n-1} u_{n-1}+\cdots+\frac{s_{n}}{s_{1}} v_{1} u_{1} . \tag{10}
\end{equation*}
$$

Note that $\operatorname{snf}\left(s_{n} A^{-1}\right)=\operatorname{Diag}\left(s_{n} / s_{n}, s_{n} / s_{n-1}, \ldots, s_{n} / s_{1}\right)$. Now consider taking equation (10) modulo $s_{n}$. Since each outer product $v_{i} u_{i}$ is scaled by $s_{n} / s_{i}$, the equation will still hold modulo $s_{n}$ if entries in $v_{i}$ and $u_{i}$ are reduced modulo $s_{i}, 1 \leq i \leq n$. This idea is made precise by the following definition.

Definition 3. An outer product adjoint formula of a nonsingular $A \in \mathrm{R}^{n \times n}$ is set of tuples $\left(s_{n-i+1}, v_{n-i+1}, u_{n-i+1}\right)_{1 \leq i \leq k}$ such that:

- the Smith form of $A$ is $\operatorname{Diag}\left(1,1, \ldots, 1, s_{n-k+1}, s_{n-k+2}, \ldots, s_{n}\right)$ with $s_{n-k+1} \neq 1$,
- $v_{n-i+1} \in \mathcal{R}\left(\mathrm{R}, s_{n-i+1)}\right)^{n \times 1}$ and $u_{n-i+1} \in \mathcal{R}\left(\mathrm{R}, s_{n-i+1}\right)^{1 \times n}$ for $1 \leq i \leq k$,
- $s_{n} A^{-1} \equiv \frac{s_{n}}{s_{n}} v_{n} u_{n}+\frac{s_{n}}{s_{n-1}} v_{n-1} u_{n-1}+\cdots+\frac{s_{n}}{s_{n-k+1}} v_{n-k+1} u_{n-k+1}\left(\bmod s_{n}\right)$.

Algorithm OuterProductAdjoint shown in Figures 4 will compute an outer product adjoint formula. The algorithm is identical to that shown in Figure 2 except with $s=s_{n}$ and $g_{i}=s_{n} / s_{n-i+1}$.

```
OuterProductAdjoint \(\left(A, s_{n}\right)\)
Input: • nonsingular \(A \in \mathrm{R}^{n \times n}\)
    - \(s_{n} \in \mathrm{R}\), the largest invariant factor of \(A\)
Output: An outer product adjoint formula of \(A\).
Let \(B_{1}=s_{n} A^{-1}\) and \(s_{n+1}=s_{n}\).
for \(i\) to \(n\) do
    Let \(x \in \mathrm{R}^{1 \times n}\) and \(y \in \mathrm{R}^{n \times 1}\) such that \(\operatorname{Rem}\left(x B_{i} y, s_{n-i+2}\right)=\operatorname{Rem}\left(\operatorname{gcd}\left(B_{i}, s_{n-i+2}\right), s_{n-i+2}\right)\).
    \(u:=\operatorname{Rem}\left(x B_{i}, s_{n-i+2}\right) ; \quad v:=\operatorname{Rem}\left(B_{i} y, s_{n-i+2}\right) ; \quad e_{i}:=\operatorname{Rem}\left(x v, s_{n-i+2}\right)\);
    if \(e_{i}=0\) then break fi;
    \(s_{n-i+1}:=s_{n-i+2} / e_{i} ; \quad u_{n-i+1}:=u / e_{i} ; \quad v_{n-i+1}:=v / e_{i} ;\)
    Let \(B_{i+1}=\frac{1}{e_{i}}\left(B_{i}-e_{i} v_{n-i+1} u_{n-i+1}\right) \in \mathrm{R}^{n \times n}\).
od;
return \(\left(s_{n-j+1}, v_{n-j+1}, u_{n-j+1}\right)_{1 \leq j \leq i-1}\)
```

Figure 3: Algorithm OuterProductAdjoint
When we specialize to the rings $\mathrm{R}=\mathbb{Z}$ and $\mathrm{R}=\mathrm{K}[x]$ most of the above loop will be implemented exactly as written. Note that the vector $y$ that needs to be chosen at the start of each iteration is required only to construct $v$. We will appeal to known results to construct $v$ as an R-linear combination of $B_{i} Y$ for a $Y$ that is randomly chosen so that $\operatorname{gcd}\left(B_{i} Y, s_{n-i+2}\right)=\operatorname{gcd}\left(B_{i}, s_{n-i+1}\right)$ with high probability. Once $v$ is known, $x$ is computed computed as the solution to an extended euclidean problem.

The main technical issue we face is to efficiently compute $\operatorname{Rem}\left(B_{i} Y, s_{n-i+2}\right)$ for a $Y \in$ $\mathcal{R}\left(\mathrm{R}, s_{n-i+1}\right)^{n \times m}$. (The problem of computing a vector-matrix product $\operatorname{Rem}\left(x B_{i}, s_{n-i+2}\right)$ is analogous so we focus here on the matrix-vector case.) Exactly as in Section 3, but here with $s=s_{n}$ and $g_{i}=s_{n} / s_{n-i+1}$, if we define

$$
\begin{equation*}
T_{i}=\frac{s_{n}}{s_{n}} v_{n} u_{n}+\frac{s_{n}}{s_{n-1}} v_{n-1} u_{n-1}+\cdots+\frac{s_{n}}{s_{n-i+2}} v_{n-i-1} u_{n-i-1} \in \mathrm{R}^{n \times n}, \tag{11}
\end{equation*}
$$

then at loop iteration $i=1,2, \ldots, k$ we work with the matrix

$$
\begin{equation*}
B_{i}=\frac{s_{n-i+2}}{s_{n}}\left(s_{n} A^{-1}-T_{i}\right) \in \mathrm{R}^{n \times n} \tag{12}
\end{equation*}
$$

and modulus $s_{n-i+2}$, where $s_{n+1}$ is defined to be $s_{n}$. $\operatorname{Rem}\left(T_{i} Y, s_{n}\right)$ can be computed efficiently for given $Y \in \mathcal{R}\left(\mathrm{R}, s_{n}\right)^{n \times m}$ by using nested multiplication.

$$
\begin{aligned}
T_{i} Y & \equiv\left(\sum_{j=1}^{i-1} \frac{s_{n}}{s_{n-j+1}} v_{n-j+1} u_{n-j+1}\right) Y\left(\bmod s_{n}\right) \\
& \equiv \sum_{j=1}^{i-1} \frac{s_{n}}{s_{n-j+1}} \underbrace{\operatorname{Rem}(v_{n-j+1}(u_{n-j+1} \overbrace{\operatorname{Rem}\left(Y, s_{n-j+1}\right)}^{Y_{n-j+1}}), s_{n-j+1})}_{X_{n-j+1}} \\
& \equiv \frac{s_{n}}{s_{n}}\left(X_{n}+\cdots+\frac{s_{n-i+5}}{s_{n-i+4}}\left(X_{n-i+4}+\frac{s_{n-i+4}}{s_{n-i+3}}\left(X_{n-i+3}+\frac{s_{n-i+3}}{s_{n-i+2}} X_{n-i+2}\right)\right) \cdots\right)
\end{aligned}
$$

Algorithm OuterProdMul shown in Figures 4 implements the above scheme. In the first loop, iteration $j$ does $O(n m)$ arithmetic operations modulo $s_{n-j+1}$ with operands from $\mathcal{R}\left(\mathrm{R}, s_{n-j+1}\right)$ and $\mathcal{R}\left(\mathrm{R}, s_{n-j+2}\right)$. Iteration $j$ of the second loop does $O(n m)$ operations modulo $s_{n-j+1}$ with operands from $\mathcal{R}\left(\mathrm{R}, s_{n-j+1}\right)$. Lemma 4 follows from the superlinearity of M.

```
OuterProdMul \(\left(s,\left(s_{n-j+1}, v_{n-j+1}, u_{n-j+1}\right)_{1 \leq j \leq i-1}, Y\right)\)
Input: • nonzero \(s \in \mathcal{A}(\mathrm{R})\)
    - \(v_{n-j+1} \in \mathcal{R}\left(\mathrm{R}, s_{n-j+1}\right)^{n \times 1}\) and \(u_{n-j+1} \in \mathcal{R}\left(\mathrm{R}, s_{n-j+1}\right)^{1 \times n}, 1 \leq j \leq i-1\)
    - \(s_{n-i+2}\left|s_{n-i+3}\right| \cdots \mid s_{n}\) with \(s_{n}=s\) and \(s_{n-i+2}\) a nonunit if \(i>1\)
    - \(Y \in \mathcal{R}(\mathrm{R}, s)^{n \times m}\)
Output: \(\operatorname{Rem}\left(\left(\sum_{j=1}^{i-1}\left(s / s_{n-j+1}\right) v_{n-j+1} u_{n-j+1}\right) Y, s\right)\).
if \(i \leq 1\) then return the \(n \times m\) zero matrix \(\mathbf{f i}\);
\(Y_{n+1}:=Y ; \quad s_{n+1}:=s_{n} ;\)
for \(j\) from 1 to \(i-1\) do
    \(Y_{n-j+1}:=\operatorname{Rem}\left(Y_{n-j+2}, s_{n-j+1}\right) ; \quad X_{n-j+1}:=\operatorname{Rem}\left(v_{n-j+1}\left(u_{n-j+1} Y_{n-j+1}\right), s_{n-j+1}\right)\)
od;
\(V:=X_{n-i+2}\);
for \(j\) from \(i-2\) downto 1 do \(V:=\operatorname{Rem}\left(X_{n-j+1}+\left(s_{n-j+1} / s_{n-j}\right) V, s_{n-j+1}\right)\) od;
return \(V\)
```

Figure 4: Algorithm OuterProdMul

Lemma 4. Algorithm 4 (DuterProdMul) is correct. The cost of the algorithm is

$$
\begin{array}{ll}
\mathrm{R}=\mathbb{Z}: & O\left(n m \mathrm{M}\left(\log \prod_{i=1}^{k} s_{n-i+1}\right)\right) \text { bit operations } \\
\mathrm{R}=\mathrm{K}[x]: & O\left(n m \mathrm{M}\left(\sum_{i=1}^{k} \operatorname{deg} s_{n-i+1}\right)\right) \text { field operations from } \mathrm{K} .
\end{array}
$$

Now consider the computation of $\operatorname{Rem}\left(B_{i} Y, s_{n-i+2}\right)$. Notice that that the matrices $B_{i}$ are not explicitly computed. Instead, from the definition of $B_{i}$ in (12) we have that

$$
\operatorname{Rem}\left(B_{i} Y, s_{n-i+2}\right)=\operatorname{Rem}\left(\frac{s_{n-i+2}}{s_{n}}\left(s_{n} A^{-1} Y-\operatorname{Rem}\left(T_{i} Y, s_{n}\right)\right), s_{n-i+2}\right)
$$

Let $N \in \mathrm{R}$ be relatively prime to $s_{n}$. If $\operatorname{Rem}\left(B_{i} Y, N\right)=B_{i} Y$ then also

$$
\operatorname{Rem}\left(\operatorname{Rem}\left(B_{i} Y, N\right), s_{n-i+1}\right)=\operatorname{Rem}\left(B_{i} Y, s_{n-i+1}\right)
$$

and we can apply the following recipe.

$$
\operatorname{Rem}\left(B_{i} Y, s_{n-i+2}\right)=\left[\begin{array}{l}
W_{1}:=\operatorname{Rem}\left(A^{-1} Y, N\right) ;  \tag{13}\\
W_{2}:=\operatorname{OuterProdMul}\left(s,\left(s_{n-j+1}, v_{n-j+1}, u_{n-j+1}\right)_{1 \leq j \leq i-1}, Y\right) ; \\
\text { return } \operatorname{Rem}\left(\operatorname{Rem}\left(s_{n-i+2} W_{1}-\left(s_{n-i+2} / s_{n}\right) W_{2}, N\right), s_{n-i+2}\right)
\end{array}\right]
$$

Theorem 5. If $Y \in \mathcal{R}\left(\mathrm{R}, s_{n-i+2}\right)^{n \times m}$ then recipe (13) returns $\operatorname{Rem}\left(B_{i} Y, s_{n-i+2}\right)$ if

$$
\begin{array}{ll}
\mathrm{R}=\mathbb{Z}: & N \geq s_{n-i+2}\left(n s_{n-i+2}\left\|A^{-1}\right\|+1\right)+2 \\
\mathrm{R}=\mathrm{K}[x]: & \operatorname{deg} N \geq \max \left(\operatorname{deg} s_{n-i+2}, \operatorname{deg} s_{n} A^{-1}+\operatorname{deg} s_{n-i+2}\right) .
\end{array}
$$

Proof. The matrices $s_{n} A^{-1} Y$ and $\operatorname{Rem}\left(T_{i} Y, s_{n}\right)$ are over R and their difference is divisible by $s_{n} / s_{n-i+2}$. The result will follow if we show that $N$ satisfies the bounds of Lemma 1, with $B_{i} Y$ playing the role of $a$.

Over $\mathrm{K}[x]$ we have $\operatorname{deg} s_{n} A^{-1} Y \leq \operatorname{deg} s_{n} A^{-1}+\operatorname{deg} Y \leq \operatorname{deg} s_{n} A^{-1}+\operatorname{deg} s_{n}-1$ and $\operatorname{deg} \operatorname{Rem}\left(T_{i} Y, s_{n}\right) \leq \operatorname{deg} s_{n}-1$. The bound for $\operatorname{deg} N$ given in theorem is obtained by adding one to the maximum of these two bounds and subtracting $\operatorname{deg} s_{n} / s_{n-i+2}$.

Over $\mathbb{Z}$ we have $\left\|s_{n} A^{-1} Y\right\| \leq n s_{n}\left\|A^{-1} \mid\right\|\|Y\| \leq n s_{n}\left\|A^{-1}\right\| s_{n-i+2} / 2$ and $\left\|\operatorname{Rem}\left(T_{i} Y, s_{n}\right)\right\| \leq$ $s_{n} / 2$, hence the difference of these matrices multiplied by $s_{n-i+2} / s_{n}$ has entries bounded in magnitude by $n s_{n-i+2}\left\|A^{-1}\right\|\|Y\|+s_{n-i+2} / 2$; the bound for $N$ in the theorem is obtained by multiplying this bound two and adding two.

## 5 Computing the inverse of an integer matrix

Algorithm IntInverse is shown in Figure 5. Phase 1 uses randomization to probabilistically compute the largest invariant factor of the input matrix $A \in \mathbb{Z}^{n \times n}$, together with a bound for $\left\|A^{-1}\right\|$. Phase 2 adapts algorithm OuterProductAdjoint shown in Figure 3 by incorporating randomization to obtain a probabilistic algorithm for computing the outer product adjoint of $A$. Phase 3 certifies correctness of the computed outer product adjoint formula from phase two and computes the explicit inverse of $A$. In the following subsections we fill in the implementation details and prove probability results and complexity estimates for each phase. First we recall some known results.

By the denominator of a rational matrix or vector we mean the smallest positive integer multiple that will clear the denominators, assuming the fractions are reduced.

```
IntInverse( }A\mathrm{ )
```

Input: Nonsingular $A \in \mathbb{Z}^{n \times n}$.

Output: $A^{-1}$ or Fail. Fail will be returned with probability $<1 / 2$.

## 1. [Initialization]

Choose $\bar{p}$ uniformly and randomly from among the first $12\left\lfloor\log _{2}\left(n^{n / 2}| | A \|^{n}\right)\right\rfloor$ primes; $p:=$ the smallest positive integer power of $\bar{p}$ such that $p \geq \log \sqrt{n}+\log \|A\|$;
if $\bar{p} \not \perp \operatorname{det} A$ return Fail else $C:=\operatorname{Rem}\left(A^{-1}, p\right)$ fi;
$\alpha:=\left\|A^{-1} X\right\|$, where $X \in\{-1,1\}^{n \times 4}$ is chosen uniformly and randomly;
$M:=6+\left\lceil 2 n\left(\log _{2} n+\log _{2}\|A\|\right)\right\rceil ; \quad L:=\{0, \ldots, M-1\}$;
$s:=$ the denominator of $A^{-1} X$, where $X \in L^{n \times 12}$ is chosen uniformly and randomly;
$m:=2\left\lceil\left(2+\log _{2} n\right) / \log _{2} 3\right\rceil ;$
2. [Compute Outer Product Adjoint Formula]

Let $B_{1}$ denote $s A^{-1}$ and $s_{n+1}=s$.
for $i$ to $n$ do
Choose $Y \in L^{n \times m}$ uniformly and randomly;
$Y:=\operatorname{Rem}\left(Y, s_{n-i+2}\right)$;
Set $N:=p^{k}$, where $k \in \mathbb{Z}_{>0}$ is minimal s.t. $p^{k} \geq s_{n-i+2}\left(n \alpha s_{n-i+2}+1\right)+2$;
Compute $V:=\operatorname{Rem}\left(B_{i} Y, s_{n-i+2}\right)$ using recipe (13);
Use Lemma 9 to compute $x$ and $y$ such that $\operatorname{gcd}\left(x V y, s_{n-i+2}\right)=\operatorname{gcd}\left(V, s_{n-i+2}\right)$;
$v:=\operatorname{Rem}\left(V y, s_{n-i+2}\right)$;
Compute $u:=\operatorname{Rem}\left(B_{i}^{T} x^{T}, s_{n-i+2}\right)^{T}$ using recipe (13);
$e_{i}:=\operatorname{Rem}\left(x v, s_{n-i+2}\right) ;$ if $e_{i}=0$ then break else $s_{n-i+1}:=s_{n-i+2} / e_{i} \mathbf{f i}$;
if ( $i=1$ and $s_{n} \neq s$ ) or $\prod_{j=1}^{i} s_{n-j+1}>n^{n / 2}\|A\|^{n}$ or $e_{i} \nmid u$ then return Fail fi;
$u_{n-i+1}:=u / e_{i} ; \quad v_{n-i+1}:=v / e_{i} ;$
Let $B_{i+1}$ denote $\frac{1}{e_{i}}\left(B_{i}-e_{i} v_{n-i+1} u_{n-i+1}\right) \in \mathbb{Z}^{n \times n}$.
od
3. [Compute Inverse and Assay Correctness]
if OuterProductMul $\left(s,\left(s_{n-j+1}, v_{n-j+1}, u_{n-j+1}\right)_{1 \leq j \leq i-1}, A\right) \neq 0_{n \times n}$ then return Fail fi; $C_{0}:=$ OuterProductMul $\left(s,\left(s_{n-j+1}, v_{n-j+1}, u_{n-j+1}\right)_{1 \leq j \leq i-1}, I_{n}\right)$;
$N_{1}:=p^{k}$, where $k \in \mathbb{Z}_{>0}$ is minimal s.t. $p^{k} s \geq 2 \alpha s+2$;
$C_{1}:=\operatorname{Rem}\left(s \operatorname{Rem}\left(A^{-1}, N_{1}\right), N_{1}\right)$, computed using $p$-adic lifting;
$C:=\operatorname{Rem}\left(N_{1} \operatorname{Rem}\left(1 / N_{1}, s\right) C_{0}+s \operatorname{Rem}\left(1 / s, N_{1}\right) C_{1}, N_{1} s\right) ;$
$N_{2}:=p^{k}$, where $k \in \mathbb{Z}_{>0}$ is minimal s.t. $p^{k} \geq(2 n / s)\|A\| \||C| \mid+2$;
if $\operatorname{Rem}\left(A \operatorname{Rem}\left((1 / s) C, N_{2}\right), N_{2}\right) \neq I_{n}$ then return Fail fi;
return $(1 / s) C$, with fractions reduced

Figure 5: Algorithm IntInverse

Lemma 6 ([5, Theorem 2.1]). Let $A \in \mathbb{Z}^{n \times n}$ be nonsingular. If $X \in L^{n \times 2}$ is chosen uniformly and randomly, where $L:=\{0,1, \ldots, M-1\}$ with $M:=6+\left\lceil 2 n\left(\log _{2} n+\log _{2}\|A\|\right)\right\rceil$, then the denominator of $A^{-1} X$ is the largest invariant factor of $A$ with probability at least $1 / 3$.

An inspection of the proof of [5, Theorem 2.1] reveals that the definition of $M$ in Lemma 6 is driven by the size of $s_{n}$, the largest invariant factor of $A$, and is otherwise independent of $\|A\|$ and $n$. Since $s_{n}$ is a factor of $\operatorname{det} A$, we have $s_{n} \leq \operatorname{det} A \leq n^{n / 2}\|A\|^{n}$, the second inequality being implied by Hadamard's bound. The next result follows as a corollary of the proof of [5, Theorem 2.1].
Corollary 7. Let $s \in \mathbb{Z}_{>0}$ satisfy the bound $s \leq n^{n / 2}\|A\|^{n}$. For any $B \in \mathbb{Z}^{n \times n}$, if $X \in L^{n \times 2}$ is chosen uniformly and randomly, then $\operatorname{gcd}(B X, s)=\operatorname{gcd}(B, s)$ with probability at least $1 / 3$.

The main computational tool in our algorithm is linear $p$-adic lifting [3, 15], used to compute $\operatorname{Rem}\left(A^{-1} X, p^{k}\right)$ for a given $X \in \mathbb{Z}^{n \times m}$ and $k$. The presentation and analysis in [3, 15] assumes standard integer arithmetic, but if the bitlength of $p$ is well chosen then fast integer arithmetic can be incorporated without much difficulty by appealing to fast algorithms for arithmetic operations such as radix conversion. We refer to [16, Section 5] for a derivation of parts (a) and (b) of the following.

Lemma 8. Let $A \in \mathbb{Z}^{n \times n}$ be nonsingular. Suppose we have $p \in \mathbb{Z}_{>0}$ with $p \perp \operatorname{det} A$ and $\log p \in \Theta(\log n+\log \|A\|)$, together with $C:=\operatorname{Rem}\left(A^{-1}, p\right)$.
(a) If $N:=p^{k}$ and $Y \in \mathbb{Z}^{n \times m}$ satisfies $\log \|Y\| \in O(\log N)$, then $\operatorname{Rem}\left(A^{-1} Y, N\right)$ can be computed with $O\left(n^{2} k m \mathrm{~B}(\log n+\log \|A\|)+n m \mathrm{~B}(k(\log n+\log \|A\|))\right.$ bit operations.
(b) If $Y \in \mathbb{Z}^{n \times m}$ satisfies $\log \|Y\| \in O(n(\log n+\log \|A\|))$ then $A^{-1} Y$ can be computed with $O\left(n^{3} m \mathrm{~B}(\log n+\log \|A\|)\right)$ bit operations.
(c) Consider recipe (13). If $\prod_{j=1}^{i-1} s_{n-i+1} \leq n^{n / 2}\|A\|^{n}$ and $\log N \in O\left(\log s_{n-i+2}+\log n+\right.$ $\left.\log \left\|A^{-1}\right\|\right)$, then $\operatorname{Rem}\left(B_{i} Y, N\right)$ can be computed in

$$
O\left(n m \mathrm{~B}(n(\log n+\log \|A\|))+n^{2} k_{n-i+2} m \mathrm{~B}(\log n+\log \|A\|)\right)
$$

bit operations, where

$$
k_{n-i+2}:=1+\frac{\log s_{n-i+2}}{\log n+\log \|A\|}+\frac{\log \kappa(A)}{\log n+\log \|A\|}
$$

with $\kappa(A):=\left\|A^{-1}\right\|\|A\|$.
Proof. For a detailed derivation of parts (a) and (b) we refer to [16, Section 5]. To understand part (c) it will be helpful to recall the cost analysis of part (a). Each step of $p$-adic lifting involves two matrix $\times$ matrix products: $A \times *$ and $C \times *$ where $*$ is an $n \times m$ matrix with entries bounded in bitlength by $O(\log p)$. This gives rise to the $O\left(n^{2} k m \mathrm{~B}(\log n+\log \|A\|)\right)$ term. The second term bounds the cost of all the additional work, especially, at the end
of the lifting, the radix conversion from $p$-adic to standard representation of $n m$ numbers bounded in bitlength by $\log N$.

Now consider part (c). Algorithm OuterProdMul computes $W_{2}$ in $O(n m \mathrm{M}(n(\log n+$ $\log \|A\|))$ ) bit operations (Lemma 4). This also bounds the cost of the arithmetic operations performed in the return statement of recipe (13). The cost of computing $W_{1}:=$ $\operatorname{Rem}\left(A^{-1} Y, N\right)$ is as stated in (a) with $k=\log _{p} N$. The definition of $k_{n-i+2}$ is such that $k_{n-i+2} \geq 1$ and $k_{n-i+2} \in \Theta\left(\log _{p} N\right)$. The cost estimate stated in part (c) simplifies the second term of the cost estimate from part (a) by using $k_{n-i+2} \in O(n)$.

The usual extended euclidean problem takes as input a column vector $v \in \mathbb{Z}^{n \times 1}$ and asks for a row vector $x \in \mathbb{Z}^{1 \times n}$ such that $x v=\operatorname{gcd}(v)$. The next lemma recalls a method for solving deterministically a variation of the problem when the input is a matrix.

Lemma 9. Given $V \in \mathcal{R}(\mathbb{Z}, s)^{n \times m}$, an $x \in \mathcal{R}(\mathbb{Z}, s)^{1 \times n}$ and $y \in \mathcal{R}(\mathbb{Z}, s)^{m \times 1}$ such that $\operatorname{gcd}(x V y, s)=\operatorname{gcd}(V, s)$ can be computed in $O(n m \mathrm{~B}(\log s))$ bit operations.

Proof. In [22, Section 4.1] a method is derived to compute a $y$ such that $\operatorname{gcd}(V y, s)=$ $\operatorname{gcd}(V, s)$. The cost of the method is $O(n m \mathrm{~B}(\log s))$ bit operations, plus $m$ calls to a subroutine that takes as input integers $a$ and $b$, and computes a $c$ such that $\operatorname{gcd}(a+b c, s)=$ $\operatorname{gcd}(a, b, c)$ : each such $c$ can be computed in $O(\mathrm{~B}(\log s))$ bit operations using operation Stab from [25]. Once $y$ has been computed, compute $x$ to be a solution to the standard extended euclidean problem with input $V y$.

## Phase 1

Phase one uses randomization to compute three numbers probabilistically: a small prime power $p$ that is relatively prime to $\operatorname{det} A$; an $\alpha$ that satisfies $\left\|A^{-1}\right\| \leq \alpha \leq n\left\|A^{-1}\right\|$; the largest invariant factor $s=s_{n}$ of the input matrix. Part (a) of the following lemma gives some properties that are guaranteed to hold by construction, while part (b) gives properties that only hold probabilistically; this distinction is important for our cost analysis of phases 2 and 3 of the algorithm.

Lemma 10. Phase 1 of algorithm IntInverse uses $O(n(\log n+\log \log \|A\|))$ random bits and completes in $O\left(n^{3} \mathrm{~B}(\log n+\log \|A\|)\right)$ bit operations. Fail will be returned with probability at most 1/12. If Fail is not returned, then (a) will hold, and (b) will hold with overall probability at least $1-1 / 6$.
(a) $p \perp \operatorname{det} A, \log p \geq \log \sqrt{n}+\log \|A\|$ and $\log p \in \Theta(\log n+\log \|A\|), \alpha \leq n\left\|A^{-1}\right\|$, and $s$ is a divisor of the largest invariant factor of $A$.
(b) $\alpha \geq\left\|A^{-1}\right\|$, and $s$ is the largest invariant factor of $A$.

Proof. By Hadamard's bound $|\operatorname{det} A| \leq n^{n / 2}\|A\|^{n}$, so $\operatorname{det} A$ is divisible by at most the log of this many distinct prime divisors. This shows that the randomly chosen prime $\bar{p}$ will be a divisor of $\operatorname{det} A$ with probability at most $1 / 12$. Assume henceforth that Fail has not been returned.

First consider part (a). From the prime number theorem we know that $\log \bar{p} \in \Theta(\log n+$ $\log \log \|A\|)$; the upper bound and lower bounds for $\log p$ follow by construction. The upper bound for $\alpha$ uses $\|X\|=1:\left\|A^{-1} X\right\| \leq n\left\|A^{-1}\right\|\|X\|=n\left\|A^{-1}\right\|$. In [1] it is shown that $s$ is a factor of $s_{n}$.

Now consider part (b). We will separately bound the probability that that $\alpha<\left\|A^{-1}\right\|$ by $1 / 16$, and that $s$ is a proper divisor of the largest invariant factor of $A$ by $64 / 729$. Since the sum of these two probabilities is less than $1 / 6$, the claim in the lemma that part (b) will hold with probability at least $1-1 / 6$ will follow.

To see that $\alpha<\left\|A^{-1}\right\|$ with probability at most $1 / 16$, let $x=\left[x_{1} \mid \bar{x}\right]^{T} \in\{-1,1\}^{n \times 1}$ be chosen uniformly and randomly, and let $a=\left[a_{1} \mid \bar{a}\right] \in \mathbb{Q}^{1 \times n}$ be a row of $A^{-1}$ which has a maximal magnitude entry, which without loss of generality we will assume is the principal entry $a_{1}$. Consider the dot product $a x=a_{1} x_{1}+\bar{a} \bar{x}$. A sufficient condition for $|a x| \geq\left\|A^{-1}\right\|$ to hold is that $\operatorname{sign}\left(a_{1} x_{1}\right)=\operatorname{sign}(\bar{a} \bar{x})$; since $x_{1}$ is chosen uniformly and randomly from $\{-1,1\}$, the probability that $|a x|<\left\|A^{-1}\right\|$ is less than $1 / 2$. Choosing $X \in\{-1,1\}^{n \times 4}$ to have four columns as in the algorithm gives four independent trials of this idea, so $\alpha<\left\|A^{-1}\right\|$ with probability less than $1 / 2^{4}$.

Now consider $s$. Since $X$ is chosen from $L^{n \times(2 \times 6)}$, the probability that $s$ is a proper divisor of the largest invariant factor of $A$ is bounded by the probability that 6 independent trials of the approach of Lemma 6 all fail: this is bounded by $(2 / 3)^{6}=64 / 729$.

Now consider the running time. The first $12\left\lfloor\log _{2}\left(n^{n / 2}\|A\|^{n}\right)\right\rfloor$ primes can be constructed in the allotted time using the sieve of Eratosthenes (see [14, Section 4.5.4]). The modular inverse $C$ can be computed in the allotted time using Gaussian elimination. By Lemma 8, part (b), the rational system solutions $A^{-1} X$ can be computed in the allotted time using $p$-adic lifting.

## Phase 2

Phase 2 of algorithm IntInverse adapts algorithm OuterProductAdjoint shown in Figure 3 to the integer case. The main change is that at iteration $i$ we recover the gcd $e_{i}$ of all entries in $B_{i}$ probabilistically by computing $\operatorname{gcd}\left(B_{i} Y, s_{n-i+2}\right)$ for a randomly chosen $Y$. Not only might some of the $e_{i}$ be computed incorrectly, phase 2 also depends on the numbers $s$ and $\alpha$ that are computed probabilistically in phase 1 (cf. part (b) of Lemma 8). For this reason, we need to to bound the running time of phase 2 independently of the correctness of $s, \alpha$, and the $e_{i}$ computed at each loop iteration.

Lemma 11. Phase 2 of algorithm IntInverse uses $O\left(n^{2}(\log n)(\log n+\log \log \|A\|)\right)$ random bits and completes in

$$
\begin{equation*}
O\left(n^{2}(\log n) \mathrm{B}(n(\log n+\log \|A\|))+n^{3}(\log n)(\log \kappa(A)) \frac{\mathrm{B}(\log n+\log \|A\|)}{\log n+\log \|A\|}\right) \tag{14}
\end{equation*}
$$

bit operations.

Proof. Each loop iteration checks that $\prod_{j=1}^{i} s_{n-j+1} \leq n^{n / 2}\|A\|^{n}$, so the sum of the bitlengths of the moduli $s_{n-i+2}$ over all loop iterations is bounded by $O(n(\log n+\log \|A\|))$. Excluding the calls to recipe (13), a total cost bound of $O(n m \mathrm{~B}(n(\log n+\log \|A\|)))$ bit operations for the entire phase follows from the superlinearity of $\mathrm{B}: \sum_{j=1}^{i-1} \mathrm{~B}\left(\log s_{n-j+1}\right) \in$ $O\left(\mathrm{~B}\left(\sum_{j=1}^{i-1} \log s_{i}\right)\right)$.

The cost of the two calls to recipe (13) for single loop iteration is given by part (c) of Lemma 8. Let $k_{n-i+2}$ be defined as in Lemma 8. Then

$$
\sum_{j=1}^{i-1} k_{n-i+2} \in O\left(n+\frac{n \log \kappa(A)}{\log n+\log \|A\|}\right) .
$$

The bound (14) for the overall cost of all calls to recipe (13) follows.
Lemma 12. If $s$ is the largest invariant factor of $A$ and $\alpha \geq\left\|A^{-1}\right\|$, then phase 2 of algorithm IntInverse will not return Fail and computes a correct outer product adjoint formula for $A$ with probability at least $1-1 / 4$.
Proof. Phase 2 is identical to algorithm OuterProductAdjoint shown in Figure 3 except that $e_{i}$ is computed probabilistically. Corollary 7 gives that $e_{i} \neq \operatorname{gcd}\left(B_{i}, s\right)$ with probability at most $(2 / 3)^{m / 2}$. Since $m$ is chosen so that $(2 / 3)^{m / 2} \leq 1 /(4 n)$, the probability that $e_{i} \neq$ $\operatorname{gcd}\left(B_{i}, s_{n-i+2}\right)$ for one or more $i$ is at most $1 / 4$.

## Phase 3

Lemma 13. If $s$ is the largest invariant factor of $A, \alpha \geq\left\|A^{-1}\right\|$, and the quantities $\left(s_{n-j+1}, v_{n-j+1}, u_{n-j+1}\right)_{1 \leq j \leq i-1}$ computed in phase 2 comprise an outer product adjoint formula for A, then phase 3 of algorithm IntInverse will not return Fail.
Proof. Assume that $s=s_{n}, \alpha \geq\left\|A^{-1}\right\|$ and $\left(s_{n-j+1}, v_{n-j+1}, u_{n-j+1}\right)_{1 \leq j \leq i-1}$ is an outer product adjoint formula, as specified in the lemma. Then the first call to OuterProdMul computes $\operatorname{Rem}\left(s A^{-1} A, s\right)$ which will be the zero matrix and the second call computes $C_{0}=$ $\operatorname{Rem}\left(s A^{-1}, s\right)$. The matrix $C_{1}$ is equal to $\operatorname{Rem}\left(s A^{-1}, N_{1}\right)$. The matrix $C$ is obtained by Chinese remaindering together $C_{0}$ and $C_{1}$ to get $C=\operatorname{Rem}\left(s A^{-1}, N_{1} s\right)$. Since $\left\|s A^{-1}\right\| \leq s \alpha$, and $N_{1} s \geq 2 \alpha s+2$, by Lemma 1 we have $s A^{-1}=\operatorname{Rem}\left(s A^{-1}, N_{1} s\right)$ and we conclude that $C=s A^{-1}$. It follows that the second last line will not return Fail since $(1 / s) A C$ is indeed the identity matrix.

Lemma 14. If phase 3 of algorithm IntInverse does not return Fail, then the correct inverse of $A$ is returned.
Proof. If the first call to OuterProductAdjoint does not return Fail, then $C_{0}$ as computed in the next line has the property that $\operatorname{Rem}\left(A C_{0}, s\right)$ is the zero matrix. By construction of $C, \operatorname{Rem}(C, s)=C_{0}$, so $\operatorname{Rem}(A C, s)$ is also the zero matrix: we conclude that $A C$ is divisible by $s$. An a priori bound for $\|(1 / s) A C\|$ is $(n / s)\|A\| C \|$. By Lemma 1, the choice of $N_{2}$ guarantees that $(1 / s) A C=\operatorname{Rem}\left((1 / s) A C, N_{2}\right)$. If $(1 / s) A C=I_{n}$ then $(1 / s) C$ is indeed the inverse of $A$.

Lemma 15. Phase 3 of algorithm IntInverse completes in

$$
O\left(n^{2} \mathrm{~B}(n(\log n+\log \|A\|))+n^{3}(\log n+\log \kappa) \frac{\mathrm{B}(\log n+\log \|A\|}{\log n+\log \|A\|}\right)
$$

## bit operations.

Proof. The cost of the two calls to algorithm OuterProductAdjoint is $O\left(n^{2} \mathrm{~B}(n(\log n+\right.$ $\log \|A\|)$ )) bit operations (Lemma 3). Since $\alpha \leq n\left\|A^{-1}\right\|$ and $s \leq n^{n / 2}\|A\|^{n}$, all of $N_{1}, N_{1} s$ and $N_{2}$ will have bitlength bounded by $O(n(\log n+\log \|A\|))$. Thus, the cost of computing $C$, $\operatorname{Rem}\left((1 / s) C, N_{2}\right)$, and reducing the fractions in $(1 / s) C$ is bounded by $O\left(n^{2} \mathrm{~B}(n(\log n+\right.$ $\log \|A\|))$ ) bit operations as well.

The costs of computing $\operatorname{Rem}\left(A^{-1}, N_{1}\right)$ via lifting and computing the matrix product $A \operatorname{Rem}\left((1 / s) C, N_{2}\right)$ are sensitive to $\alpha$. In particular, both $\log N_{1}$ and $\log N_{2}$ are $O(\log n+$ $\log \kappa(A))$. By Lemma 8, $C_{1}$ can be computed in

$$
\begin{equation*}
O\left(n^{3}(\log n+\log \kappa(A)) \frac{\mathrm{B}(\log n+\log \|A\|)}{\log n+\log \|A\|}\right) \tag{15}
\end{equation*}
$$

bit operations. Now consider the matrix product $A \operatorname{Rem}\left((1 / s) C, N_{2}\right)$. First we convert $\operatorname{Rem}\left((1 / s) C, N_{2}\right)$ to a $p$-adic expansion with $O((\log \kappa) /(\log n+\log \|A\|))$ terms. Multiplying each term by $A$ has the same cost as in (15). Convert back to standard representation. The cost of the radix conversion is bounded by $O\left(n^{2} \mathrm{~B}(n(\log n+\log \|A\|))\right)$ bit operations.

Theorem 16. Let $A \in \mathbb{Z}^{n \times n}$ be nonsingular. Algorithm IntInverse uses $O\left(n^{2}(\log n)(\log n+\right.$ $\log \log \|A\|)$ ) random bits and completes in the cost bound stated in (14). The algorithm either returns Fail or $A^{-1}$. Fail is returned with probability less than $1 / 2$.

Proof. Lemma 14 guarantees that an incorrect result will not be returned. By Lemma 13, the algorithm will not return Fail provided that phase 1 does not return fail and computes $s$ and $\alpha$ correctly, and that phase 2 computes a correct outer product adjoint formula. Summing the probabilities from Lemmas 8 and 12 gives $1 / 12+1 / 6+1 / 4 \leq 1 / 2$. Comparing Lemmas 10, 11 and 13 (the costs of phases 1,2 and 3 ) reveals that (14) from phase 2 dominates.

Corollary 17. The exact inverse of a nonsingular $A \in \mathbb{Z}^{n \times n}$ can be computed in an expected number of $O^{\sim}\left(n^{3}(\log \|A\|+\log \kappa(A))\right)$ bit operations.

## 6 Computing the inverse of a polynomial matrix

Let $A \in \mathrm{~K}[z]^{n \times n}$ be nonsingular with degree $d>0$, K a field. Recall that the degree of the determinant of $A$ is bounded by $n d$, while entries in the adjoint matrix $A^{\text {adj }}:=(\operatorname{det} A) A^{-1}$ are minors of dimension $n-1$ and thus are bounded in degree by $(n-1) d$. Similar to the integer case, the cost of computing an outer product adjoint formula for $A$ will be sensitive to the difference between $\operatorname{deg} A^{\text {adj }}$ and $\operatorname{deg} \operatorname{det} A$. Generically, $\operatorname{deg} A^{\text {adj }}-\operatorname{deg} \operatorname{det} A=(n-1) d-n d<$

PolyInverse $(A)$
Input: Nonsingular $A \in \mathrm{~K}[z]^{n \times n}$ of degree $d>0$, K a field.
Output: $A^{-1}$ or Fail. Fail will be returned with probability $<1 / 2$.
Condition: $\# \mathrm{~K} \geq 10 \mathrm{nd}$.

1. [Initialization]

Let $\Lambda$ be a set of $10 n d$ distinct elements of K .
Choose $\alpha$ and $\beta$ uniformly and randomly from $\Lambda$;
Compute $A_{1}:=\left.A\right|_{z=z+\alpha}$ and let $A_{2}:=z^{d}\left(\left.A_{1}\right|_{z=1 / z}\right)$;
if $\operatorname{det} \operatorname{Rem}\left(A_{1}, z\right)=0$ or $\operatorname{det} \operatorname{Rem}\left(A_{2}, z-\beta\right)=0$ then Fail fi;
$p:=(z-\beta)^{d} ; \quad C:=\operatorname{Rem}\left(A_{2}^{-1}, p\right)$;
Choose $y \in \Lambda^{n \times 1}$ is chosen uniformly and randomly;
$s:=$ the denominator of $A^{-1} y$;
2. [Compute Outer Product Adjoint Formula]

Let $B_{1}$ denote $s A_{2}^{-1}$ and $s_{n+1}=s$.
for $i$ to $n$ do
Choose $y \in \Lambda^{n \times 1}$ uniformly and randomly;
Set $N:=p^{k}$, where $k \in \mathbb{Z}_{>0}$ is minimal s.t. $p^{k} \geq \operatorname{deg} s_{n-i+2}$;
Compute $v:=\operatorname{Rem}\left(B_{i} y, s_{n-i+2}\right)$ using recipe (13);
Compute $x \in \mathcal{R}\left(\mathrm{~K}[x], s_{n-i+2}\right)^{1 \times n}$ such that $\operatorname{gcd}\left(x v, s_{n-i+2}\right)=\operatorname{gcd}\left(v, s_{n-i+2}\right)$;
Compute $u:=\operatorname{Rem}\left(B_{i}^{T} x^{T}, s_{n-i+2}\right)^{T}$ using recipe (13);
$e_{i}:=\operatorname{Rem}\left(x v, s_{n-i+2}\right) ;$ if $e_{i}=0$ then break else $s_{n-i+1}:=s_{n-i+2} / e_{i} \mathbf{f i}$;
if $\left(i=1\right.$ and $\left.s_{n} \neq s\right)$ or $\sum_{j=1}^{i} s_{n-j+1}>n d$ or $e_{i} \nmid u$ then return Fail fi;
$u_{n-i+1}:=u / e_{i} ; \quad v_{n-i+1}:=v / e_{i} ;$
Let $B_{i+1}$ denote $\frac{1}{e_{i}}\left(B_{i}-e_{i} v_{n-i+1} u_{n-i+1}\right) \in \mathrm{K}[x]^{n \times n}$.
od;
if $\sum_{j=1}^{i-1} \neq n d$ then Fail $\mathbf{f}$;
3. [Compute Inverse and Assay Correctness]
if OuterProductMul $\left(s,\left(s_{n-j+1}, v_{n-j+1}, u_{n-j+1}\right)_{1 \leq j \leq i-1}, A_{2}\right) \neq 0_{n \times n}$ then return Fail fi;
$C:=$ OuterProductMul $\left(s,\left(s_{n-j+1}, v_{n-j+1}, u_{n-j+1}\right)_{1 \leq j \leq i-1}, I_{n}\right)$;
if $\operatorname{Rem}\left(A_{2} \operatorname{Rem}((1 / s) C, p), p\right) \neq I_{n}$ then return Fail f;
return $\left.\left(\left(z^{d} / s\right) C\right)\right)\left.\right|_{z=1 /(z-\alpha)}$, with fractions reduced

Figure 6: Algorithm PolyInverse

0 , but in general $\operatorname{deg} A^{\text {adj }}-\operatorname{deg} \operatorname{det} A \leq(n-1) d$, the upper bound being achieved for certain unimodular matrices.

Fortunately, we can apply some simple algebraic preconditioning techniques to transform the original $A$ into another matrix of degree $d$ that has determinant of degree $n d$. Consider the following input matrix over $\mathrm{K}[z]$ where $\mathrm{K}=\mathbb{Z} /\langle 97\rangle$.

$$
A_{1}=\left[\begin{array}{ccc}
72 z^{2}+37 z+74 & 87 z^{2}+44 z+29 & 7 z+56 \\
89 z^{2}+68 z+95 & 11 z^{2}+48 z+50 & 64 z+75 \\
87 z^{2}+31 z+46 & 77 z^{2}+95 z+1 & 63 z+8
\end{array}\right]
$$

The determinant of $A_{1}$ is 17 . But consider the matrix obtained from $A_{1}$ by reverting the polynomials via a change of variables.

$$
A_{2}=z^{2}\left(\left.A_{1}\right|_{z=1 / z}\right)=\left[\begin{array}{ccc}
74 z^{2}+37 z+72 & 29 z^{2}+44 z+87 & 56 z^{2}+7 z \\
95 z^{2}+68 z+89 & 50 z^{2}+48 z+11 & 75 z^{2}+64 z \\
46 z^{2}+31 z+87 & z^{2}+95 z+77 & 8 z^{2}+63 z
\end{array}\right]
$$

The leading coefficient matrix of $A_{2}$ (the coefficient of $z^{2}$ ) is equal to the constant coefficient matrix of $A_{1}$ : since this is nonsingular the degree of the determinant of $A_{2}$ will be equal to $n d$. Indeed, $\operatorname{det} A_{2}=17 z^{6}$. If we start with an $A$ that does not have a nonsingular constant coefficient matrix we apply a shift to produce $A_{1}:=\left.A\right|_{z=z+\alpha}$ for a randomly chosen $\alpha \in \mathrm{K}$. These ideas are summarized in the following lemma, the proof of which is elementary.

Lemma 18. Let $A \in \mathrm{~K}[z]^{n \times n}$ with degree $d$ and let $\alpha \in \mathrm{K}$.

- If $A_{1}:=\left.A\right|_{z=z+\alpha}$ then $A^{-1}=\left.\left(A_{1}^{-1}\right)\right|_{z=z-\alpha}$.
- If $A_{2}:=z^{d}\left(\left.A_{1}\right|_{z=1 / z}\right)$ then $A_{1}^{-1}=\left.\left(z^{d}\left(A_{2}^{-1}\right)\right)\right|_{z=1 / z}$ and $\operatorname{det} A_{2}=z^{n d}\left(\left.\left(\operatorname{det} A_{1}\right)\right|_{z=1 / z}\right)$.
- If $\operatorname{det} \operatorname{Rem}\left(A_{1}, z\right) \neq 0$ then $\operatorname{deg} \operatorname{det} A_{2}=n d$.

The following result is the analogue of Corollary 7.
Lemma 19. Let $s \in \mathrm{~K}[z]$ be nonzero. For any $B \in \mathrm{~K}[z]^{n \times n}$, if $y \in \Lambda^{n \times 1}$ is chosen uniformly and randomly, then $\operatorname{gcd}(B y, s)=\operatorname{gcd}(B, s)$ with probability at least $1-(\operatorname{deg} s) / \# \Lambda$.

Proof. Let $g=\operatorname{gcd}(B, s)$. Then $\operatorname{gcd}(B y, s)=\operatorname{gcd}(B, s)$ if and only if $\operatorname{gcd}((B / g) y, s)=1$. Let $p$ be an irreducible divisor of $s$. Then the residue class ring $\mathrm{K}[z] /\langle p\rangle$ is a field and it is easy to see that $\operatorname{gcd}((B / g) y, p)=1$ with probability at least $1-1 / \# \Lambda$. The result follows by noting that $s$, and hence also $s / g$, has at most $\operatorname{deg} s$ distinct irreducible divisors.

The next result follows from Lemma 19 since the largest invariant factor $s_{n}$ of $A$ has degree bounded by $n d$. In particular, $A^{-1} y$ has denominator $s_{n}$ if and only if $\operatorname{gcd}\left(\left(s_{n} A^{-1}\right) y, s_{n}\right)=1$.

Corollary 20. Let $A \in \mathrm{~K}[z]^{n \times n}$ be nonsingular of degree d. If $y \in \Lambda^{n \times 1}$ is chosen uniformly and randomly, then the denominator of $A^{-1} y$ is equal to the largest invariant factor of $A$ with probability at least $1-n d / \# \Lambda$.

We refer to [16, Section 5] for a derivation of part (a) of the next lemma. The derivation of part (b) is similar to that of part (c) of Lemma 8 . The $n \mathrm{~B}(n d)$ term comes from the application of algorithm OuterProdMul. The $n^{2} k_{n-i+2} \mathrm{~B}(d)$ term captures the cost of performing $O\left(k_{n-i+2}\right)$ lifting steps to obtain $\operatorname{Rem}\left(A^{-1} y, N\right)$.

Lemma 21. Let $A \in \mathrm{~K}[z]^{n \times n}$ be nonsingular of degree $d$. Suppose we have $p \in \mathrm{~K}[z]_{>0}$ with $p \perp \operatorname{det} A$ and $\operatorname{deg} p=d$, together with $C:=\operatorname{Rem}\left(A^{-1}, p\right)$.
(a) If $y \in \mathrm{~K}[z]^{n \times 1}$ satisfies $\operatorname{deg} y \in O(n d)$, then $A^{-1} y$ can be computed with $O\left(n^{3} \mathrm{~B}(d)\right)$ field operations from K .
(b) Consider recipe (13). If $\sum_{j=1}^{i-1} \operatorname{deg} s_{n-i+1} \leq n d$ and $\operatorname{deg} N \in O\left(\operatorname{deg} s_{n-i+2}\right)$, then $\operatorname{Rem}\left(B_{i} y, N\right)$ can be computed in $O\left(n^{2} k_{n-i+2} \mathrm{~B}(d)+n \mathrm{~B}(n d)\right)$ field operations from K , where $k_{n-i+2}:=1+\left(\operatorname{deg} s_{n-i+2}\right) / d$.

## Phase 1

Lemma 22. Phase 1 of algorithm PolyInverse completes in $O\left(n^{3} \mathrm{~B}(n d)\right)$ field operations. Fail is returned with probability at most $1 / 5$. If Fail is not returned, then $\operatorname{deg} \operatorname{det} A_{2}=n d$ and $s$ is the largest invariant factor of $A$ with probability at least $1-1 / 10$.

Proof. Since $\operatorname{deg} \operatorname{det} A \leq n d$, the randomly chosen shift $\alpha$ is a root of $\operatorname{det} A$ with probability at most $n d / \# \Lambda$. Similarly, $\operatorname{det} \operatorname{Rem}\left(A_{2}, t-\beta\right)=0$ with probability at most $n d / \# \Lambda$. Since $\# \Lambda=10 n d$ the claimed bound on the probability of Fail being returned follows.

Now assume that Fail is not returned. That $\operatorname{deg} \operatorname{det} A_{2}=n d$ follows from Lemma 18. Corollary 20 gives the probability estimate for the correctness of $s$. By Lemma 21, the computation of $A^{-1} y$, which dominates the cost of the phase, can be accomplished in the allotted time.

Note that the second probability estimate in Lemma 22 is conditional. In other words, the probability that phase does not return fail and $s$ is computed correctly is at least $1-$ $1 / 5-1 / 10$.

## Phase 2

Lemma 23. Phase 2 of algorithm PolyInverse completes in $O\left(n^{2} \mathrm{~B}(n d)\right)$ field operations.
Proof. Each iteration check that $\sum_{j=1}^{i} \operatorname{deg} s_{n-j+1} \leq n d$, so the sum of the degrees of the moduli $s_{n-i+2}$ over all the loop iterations is bounded by $O(n d)$. Excluding the calls to recipe 13, a total cost bound of $O(n \mathrm{~B}(n d))$ field operations for the entire phase follows from the superlinearity of $B$.

The cost of the two calls to recipe 13 for a single loop iterations is given by part (b) of Lemma 21. Let $k_{n-i+2}$ be as defined in Lemma 21. Then $\sum_{j=1}^{i-1} k_{n-i+2} \in O(n)$. The overall cost of all calls to recipe (13) is thus $O\left(n^{2} \mathrm{~B}(n d)\right)$ field operations.

Lemma 24. If $s$ is the largest invariant factor of $A_{2}$, then phase 2 of algorithm PolyInverse computes an outer product adjoint formula for $A$ with probability at least $1-1 / 5$.
Proof. Phase 2 is identical to algorithm OuterProductAdjoint shown in Figure 3 except that $e_{i}$ is computed probabilistically. By assumption $s=s_{n}$, so during the first iteration the modulus $s_{n+1}$ and $B_{1}=s A_{2}^{-1}$ are correct. Suppose $s_{n+1}, s_{n}, \ldots, s_{n-i+2}$ and hence $B_{1}, B_{2}, \ldots, B_{i}$ are computed correctly up to some $i$. Then by Lemma 19 , the probability that $e_{i}$ is computed incorrectly is bounded by $\left(\operatorname{deg} s_{n-i+2}\right) /(10 n d)$. Since $\sum_{j=1}^{i} \operatorname{deg} s_{n-i+2} \leq$ $\operatorname{deg} s_{n}+\sum_{j=1}^{n} \operatorname{deg} s_{i} \leq 2 n d$, the sum of the failure probability over all loop iterations is bounded by $1 / 5$.

## Phase 3

Lemma 25. If $s$ is the largest invariant factor of $A$, and phase 2 computes a correct outer product adjoint formula for $A$, then phase 3 of algorithm PolyInverse will not return Fail.

Proof. Assume that $s=s_{n}$ and $\left(s_{n-j+1}, v_{n-j+1}, u_{n-j+1}\right)_{1 \leq j \leq i-1}$ is an outer product adjoint formula for $A_{2}$, as specified in the lemma. The first call to OuterProdMul computes $\operatorname{Rem}\left(s A^{-1}, s\right)$ which will be the zero matrix. The second call computes $C=\operatorname{Rem}\left(s A_{2}^{-1}, s\right)$. Since $\operatorname{deg} \operatorname{det} A_{2}=n d$, we have $\operatorname{deg} s A_{2}^{-1}<\operatorname{deg} s$ and by Lemma 1 we conclude that $C=s A_{2}^{-1}$. It follows that the second last line will not return Fail since $A_{2}(1 / s) C=I_{n}$.
Lemma 26. If phase 3 of algorithm PolyInverse does not return Fail, then the correct inverse of $A$ is returned.

Proof. If the first call to OuterProductAdjoint does not return fail, then $C$ as computed in the next line has the property that $\operatorname{Rem}\left(A_{2} C, s\right)$ is the zero matrix. Thus $A_{2} C$ is divisible by $s$. An a priori upper bound for the degree of $(1 / s) A_{2} C$ is $\operatorname{deg} A_{2}-1$, so if the second last line does not return Fail, then $A_{2}$ is indeed the inverse of $A$.

Theorem 27. Let $A \in \mathrm{~K}[x]^{n \times n}$ be nonsingular with degree $d$. Algorithm PolyInverse completes in $O\left(n^{2} \mathrm{~B}(n d)\right)$ field operations. The algorithm either returns Fail or $A^{-1}$. Fail is returned with probability $\leq 1 / 2$. This result assume that $\# \mathrm{~K} \geq 10$ nd.

## 7 Conclusions

The restriction in $\mathrm{K} \geq 10 \mathrm{nd}$ in Theorem 27 is not essential. If K is too small we can work over an algebraic extension of degree $O(\log n d)$ over K. The cost bound of Theorem 27 increases by some factors of $\log n d$.

The main outstanding problem is to remove the dependence of the complexity of the integer matrix inversion algorithm on $\kappa(A)$. A promising approach is to use additive preconditioning as described in [18].

## 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, New York, 1999.
[2] P. Bürgisser, M. Clausen, and M. A. Shokrollahi. Algebraic Complexity Theory, volume 315 of Grundlehren der mathematischen Wissenschaften. Springer-Verlag, 1996.
[3] J. D. Dixon. Exact solution of linear equations using p-adic expansions. Numer. Math., 40:137-141, 1982.
[4] P. D. Domich, R. Kannan, and L. E. Trotter, Jr. Hermite normal form computation using modulo determinant arithmetic. Mathematics of Operations Research, 12(1):5059, 1987.
[5] W. Eberly, M. Giesbrecht, and G. Villard. Computing the determinant and Smith form of an integer matrix. In Proc. 31st Ann. IEEE Symp. Foundations of Computer Science, pages 675-685, 2000.
[6] J. von zur Gathen and J. Gerhard. Modern Computer Algebra. Cambridge University Press, 2 edition, 2003.
[7] M. Giesbrecht. Fast computation of the Smith normal form of an integer matrix. In A. Levelt, editor, Proc. Int'l. Symp. on Symbolic and Algebraic Computation: ISSAC '95, pages 110-118. ACM Press, New York, 1995.
[8] J. L. Hafner and K. S. McCurley. Asymptotically fast triangularization of matrices over rings. SIAM Journal of Computing, 20(6):1068-1083, Dec. 1991.
[9] C. S. Iliopoulos. Worst-case complexity bounds on algorithms for computing the canonical structure of finite abelian groups and the Hermite and Smith normal forms of an integer matrix. SIAM Journal of Computing, 18(4):658-669, 1989.
[10] C. P. Jeannerod and G. Villard. Essentially optimal computation of the inverse of generic polynomial matrices. Journal of Complexity, 21:72-86, 2005.
[11] E. Kaltofen. On computing determinants of matrices without divisions. In P. S. Wang, editor, Proc. Int'l. Symp. on Symbolic and Algebraic Computation: ISSAC '92, pages 342-349. ACM Press, New York, 1992.
[12] E. Kaltofen and G. Villard. On the complexity of computing determinants. Computational Complexity, 13(3-4):91-130, 2004.
[13] I. Kaplansky. Elementary divisors and modules. Trans. of the Amer. Math. Soc., 66:464-491, 1949.
[14] D. E. Knuth. The Art of Computer Programming, Vol.2, Seminumerical Algorithms. Addison-Wesley (Reading MA), 2 edition, 1981.
[15] R. T. Moenck and J. H. Carter. Approximate algorithms to derive exact solutions to systems of linear equations. In Proc. EUROSAM '79, volume 72 of Lecture Notes in Compute Science, pages 65-72, Berlin-Heidelberg-New York, 1979. Springer-Verlag.
[16] T. Mulders and A. Storjohann. Certified dense linear system solving. Journal of Symbolic Computation, 37(4):485-510, 2004.
[17] M. Newman. Integral Matrices. Academic Press, 1972.
[18] V. Pan, R. R. D. Ivolgin, B. Murphy, Y. Tang, and X. Yan. Additive preconditioning for matrix computations. In Third International Computer Science Symposium in Russia, CSR 2008 Moscow, Russia, June 7-12, 2008 Proceedings, LNCS 5010, pages 327-383. Springer Verlag, 2008.
[19] V. Y. Pan and X. Wang. On rational number reconstruction and approximation. SIAM Journal of Computing, 33(2):502-503, 2004.
[20] A. Schönhage. Schnelle Berechnung von Kettenbruchentwicklungen. Acta Informatica, 1:139-144, 1971.
[21] A. Schönhage and V. Strassen. Schnelle Multiplikation grosser Zahlen. Computing, 7:281-292, 1971.
[22] A. Storjohann. A solution to the extended gcd problem with applications. In W. W. Küchlin, editor, Proc. Int'l. Symp. on Symbolic and Algebraic Computation: ISSAC '97, pages 109-116. ACM Press, New York, 1997.
[23] A. Storjohann. Algorithms for Matrix Canonical Forms. PhD thesis, Swiss Federal Institute of Technology, ETH-Zurich, 2000.
[24] A. Storjohann. The shifted number system for fast linear algebra on integer matrices. Journal of Complexity, 21(4):609-650, 2005. Festschrift for the 70th Birthday of Arnold Schönhage.
[25] A. Storjohann and T. Mulders. Fast algorithms for linear algebra modulo N. In G. Bilardi, G. F. Italiano, A. Pietracaprina, and G. Pucci, editors, Algorithms - ESA '98, LNCS 1461, pages 139-150. Springer Verlag, 1998.

