The shifted number system for fast linear algebra on integer matrices

Arne Storjohann [1]

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

Abstract

The shifted number system is presented: a method for detecting and avoiding error producing carries during approximate computations with truncated expansions of rational numbers. Using the shifted number system the high-order lifting and integrality certification techniques of Storjohann 2003 for polynomial matrices are extended to the integer case. Las Vegas reductions to integer matrix multiplication are given for some problems involving integer matrices: the determinant and a solution of a linear system can be computed with about the same number of bit operations as required to multiply together two matrices having the same dimension and size of entries as the input matrix. The algorithms are space efficient.

1 Introduction

From the point of view of computational complexity there is an important analogy between the ring of integers $\mathbb{Z}$ and the ring of univariate polynomials $k[x]$ with coefficients from a field k. The cost of computations over $k[x]$ is usually estimated in terms of the number of field operations from k and thus depends on the degrees of polynomials, while over $\mathbb{Z}$ the measure is the number of bit operations and thus the cost depends on the bitlength of the integers. Many of the core algorithms and techniques in computer algebra have polynomial and integer analogues (for example, multiplication and gcd computation, homomorphic imaging and Chinese remaindering). These and many other examples are addressed in detail in the textbooks [2,15,16].

Email address: astorjoh@uwaterloo.ca (Arne Storjohann).
URL: http://uwaterloo.ca/~astorjoh (Arne Storjohann).

[1] Partially supported by the National Sciences and Engineering Research Council of Canada under Grant No. RGPIN 26082-03: Algorithms for the exact solution to problems in linear algebra.

But the analogy between $\mathbb{Z}$ and $k[x]$ is not complete. Over $k[x]$ we have the possibility of reversion: instead of working with $a = a_0 + a_1x + \cdots + a_kx^k$ we can choose to work with $\mathrm{rev}(a) = a_k + a_{k-1}x + \cdots + a_0x^k$. If b is another polynomial then the leading three terms of ab can be recovered by computing $\mathrm{rev}(a) \times \mathrm{rev}(b)$ modulo x^3. This technique has no analogue over $\mathbb{Z}$. In particular, consider the situation when x is a positive integer radix and a and b are integers written as x-adic expansions. Another difference is that the degree norm over $k[x]$ is non-Archimedean ($\deg(a+b) \leq \max(\deg a, \deg b)$) while the absolute value norm over $\mathbb{Z}$ is Archimedean — the triangle inequality $|a+b| \leq |a|+|b|$ is tight. This property of $|\cdot|$ is well known for causing complications and errors in algorithms due to carry propagation. Even a small additive perturbation can affect the high order coefficients, for example $\underline{665}9999999999999989 + 911 = \underline{666}0000000000000900$.

A method for detecting error producing carries has been proposed in [29]. There, the authors consider the problem of computing the most significant M digits of the product of two multi-precision floating point numbers using a "short product", outlined in [28, Exercise 15 in Section 4.3.1], which can speed the product operation by avoiding the computation of low order terms if these are not required. Unfortunately, the phenomenon of integer carries means the lower order terms might cause an under- or over-flow, leading to an incorrect rounding of the floating point number. The solution proposed in [29] is to compute the leading $M + 2$ digits of the product, then check certain conditions on the trailing coefficients, which we will call guard coefficients. If these conditions are satisfied, the short product is guaranteed to be correct. Under a hypothesis that the guard coefficients are distributed uniformly (which seems reasonable in the context of multi-precision floating point computation) they show that this check on the guard coefficients will rarely fail. In any case, if the check does fail, the full operands may be used to compute the correct result.

In this paper we generalize the technique of using guard coefficients to computations with approximate rational numbers, a number $a + \gamma$ where γ is an integer perturbation. Similar to [29], we establish a sufficient and easily assayable condition on the approximation $a + \gamma$ so that, if this condition holds, we know the leading terms in the X-adic expansion of $a + \gamma$ equal the leading terms in the expansion of a. Unlike the scenario in [29], we don't have recourse to using the full operands if the check on the guard coefficients should fail. We show how to ensure, based on a single random shift choice at the start of the computation, that the required condition on all the guard coefficients will hold throughout a given computation with high probability. For this reason we call our scheme the *shifted number system*.

In [43] we introduced the high-order lifting and integrality certification techniques for polynomial matrices. In this paper we extend these techniques to

the ring $\mathbb{Z}$ and give new complexity bounds for some linear algebra problems on integer matrices, including linear system solving and determinant computation. These algorithms are randomized of the Las Vegas type; the output is certified to be correct but the running time is expected.

The most fundamental problem we consider is linear system solving. Let $A \in \mathbb{Z}^{n\times n}$ be nonsingular and $b \in \mathbb{Z}^{n\times 1}$. The rational system solving problem is to compute $A^{-1}b \in \mathbb{Q}^{n\times 1}$. The bitlength of the numerators and denominators of entries in $A^{-1}b$ will be about n times the bitlength of entries in A. The most effective algorithms for rational system solving are based on p-adic lifting [11,30] and cost $(n^3 \log \|A\|) \times O\tilde{}(\log n + (\log\log \|A\|)^2)$ bit operations [31, Section 5], assuming $\|b\| = \|A\|^{O(n)}$, where $\|A\|$ denotes the maximum entry in absolute value. The radix p should usually be relatively prime to $\det A$ and is typically chosen at random, but if a suitable p is known the algorithm supporting the $O\tilde{}(n^3 \log \|A\|)$ bound is deterministic. In this paper we show how to computes $A^{-1}b$ with $(n^\omega \log \|A\|) \times O\tilde{}((\log n + \log\log \|A\|)^2)$ bit operations (Las Vegas), where $O(n^\omega)$ scalar operations are sufficient to multiply two $n \times n$ matrices.

The complexity of computing the determinant of an integer matrix has been well studied [1,3,7,13,14,23,24,26,27,34]. We refer to [26] for a recent survey. Our focus here is the asymptotic worst-case complexity. The previous best complexity bound in [27] is $O\tilde{}(n^{3.2} \log \|A\|)$ bit operations (Las Vegas) without using sub-cubic matrix multiplication algorithms, and $O\tilde{}(n^{2.697263} \log \|A\|)$ bit operations (Las Vegas) using the fast matrix multiplication algorithms in [8,9]. Our approach here is more similar to [13], who achieve $O\tilde{}(n^{3.5}(\log \|A\|)^{1.5})$ bit operations (Monte Carlo) without the use of fast matrix multiplication techniques. In Theorem 59 we establish that the determinant can be computed with $(n^\omega \log \|A\|) \times O\tilde{}((\log n)^3 + (\log\log \|A\|)^2)$ bit operations, or $O\tilde{}(n^{2.376} \log \|A\|)$ bit operations (Las Vegas) assuming the currently best known value for ω.

We now give an outline of the paper. Section 2 defines our cost model. The results in the rest of the paper can be partitioned along the following lines.

The shifted number system Sections 3 and 4 present the shifted number system for certified computation as discussed above. This part of the paper is self-contained and should be of independent interest.

Low level algorithms for high-order lifting and integrality certification Sections 5, 6, 8, 10 and 11 extend the high-order lifting and integrality certification techniques of [43] to the case of integer matrices. The algorithms in these sections compute parts of the p-adic expansion of A^{-1} or $A^{-1}B$. The algorithms are specified with pre/post conditions and detailed pseudo-code in

terms of low level basic operations such as integer matrix multiplication and integer division with remainder. The algorithms are deterministic, taking as input a shifted number system, and either return the correct result or fail.

Las Vegas reductions to integer matrix multiplication Sections 7, 9 and 13 take a more high level approach, applying the algorithms of the previous sections, up to that point in the paper, to get Las Vegas reductions to matrix multiplication for various linear algebra problems on integer matrices. Section 7 gives an algorithm for unimodularity certification. Section 9 focuses on problems related to linear system solving, and discusses the details of choosing random primes and initializing a suitable shifted number system so that the called-upon low level algorithm has a positive probability of success. Section 13 develops the algorithm for the determinant. Cost estimates are presented in a slightly less precise but more standard and usable form than the low level algorithms.

Augmentative preconditioning of integer matrices Section 12 has a different flavor than the rest of the paper. Here we develop some preconditioners that will be used in the integer determinant algorithm in Section 13. For the polynomial case of the problem [43] we could directly appeal to multiplicative preconditioners from [25] of the form UAV for randomly chosen U and V. Due to the density of integer primes, in particular also small primes, the technique used to prove the multiplicative preconditioners breaks down in the integer case. Instead, here we use augmentative preconditioners: the original matrix A is embedded into a larger matrix, parts of which are chosen randomly. One of the main results in this section is an extension of a result in [13], where the authors prove that an integer matrix chosen uniformly and randomly from $\{0, 1, \ldots, n-1\}^{n \times n}$ will have an expected constant number of nontrivial invariant factors.

2 Basic operations and cost functions

Cost estimates will be given in terms of the following functions. The parameter t is a bitlength, while n is a matrix dimension and X is a bound on the magnitude of integer matrix coefficients.

Cost Function	Operations
$\mathsf{M}(t)$	integer multiplication
$\mathsf{B}(t)$	integer multiplication and gcd-like operations
$\mathsf{MM}(n)$	matrix multiplication over a ring
$\mathsf{MM}(n, X)$	integer matrix multiplication
$\overline{\mathsf{MM}}(n, X)$	reduction to integer matrix multiplication

This section defines these functions and states our assumptions on them.

Let $\mathsf{M} : \mathbb{Z}_{>0} \longrightarrow \mathbb{R}_{>0}$ be such that integers bounded in magnitude by 2^t can be multiplied using at most $\mathsf{M}(t)$ bit operations. The Schönhage–Strassen algorithm [39] allows $\mathsf{M}(t) = O(t(\log t)(\log\log t))$. We assume that $\mathsf{M}(a) + \mathsf{M}(b) \leq \mathsf{M}(a+b)$ and $\mathsf{M}(ab) \leq \mathsf{M}(a)\mathsf{M}(b)$ for $a, b \in \mathbb{Z}_{\geq 2}$. See [15, 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 gcd–related computations. We assume that $\mathsf{B}(t) = \mathsf{M}(t)\log t$ or $\mathsf{B}(t) = t^2$. Then the extended gcd problem with two integers bounded in magnitude by 2^t, and the rational number reconstruction problem [15, Section 5.10] with modulus bounded by 2^t, can be solved with $O(\mathsf{B}(t))$ field operations [38] (compare with [35]).

Let $\mathsf{MM} : \mathbb{Z}_{>0} \longrightarrow \mathbb{R}_{>0}$ be such that two $n \times n$ matrices over a ring (commutative, with 1) can be multiplied using at most $\mathsf{MM}(n)$ ring operations. The classical method has $\mathsf{MM}(n) = 2n^3 - n^2$. Strassen's algorithm [44] allows $\mathsf{MM}(n) = 42n^{\log 7}$. The asymptotically fastest known method allows $\mathsf{MM}(n) = O(n^{2.376})$. We refer to [15, Section 12.1] and [4] for further references and detailed discussion about matrix multiplication.

We now define MM with two arguments. Let $\mathsf{MM} : \mathbb{Z}_{>0} \times \mathbb{Z}_{>0} \longrightarrow \mathbb{R}_{>0}$ be such that

- two matrices from $\mathbb{Z}^{n\times n}$ with entries bounded in magnitude by X can be multiplied using at most $\mathsf{MM}(n, X)$ bit operations, and
- $n^2\,\mathsf{M}(\log X + \log n) \leq \mathsf{MM}(n, X)$.

The second part of the definition will be motivated below. First we consider bounding $\mathsf{MM}(n, X)$ in terms of $\mathsf{MM}(n)$ and M. We need to account for the possible growth in the magnitude of entries in the product matrix. Recall that $\|\cdot\|$ denotes the largest entry in absolute value.

Fact 1 *If $A \in \mathbb{Z}^{*\times n}$ and $B \in \mathbb{Z}^{n\times *}$ then $\|AB\| \leq n\|A\|\|B\|$.*

The use of $*$ for the row dimension of A and column dimension of B in Fact 1 indicates that the result is valid for these dimensions chosen arbitrary. Now, let t be the smallest positive integer such that $2^t > 2n\|A\|\|B\|$. Then we can multiply A and B over the residue class ring $\mathbb{Z}/(2^t)$, elements of $\mathbb{Z}/(2^t)$ represented in the symmetric range. This gives $\mathsf{MM}(n, X) = O(\mathsf{MM}(n)\mathsf{M}(\log X + \log n))$ but better bounds are possible (for example, by employing a homomorphic imaging and Chinese remaindering scheme).

In our algorithms, every time we multiply two integer matrices we will need to perform $O(n^2\,\mathsf{M}(\log X + \log n))$ bit operations additional work (for example, reduce all entries in the product matrix modulo X). For this reason, the definition of $\mathsf{MM}(n, X)$ includes the stipulation that $n^2\,\mathsf{M}(\log X + \log n) = O(\mathsf{MM}(n, X))$. This is a reasonable assumption. On the one hand there exist algorithms supporting the bound $\mathsf{M}(\log X + \log n) = O\tilde{}(\log X + \log n)$. On the other hand the total size of the product matrix is $\Theta(n^2(\log X + \log n))$ bits so $n^2(\log X + \log n) = O(\mathsf{MM}(n, X))$.

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

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

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

Suppose $X \in \mathbb{Z}$ is nonzero. Then $\mathsf{R} := \mathbb{Z}/(X)$ is a principal ideal ring. R can be taken to be the set of all nonnegative integers with magnitude strictly less than X. Multiplication in R costs $O(\mathsf{M}(\log X))$ bit operations and is accomplished by first multiplying over $\mathbb{Z}$ and then reducing modulo X. Similarly, matrices in $\mathsf{R}^{n\times n}$ can be multiplied with $\mathsf{MM}(n, X)$ bit operations. Given an $A \in \mathsf{R}^{n\times n}$, the following can be performed with $O(\overline{\mathsf{MM}}(n, X))$ bit operations:

- Compute a unimodular matrix U such that UA is upper triangular.
- Compute the inverse of A or determine that A is not invertible.
- Compute the Smith canonical form of A.

An algorithm supporting the running time $O(\overline{\mathsf{MM}}(n, X))$ bit operations for the first problem is given in [21]. Now consider the second problem. The matrix A will be invertible precisely if all diagonal entries of UA are invertible. If so, the inverse of UA can be found using an additional $O(\mathsf{MM}(n, X) + n\,\mathsf{B}(\log X + \log n))$ bit operations: first multiply UA by the diagonal matrix D such that diagonal entries in DUA are equal to one, then apply a standard recipe for triangular matrix inversion, see for example [10]. The result for computing the Smith form is given in [41, Chapter 7], see also [40].

We always have $\overline{\mathsf{MM}}(n, X) = O((\log n)\mathsf{MM}(n)\mathsf{B}(\log X + \log n))$. If there exists an absolute constant $\gamma > 0$ such that $n^{2+\gamma} = O(\mathsf{MM}(n))$, then $\overline{\mathsf{MM}}(n, X) = O(\mathsf{MM}(n)\mathsf{B}(\log X + \log n))$. In this paper we don't assume that $n^{2+\gamma} = O(\mathsf{MM}(n))$.

Simplification of cost estimates using assumptions

Some cost estimates will be greatly simplified by explicitly making one of the following assumptions:

- $\mathsf{B}(t) = O(\mathsf{MM}(t)/t)$
- $\mathsf{MM}(a)\mathsf{B}(b) = O(\mathsf{MM}(ab)/b)$

These assumptions are nearly identical; the first is implied by the second with $a = 1$, and the second follows from the first if $\mathsf{MM}(a)\mathsf{MM}(b) = O(\mathsf{MM}(ab))$. These assumptions stipulate that if fast matrix multiplication techniques are used then fast integer multiplication should be used also. For example, the algorithms we present for nonsingular rational system solving requires us to do n gcd-like operations on integers bounded in bitlength by nd. Making the assumption $\mathsf{B}(t) = O(\mathsf{MM}(t)/t)$ allows us to bound $n\,\mathsf{B}(nd)$ by $O(\mathsf{MM}(n)\mathsf{B}(d))$.

3 (X, t)-adic expansions of rational numbers

Let $a \perp b$ denote that two integer a and b are relatively prime. Fix an integer radix $X > 1$, and consider the set of rational numbers

$$\mathcal{S} := \{n/d \mid n, d \in \mathbb{Z},\ d \perp X\}.$$

The set $\mathcal{S}$ is closed under the operations $\{+, -, \times\}$. We are going to define two additional operations Left and Trunc and give some of their properties. First we need to consider the X-adic expansion of elements of $\mathcal{S}$. Fix an integer shift t, $0 \le t < X - 1$. Then every $a \in \mathcal{S}$ has a unique and possibly infinite expansion

$$a = a_0 + a_1 X + a_2 X^2 + a_3 X^3 + \cdots, \tag{2}$$

where each integer coefficient a_i is chosen in the range $[-t, X-1-t]$ so that the partial sum $a_0 + a_1 X + \cdots + a_i X^i$ is congruent to a modulo X^{i+1}. We call (2) the (X, t)-adic expansion of a. The (X, t)-adic expansion gives an embedding of elements of $\mathcal{S}$ into the ring of (X, t)-adic expansions. The functions Left and Trunc will be parameterized in terms of a proscribed X and t. By default, Trunc := Trunc$[(X, t)]$ and Left := Left$[(X, t)]$.

Let $a \in \mathcal{S}$ have (X, t)-adic expansion as in (2), and let k be a nonnegative

integer. Then Trunc is defined as follows:

$$\mathrm{Trunc}(a, k) := a_0 + a_1 X + a_2 X^2 + \cdots + a_{k-1} X^{k-1}.$$

The Trunc operation truncates an X-adic expansion:

$$\begin{aligned} a &= a_0 + a_1 X + a_2 X^2 + a_3 X^3 + a_4 X^4 + a_5 X^5 + a_6 X^6 + \cdots \\ \mathrm{Trunc}(a, 4) &= a_0 + a_1 X + a_2 X^2 + a_3 X^3. \end{aligned}$$

Every finite, and thus every truncated (X, t)-adic expansion is an integer.

The Left operation is defined as follows.

$$\mathrm{Left}(a, k) := a_k + a_{k+1} X + a_{k+2} X^2 + \cdots .$$

The Left operation corresponds to division by a power of X. The name for this operation comes from the fact that all coefficients of the (X, t)-adic expansion are shifted left:

$$\begin{aligned} a &= a_0 + a_1 X + a_2 X^2 + a_3 X^3 + a_4 X^4 + a_5 X^5 + a_6 X^6 + \cdots \\ \mathrm{Left}(a, 3) &= a_3 + a_4 X + a_5 X^2 + a_6 X^3 + a_7 X^4 + a_8 X^5 + a_9 X^6 + \cdots . \end{aligned}$$

The following lemmas follow from the definition of Left and Trunc.

Lemma 2 $a = \mathrm{Trunc}(a, k) + \mathrm{Left}(a, k) X^k$.

Lemma 3 *If* $l \leq k$ *then* $\mathrm{Left}(\mathrm{Trunc}(a, k), l) = \mathrm{Trunc}(\mathrm{Left}(a, l), k - l)$.

All the above definitions extend naturally to the case of matrices by element-wise application. Just replace the scalar a with a matrix A.

The (X, t)-adic shifted number system

Negative integers have infinite $(X, 0)$-adic expansions. For this reason we want to disallow the shift choice $t = 0$. Moreover, for reasons that will become clear in the next section where we discuss the problem of carry propagation, we want the endpoints of the coefficient range $[-t, X - 1 - t]$ to be a distance of at least two away from zero. This is achieved by stipulating that $1 < t < X - 2$, which implies that $X > 4$.

Recall that $\mathcal{S} := \{n/d \mid n, d \in \mathbb{Z}, d \perp X\}$.

Definition 4 *For* $1 < t < X-2$ *the* (X,t)-adic shifted number system *is the set of rational numbers* $\mathcal{S}$ *together with the operations* $\{+,-,\times,\text{Left},\text{Trunc}\}$ *as defined above.*

In a shifted number system there is a natural isomorphism between $\mathbb{Z}$ and the subset of $\mathcal{S}$ comprised of all elements having a finite (X,t)-adic expansion. We have

$$\sum_{i=0}^{k-1}(-t)X^i \le \text{Trunc}(a,k) \le \sum_{i=0}^{k-1}(X-1-t)X^i.$$

Converting the sums to closed form gives the following.

Lemma 5 *In an* (X,t)*-adic shifted number system,* $\text{Trunc}(a,k) = a$ *and* $\text{Left}(a,k) = 0$ *if and only if* $a \in \mathbb{Z}$ *and*

$$\frac{-t\left(X^k-1\right)}{X-1} \le a \le \frac{(X-1-t)\left(X^k-1\right)}{X-1}.$$

As expected, the size of the range is X^k. It will be useful to have a version of Lemma 5 that does not depend on t. The stipulation that $1 < t < X-2$ gives the following.

Corollary 6 *If* $a \in \mathbb{Z}$ *and* $|a| \le 2(X^k-1)/(X-1)$ *then* $\text{Trunc}(a,k) = a$.

Lemma 5 and Corollary 6 extend naturally to the case of matrices. Just replace the scalar a with a matrix A. The condition on A is that each entry is an integer in the specified range. The next result gives an additional extension to matrices.

Corollary 7 *If* $A \in \mathbb{Z}^{*\times n}$ *and* $B \in \mathbb{Z}^{n\times *}$ *then* $\text{Left}(A\,\text{Trunc}(B,k),k) \le n\|A\|$.

PROOF. Corollary 6 was obtained by minimizing the range for a in Lemma 5. Maximizing the range gives the upper bound $|\text{Trunc}(a,k)| \le (X-3)(X^k-1)/(X-1)$, which is strictly less than X^k. Thus $\|\text{Trunc}(B,k)\| < X^k$. Let c be any entry of $A\,\text{Trunc}(B,k)$. Then Fact 1 gives $|c| < n\|A\|X^k$. Recall that $c = \text{Trunc}(c,k) + \text{Left}(c,k)X^k$. Using $|\text{Left}(c,k)|X^k - |\text{Trunc}(c,k)| \le |c|$ and $|\text{Trunc}(c,k)| < X^k$ gives $\text{Left}(c,k) < n\|A\| + |\text{Trunc}(c,k)|/X^k < n\|A\|+1$. □

The (X,t,s)*-adic shifted number system*

In the next section we demonstrate the problem of error producing carries and develop a technique, based on using "guard" coefficients, to detect and avoid

such carries. Using an entire coefficient in the (X,t)-adic expansion for this purpose will be wasteful. This motivates the following definition.

Let (X,t) and $(\bar{X},\bar{t})$ be shifted number systems such that $(X,t) = (\bar{X}^s, \bar{t}(1 + \bar{X} + \cdots + \bar{X}^{s-1}))$, for some integer $s \geq 2$. We call this an (X,t,s)-adic shifted number system. Note that the (X,t)-adic and $(\bar{X},\bar{t})$-adic expansions of a are related as follows:

$$a = \overbrace{\bar{a}_0 + \bar{a}_1\bar{X} + \cdots + \bar{a}_{s-1}\bar{X}^{s-1}}^{a_0} + \overbrace{\bar{a}_s\bar{X}^s + \bar{a}_{s+1}\bar{X}^{s+1} + \cdots + \bar{a}_{2s-1}\bar{X}^{2s-1}}^{a_1 X} + \cdots .$$

Then $\mathrm{Trunc}(a,k) = \mathrm{Trunc}[(\bar{X},\bar{t})](a,sk)$ and $\mathrm{Left}(a,k) = \mathrm{Left}[(\bar{X},\bar{t})](a,sk)$. (Recall that $\mathrm{Trunc} := \mathrm{Trunc}[(X,t)]$ and $\mathrm{Left} := \mathrm{Left}[(X,t)]$ by default.) Corollary 6 then states that $\mathrm{Trunc}[(\bar{X},\bar{t})](a,sk) = a$ if $a \in \mathbb{Z}$ and $|a| \leq 2(X^k-1)/(\bar{X}-1)$. Noting that $X^k/\bar{X} \leq (X^k-1)/(\bar{X}-1)$ if $k \geq 1$ gives the following simplified version.

Corollary 8 *If $a \in \mathbb{Z}$ and $|a| \leq 2X^k/\bar{X}$ then $\mathrm{Trunc}(a,k) = a$.*

Most of the computations in our algorithms will be performed in the (X,t)-adic system, but at those points where we need to check guard coefficients we work in the $(\bar{X},\bar{t})$-adic system. Then, instead of using a_1 as a guard coefficient, for example, it will suffice to use $\bar{a}_{2s-1}$.

Computing in a shifted number system

Considering (X,t)-adic expansions of elements of $\mathcal{S}$ allowed for natural definitions of Left and Trunc, but to compute in an (X,t)-adic number system we don't necessarily need to work with (X,t)-adic expansions. Let $\mathrm{mod}(a,b)$ denote the unique integer $r \in [0, b-1]$ that is congruent to a modulo b. Then the Trunc and Left operations can be implemented as follows.

$$\texttt{Trunc}[(X,t)](a,k) := \left\lceil \begin{array}{l} r := \mathrm{mod}(a, X^k); \\ \textbf{if } r > (X-1-t)(X^k-1)/(X-1) \textbf{ then} \\ \quad r := r - X^k \\ \textbf{fi}; \\ \textbf{return } r \end{array} \right.$$

$$\texttt{Left}[(X,t)](a,k) := \left\lceil \begin{array}{l} \#\ \text{Let } \texttt{Trunc} := \texttt{Trunc}[(X,t)]. \\ \textbf{return } (a - \texttt{Trunc}(a,k))/X^k \end{array} \right.$$

4 Certified computation in shifted number systems

The algorithms in subsequent sections work in an (X,t)-adic shifted number system and use approximate arithmetic to compute exact results. Let $a \in \{n/d \mid n,d \in \mathbb{Z}, d \perp X\}$, γ be an integer, and k be a positive integer. Suppose a represents an exact quantity, while γ is a small perturbation: $|\gamma| \leq X^{k-1}$ in which case $\mathrm{Left}(\gamma,k) = 0$. The scenario is that we have computed an approximation $a+\gamma$ of a, and hope to recover from this the high-order coefficients $\mathrm{Left}(a,k)$ of a.

$$\begin{aligned} a &= a_0 + a_1 X + \cdots + a_{k-1}X^{k-1} + \mathrm{Left}(a,k)X^k \\ \gamma &= \gamma_0 + \gamma_1 X + \cdots + \gamma_{k-1}X^{k-1} \qquad (|\gamma| \leq X^{k-1}) \\ a+\gamma &= b_0 + b_1 X + \cdots + b_{k-1}X^{k-1} + \mathrm{Left}(a+\gamma,k)X^k \end{aligned}$$

The obvious approach is to compute $\mathrm{Left}(a+\gamma,k)$ and hope that this equals $\mathrm{Left}(a,k)$. Unfortunately, the phenomenon of carry propagation means that this may not be the case. For example, if $(X,t) = (10,5)$ then $\mathrm{Trunc}(4,1) = 4$ and $\mathrm{Trunc}(1,1) = 1$, but $\mathrm{Left}(4+1,1) \neq \mathrm{Left}(4,1) + \mathrm{Left}(1,1)$. From an algorithm design point of view there are two questions we must address.

(1) How can we assay if the high order coefficients of $a+\gamma$ have been corrupted by a carry propagation?
(2) How can we ensure that error producing carry propagations will be rare?

This section gives answers.

Let the (X,t)-adic expansion of a, γ and $a+\gamma$ be as above. Consider adding together $\mathrm{Trunc}(a,k)$ and γ:

$$\begin{aligned} \mathrm{Trunc}(a,k) &= a_0 + a_1 X + \cdots + a_{k-1}X^{k-1} \\ \gamma &= \gamma_0 + \gamma_1 X + \cdots + \gamma_{k-1}X^{k-1} \qquad (|\gamma| \leq X^{k-1}) \end{aligned}$$

The following lemma gives a sufficient condition on a_{k-1} that ensures the expansion of $\mathrm{Trunc}(a,k)+\gamma$ looks like:

$$\mathrm{Trunc}(a,k) + \gamma = b_0 + b_1 X + \cdots + b_{k-1}X^{k-1}.$$

In other words, there will be enough "room" in the first k coefficients of the expansion of a to "absorb" the perturbation γ, without causing an under- or over-flow to the coefficient of X^k in the expansion of $\mathrm{Trunc}(a,k)+\gamma$.

Lemma 9 *If* $a_{k-1} \notin \{-t, X-1-t\}$ *then* $\mathrm{Left}(a+\gamma,k) = \mathrm{Left}(a,k)$.

PROOF. The bounds $|\gamma| \leq X^{k-1}$ and $-t < a_{k-1} < X-1-t$ give upper and lower bounds for $a - \text{Left}(a,k)X^k + \gamma$ that allow application of Lemma 5:

$$\overbrace{\text{Trunc}(a - \text{Left}(a,k)X^k + \gamma, k)}^{= \text{Trunc}(a+\gamma,k)} = a - \text{Left}(a,k)X^k + \gamma$$

By definition, $\text{Left}(a+\gamma,k) = (a + \gamma - \text{Trunc}(a+\gamma,k))/X^k$. Now substitute $\text{Trunc}(a+\gamma,k) = a - \text{Left}(a,k)X^k + \gamma$ to get the result. □

By symmetry, we get the following corollary.

Lemma 10 *If* $b_{k-1} \notin \{-t, X-1-t\}$ *then* $\text{Left}(a+\gamma,k) = \text{Left}(a,k)$.

Let us remark on the crucial difference between Lemmas 9 and 10 in the context of the scenario we have sketched above. On the one hand, we can't check that $a_{k-1} \notin \{-t, X-1-t\}$ because we don't know a. On the other hand, we can check that $b_{k-1} \notin \{-t, X-1-t\}$ because b_{k-1} is a coefficient of the computed approximation $a+\gamma$. Thus, Lemma 10 gives an answer to our first question.

Now we turn our attention to the second question. The next theorem gives a sufficient condition on a_{k-1} for the test suggested by Lemma 10 not to fail.

Theorem 11 *If* $a_{k-1} \notin \{-t, -t+1, X-2-t, X-1-t\}$ *then* $b_{k-1} \notin \{-t, X-1-t\}$.

PROOF. The bounds $|\gamma| \leq X^{k-1}$ and $-t+1 < a_{k-1} < X-2-t$ give upper and lower bounds for $\text{Trunc}(a,k) + \gamma$ that imply $-t < b_{k-1} < X-1-t$. □

The key idea of this section is that the condition of Theorem 11 can be made to hold with high probability (if X is large enough) by making a single random choice for the shift t at the start of the computation. Although X, a and k are fixed, the perturbation γ as well as coefficients a_{k-1} and b_{k-1} in the (X,t)-adic expansions of a and $a+\gamma$ will depend on the choice of t. We want to identify possible bad choices of t: choices for which the condition of Theorem 11 on a_{k-1} does not hold. Let $c = \text{mod}((a - \text{mod}(a, X^{k-1}))/X^{k-1}, X)$, where $\text{mod}(a,b)$ denotes the unique $r \in [0, b-1]$ that is is congruent to a modulo b. Then c is invariant of the choice of the shift t. The next lemma follows from the definition of Left and Trunc.

Lemma 12 $a_{k-1} \in \{c, c-X, c+1, c+1-X\}$.

Lemma 12 observes that over all possible values of t, a particular coefficient of the (X,t)-adic expansion can take on at most four different values. Considering Lemma 12, the condition of Theorem 11 on a_{k-1} will be satisfied if the following set is empty:

$$\{c, c-X, c+1, c+1-X\} \cap \{X-2-t, X-1-t, -t, -t+1\}$$

A sufficient condition for the intersection to be empty is that no element of $\{c, c-X, c+1, c+1-X\}$ be congruent modulo X with an element of $\{X-2-t, X-1-t, -t, -t+1\}$. This gives the following result.

Theorem 13 *If* $t \notin \{\mathrm{mod}(-c-3,X),\ \mathrm{mod}(-c-2,X),\ \mathrm{mod}(-c-1,X),\ \mathrm{mod}(-c,X),\ \mathrm{mod}(-c+1,X)\}$ *then* $b_{k-1} \notin \{-t, X-1-1\}$.

Our algorithm for the certified Left operation works in an (X,t,s)-adic shifted number system: $(X,t) = (\bar{X}^s, \bar{t}(1+\bar{X}+\cdots+\bar{X}^{s-1}))$. Let the $(\bar{X},\bar{t})$-adic expansions of a, γ and $a+\gamma$ be

$$\begin{aligned} a &= \bar{a}_0 + \bar{a}_1\bar{X} + \cdots + \bar{a}_{sk-1}\bar{X}^{sk-1} + \mathrm{Left}[(\bar{X},\bar{t})](a, sk)\bar{X}^{sk} \\ \gamma &= \bar{\gamma}_0 + \bar{\gamma}_1\bar{X} + \cdots + \bar{\gamma}_{sk-1}\bar{X}^{sk-1} \\ a+\gamma &= \bar{b}_0 + \bar{b}_1\bar{X} + \cdots + \bar{b}_{sk-1}\bar{X}^{sk-1} + \mathrm{Left}[(\bar{X},\bar{t})](a+\gamma, sk)\bar{X}^{sk} \end{aligned}$$

Note that $\mathrm{Left}(a+\gamma,k) = \mathrm{Left}(a,k)$ if and only if $\mathrm{Left}[(\bar{X},\bar{t}])(a+\gamma,sk) = \mathrm{Left}[(\bar{X},\bar{t})](a,sk)$. Working in the (X,t,s)-adic system has the advantage that we can relax the condition on the magnitude of γ. While before we had $|\gamma| \leq X^{k-1}$, here it suffices that $\gamma \leq \bar{X}^{sk-1} = X^k/\bar{X}$. While before we checked coefficient b_{k-1} of the (X,t)-adic expansion of $a+\gamma$, here we check coefficient $\bar{b}_{sk-1}$ of the $(\bar{X},\bar{t})$-adic expansion:

$$b_{k-1}X^{k-1} = \bar{b}_{s(k-1)}\bar{X}^{s(k-1)} + \bar{b}_{s(k-1)+1}\bar{X}^{s(k-1)+1} + \cdots + \bar{b}_{sk-1}\bar{X}^{sk-1}.$$

We obtain the following subroutine.

Subroutine 14 `CertLeft`$[(X,t,s)](a+\gamma,k)$
Remark: $(X,t) = (\bar{X}^s, \bar{t}(1+\bar{X}+\cdots+\bar{X}^{s-1}))$
Input: $a+\gamma \in \mathbb{Z}$ such that $\gamma \in \mathbb{Z}$ with $|\gamma| \leq X^k/\bar{X}$, $k > 0$.
Output: Failure, or $\mathrm{Left}(a,k)$

```
b̄_{sk-1} := Left[(X̄, t̄)](Left(Trunc(a + γ, k), k − 1), s − 1);
if b̄_{sk-1} ∈ {−t̄, X̄ − 1 − t̄} then
    return fail
else
    return Left(a + γ, k)
fi;
```

The next result follows as a corollary of Lemma 10 and Theorem 13.

Theorem 15 *Subroutine 14* (`CertLeft`) *is correct. Fix X, s and k and suppose that $[(X, *, s)](a + \gamma, k)$ is a valid input to the algorithm. If $\bmod((a - \bmod(a, X^{k-1}))/X^{k-1}, X)$ is also fixed (independent of the choice of t) then there are at most five choices for $\bar{t} \in [2, \bar{X}-3]$ for which* `CertLeft`$[(X, t, s)](a+\gamma, k)$ *will return fail, $|\gamma| \leq X^k/\bar{X}$.*

Subroutine 14 extends naturally to handle matrix arguments. Just replace a and γ with matrices $A \in \mathbb{Z}^{n\times m}$ and $\gamma \in \mathbb{Z}^{n\times m}$. The condition on $\bar{b}_{sk-1}$ must hold for all nm entries of $\bar{B}_{sk-1}$.

5 (X, t)-adic lifting using short products

Let $A \in \mathbb{Z}^{n\times n}$ be nonsingular, $X \perp \det A$. Let $B \in \mathbb{Z}^{n\times m}$. Lifting can be used to compute a truncated (X, t)-adic expansion of $A^{-1}B$. The next definition and lemma give the key idea. Note that the division by X^k is exact, $k \geq 0$.

Definition 16 $\mathrm{Res}(A, B, k) := (B - A\,\mathrm{Trunc}(A^{-1}B, k))/X^k$.

Lemma 17 $A^{-1}B = \mathrm{Trunc}(A^{-1}B, k) + A^{-1}\,\mathrm{Res}(A, B, k)X^k$.

Lemma 17 may be understood as follows. The problem of computing $A^{-1}B$ up to a certain order can be divided into two parts. The first is to compute $\mathrm{Trunc}(A^{-1}B, k)$. The second is to continue by computing the expansion of $A^{-1}\mathrm{Res}(A, B, k)$. A fact we will use later is that $\|\mathrm{Res}(A, B, k)\|$ may be small even if $\|B\|$ is large. The next lemma is used to prove this claim.

Lemma 18 $\mathrm{Res}(A, B, k) = \mathrm{Left}(-A\,\mathrm{Trunc}(A^{-1}B, k), k)$ *if and only* $B = \mathrm{Trunc}(B, k)$.

PROOF. Let $B = L_1 + H_1X^k$ where $L_1 = \mathrm{Trunc}(B, k)$ and $H_1 = \mathrm{Left}(B, k)$. Similarly, let $A\,\mathrm{Trunc}(A^{-1}B, k) = L_2 + H_2X^k$. Then $\mathrm{Res}(A, B, k) = (L_1 - L_2)/X^k + H_1 - H_2$ where $L_1 \equiv L_2 \bmod X^k$. We must have $L_1 = L_2$ since $L_1 = \mathrm{Trunc}(L_1, k)$, $L_2 = \mathrm{Trunc}(L_2, k)$. The result follows. □

Corollary 7 gives the bound $\|\mathrm{Left}(-A\,\mathrm{Trunc}(A^{-1}B, k), k)\| \leq n\|A\|$. At this point we assume we are working over an (X, t, s)-adic number system. Recall that $(X, t) = (\bar{X}^s, \bar{t}(1 + \bar{X} + \cdots + \bar{X}^{s-1}))$. Then a sufficient condition that $B = \mathrm{Trunc}(B, k)$ is furnished by Corollary 8: $\|B\| \leq 2X^k/\bar{X}$. The next result now follows from Lemma 18.

Corollary 19 *If k satisfies $\|B\| \leq 2X^k/\bar{X}$ then $\|\mathrm{Res}(A, B, k)\| \leq n\|A\|$.*

We now develop the two key subroutines in this paper.

The short product lift: `SPL`

Let $k \geq 2$, and consider the (X,t)-adic expansion

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

Suppose we want to compute only the single high-order coefficient

$$D := \mathrm{Left}(\mathrm{Trunc}(A^{-1}B,k),k-1)$$

together with $M := \mathrm{Left}(\mathrm{Trunc}(A^{-1},k)B,k)$, as shown in (4).

$$\mathrm{Trunc}(A^{-1},k)B = \overbrace{\mathrm{Trunc}(A^{-1}B,k-1)+DX^{k-1}}^{\mathrm{Trunc}(A^{-1}B,k)} + MX^k. \tag{4}$$

In general, we need all coefficients of $\mathrm{Trunc}(A^{-1},k)$ to compute D and M. The next result shows that it may suffice to have only E in case $\|B\|$ is small enough.

Theorem 20 *If* $n\|B\| \leq X/\bar{X}$*, and* `CertLeft`$(EB,1)$ *does not return fail, then* $\mathrm{Left}(EB,1) = D + MX$*.*

PROOF. $\mathrm{Trunc}(A^{-1},k) = C + EX^{k-2}$. This gives $D + MX = \mathrm{Left}(CB + EBX^{k-2}, k-1)$. Assume that $n\|B\| \leq X/\bar{X}$. Then $\|CB\| \leq n\|C\|\|B\| < nX^{k-2}\|B\| \leq X^{k-1}/\bar{X}$, so the conditions of Subroutine 14 (`CertLeft`) are satisfied, that is, if `CertLeft` does not return fail, then

$$\mathrm{Left}(\overbrace{EBX^{k-2}+CB}^{a} + \overbrace{(-CB)}^{\gamma}, k-1) = \mathrm{Left}(\overbrace{EBX^{k-2}+CB}^{a}, k-1). \tag{5}$$

The result follows. □

Based on Theorem 20 we get the following subroutine for computing the tuple (D,M). The subroutine `MUL`$(*,*)$ computes the product of two integer matrices.

Subroutine 21 `SPL`$[(X,t,s)](E,B)$
Remark: $(X,t) = (\bar{X}^s, \bar{t}(1+\bar{X}+\cdots+\bar{X}^{s-1}))$
Input: $E = \mathrm{Left}(\mathrm{Trunc}(A^{-1},k),k-2)$ for some $A \in \mathbb{Z}^{n\times n}$ and $k \geq 2$, $B \in$

$\mathbb{Z}^{n\times m}$
Output: Failure, or $(\mathrm{Left}(\mathrm{Trunc}(A^{-1}B,k),k-1),\mathrm{Left}(\mathrm{Trunc}(A^{-1},k)B,k))$.
Condition: $n\|B\| \le X/\bar{X}$.
$S := \mathtt{CertLeft}(\mathtt{MUL}(E,B),1)$;
if $S =$ **fail then**
  return fail
else
  return $(\mathrm{Trunc}(S,1),\mathrm{Left}(S,1))$
fi

The call to `CertLeft` in Subroutine 21 (`SPL`) is the only point where any of the algorithms developed in the subsequent sections check guard coefficients. In order to be able to apply Theorem 15 to bound the number of bad choices for $\bar{t} \in [2, \bar{X}-3]$ (that is, choices for which $\mathtt{SPL}[(X,t,s)](E,B)$ will return fail) we need to be able to relate EB, the argument to `CertLeft`, to a fixed quantity (independent of t) plus an additive perturbation (which may depend on t). We now explain how this will be done.

The algorithms in subsequent sections also work in an (X,t,s)-adic number system, and a particular input tuple (E,B) to `SPL`, the result of an intermediate computation, will depend on the choice of t. Thus, the a shown in (5) is not fixed with respect to the choice of t and is unsuitable for our current purpose. However, in all invocations of `SPL` made by the algorithms in subsequent sections, we have $E = \mathrm{Left}(\mathrm{Trunc}(A^{-1},k),k-2)$ and $B = \mathrm{Res}(A,R,j)$ for a fixed $(A,R,k,j) \in (\mathbb{Z}^{n\times n},\mathbb{Z}^{n\times m},\mathbb{Z}_{\ge 2},\mathbb{Z}_{\ge 0})$. The key observation now, using Lemma 17, is that

$$A^{-1}R \equiv \mathrm{Trunc}(A^{-1}R,j) + (CB + EBX^{k-2})X^j \pmod{X^{j+k-1}}$$

Let $a = A^{-1}R$ and $\gamma = -(\mathrm{Trunc}(A^{-1}R,j)+CBX^j)$. Then $EBX^{j+k-2} \equiv a+\gamma \pmod{X^{j+k-1}}$ and $\mathtt{CertLeft}(EB,1)$ will fail if and only if $\mathtt{CertLeft}(a+\gamma, j+k-1)$ fails. Finally, note that

$$\begin{aligned}\|\gamma\| &\le \|\mathrm{Trunc}(A^{-1}R,j)\| + \|CB\|X^j \\ &\le (X^j-1) + n(X^{k-2}-1)\|B\|X^j \\ &< X^j + (X^{k-2}-1)X^{j+1}/\bar{X} \\ &= X^j - X^{j+1}/\bar{X} + X^{j+k-1}/\bar{X} \\ &< X^{j+k-1}/\bar{X}.\end{aligned}$$

so that the conditions of Theorem 15 are all satisfied. We obtain the following result.

Lemma 22 *Let $[(X,*,s)](E,B)$ be a valid input tuple to Subroutine 21* (`SPL`). *If $E = \mathrm{Left}(\mathrm{Trunc}(A^{-1},k),k-2)$ and $B = \mathrm{Res}(A,R,j)$ for a fixed $(A,R,k,j) \in$*

$(\mathbb{Z}^{n\times n}, \mathbb{Z}^{n\times m}, \mathbb{Z}_{\geq 2}, \mathbb{Z}_{\geq 0})$, *independent of* t, *then there are most* $5n^2$ *choices for* $\bar{t} \in [2, \bar{X} - 3]$ *for which* $\texttt{SPL}[(X, t, s)](E, B)$ *will return fail.*

The short product residue: SPR

Now we consider the computation of $\mathrm{Res}(A, B, k)$, shown in (6).

$$A^{-1}B = \overbrace{* + *X + \cdots + *X^{k-2} + DX^{k-1}}^{\mathrm{Trunc}(A^{-1}B,\,k)} + A^{-1}\mathrm{Res}(A, B, k)X^k. \qquad (6)$$

In general, we need all coefficients of $\mathrm{Trunc}(A^{-1}B, k)$ to compute $\mathrm{Res}(A, B, k)$. In fact, it suffices to have only D in case $\|A\|$ and $\|B\|$ are small enough.

Lemma 23 $\mathrm{Res}(A, B, k) = \mathrm{Left}(-AD, 1)$ *if and only if* $\mathrm{Left}(B - A\,\mathrm{Trunc}(A^{-1}B, k-1), k) = 0$.

PROOF. By definition, $\mathrm{Res}(A, B, k) = (B - A\,\mathrm{Trunc}(A^{-1}B, k))/X^k$. Using

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

gives

$$-ADX^{k-1} = \mathrm{Res}(A, B, k)X^k - (B - A\,\mathrm{Trunc}(A^{-1}B, k-1).$$

The result follows. □

An *a priori* bound is $\|B - A\,\mathrm{Trunc}(A^{-1}B, k-1)\| \leq \|B\| + n\|A\|X^{k-1}$. The next theorem follows as a corollary of Corollary 8 and Lemma 23.

Theorem 24 *If* $\|B\| + n\|A\|X^{k-1} \leq 2X^k/\bar{X}$, *then* $\mathrm{Res}(A, B, k) = \mathrm{Left}(-AD, 1)$.

Instead of the condition of Theorem 24, we use in the following subroutine the slightly more restrictive, but simpler condition: $\|B\| \leq X^k/\bar{X}$ and $n\|A\| \leq X/\bar{X}$.

Subroutine 25 $\texttt{SPR}[(X, t, s)](A, D)$
Remark: $(X, t) = (\bar{X}^s, \bar{t}(1 + \bar{X} + \cdots + \bar{X}^{s-1}))$
Input: $A \in \mathbb{Z}^{n\times n}$, $D = \mathrm{Left}(\mathrm{Trunc}(A^{-1}B, k), k-1)$ for some $k \geq 1$, $B \in \mathbb{Z}^{n\times m}$.
Output: $\mathrm{Res}(A, B, k)$.
Condition: $\|B\| \leq X^k/\bar{X}$ and $n\|A\| \leq X/\bar{X}$.
return $\mathrm{Left}(-\texttt{MUL}(A, D), 1)$

6 High-order segment of inverse

We are working over an (X,t)-adic shifted number system. Let $A \in \mathbb{Z}^{n\times n}$ be nonsingular, and suppose $X \perp \det A$. We present an algorithm to compute a high-order segment $E = \mathrm{Left}(\mathrm{Trunc}(A^{-1},k),k-2)$, $k = 2^l$ for a given $l \geq 1$.

Algorithm 26 `Segment`$[(X,t,s)](A,l)$
Remark: $(X,t) = (\bar{X}^s, \bar{t}(1+\bar{X}+\cdots+\bar{X}^{s-1}))$
Input: $A \in \mathbb{Z}^{n\times n}$, $l \geq 1$.
Output: Failure, or $\mathrm{Left}(\mathrm{Trunc}(A^{-1},k),k-2)$ for $k = 2^l$.
Condition: $X \perp \det A$ and $n^2\|A\| \leq X/\bar{X}$.

(1) $L := \mathrm{Trunc}(A^{-1},1)$
$H := \mathrm{Trunc}(\mathtt{MUL}(L,\mathrm{Left}(I-\mathtt{MUL}(A,L),1)),1)$;
$E := L + HX$;
(2) # If any call to `SPL` fails then exit early with fail.
for i **from** 1 **to** $l-1$ **do**
$(L,*) := \mathtt{SPL}(E,\mathtt{SPR}(A,L))$;
$(H,*) := \mathtt{SPL}(E,\mathtt{SPR}(A,H))$;
$E := L + HX$
od;
return E

We now prove correctness. Phase 1 computes $E = \mathrm{Trunc}(A^{-1},2)$. Now consider phase 2. Assume that none of the calls to `SPL` fail. Provided we show that all the preconditions hold for each call to Subroutines 21 (`SPL`) and 25 (`SPR`), then induction on i together with the specification of the subroutines shows that at the start of each iteration of the loop, E is equal to the following segment of coefficients in the truncated (X,t)-adic expansion of A^{-1}:

$$\mathrm{Trunc}(A^{-1},2^i) = *+*X+\cdots+*X^{2^i-3}+\overbrace{LX^{2^i-2}+HX^{2^i-1}}^{EX^{2^i-2}}. \qquad (7)$$

The result will follow.

The first line of the loop body computes $\mathtt{SPR}(A,L) = \mathrm{Res}(A,I,2^i-1)$ and then the new L. Similarly, the next line computes $\mathtt{SPR}(A,H) = \mathrm{Res}(A,I,2^i)$ and the new H. We now show the preconditions for the subroutines hold. The preconditions for `SPR` in the first line of the loop is: $\|I\| \leq X^{2^{i-1}}/\bar{X}$ and $n\|A\| \leq X/\bar{X}$. The latter is satisfied by the precondition of the algorithm. The former is satisfied because $i \geq 1$. The precondition for `SPL` in the first line is: $n\|\mathtt{SPR}(A,L)\| \leq X/\bar{X}$. Corollary 19 gives $n\|\mathtt{SPR}(A,L)\| \leq n^2\|A\|$, which is $\leq X/\bar{X}$, again using the precondition of the algorithm. Similarly, the preconditions for the second calls to `SPR` and `SPL` also hold. This ends the

inductive proof of correctness.

The algorithm will fail only if one of the $2(l-1)$ calls to `CertLeft` inside `SPL` fails. Lemma 22 bounds the number of bad choices for $\bar{t}$ by $10n^2(l-1)$.

Theorem 27 *Algorithm 26* (`Segment`) *is correct. The cost of the algorithm is* $O(l\,\mathsf{MM}(n,X) + \overline{\mathsf{MM}}(n,X))$ *bit operations. Corresponding to a valid input* $[(X,*,s)](A,l)$*, there are fewer than* $10n^2(l-1)$ *choices for* $\bar{t} \in [2, \bar{X}-3]$ *for which* `Segment`$[(X,t,s)](A,B,l)$ *will fail.*

7 Unimodularity certification

A matrix $A \in \mathbb{Z}^{n\times n}$ is said to be unimodular if A is invertible over $\mathbb{Z}$. The unimodular matrices are precisely those with determinant equal to ± 1. We present an algorithm to assay if a given $A \in \mathbb{Z}^{n\times n}$ is unimodular. Our approach is to assay if the (X,t)-adic expansion of A^{-1} is finite, where X is a power of two.

Algorithm 28 `UniCert`(A)
Input: $A \in \mathbb{Z}^{n\times n}$.
Output: Failure, or true if A is unimodular and false otherwise

(1) **if** $\det(A) \bmod 2 = 0$ **then return** false **fi**;
(2) # Let $\bar{X} := 2^d$ and $X := \bar{X}^s$.
    # Choose $(d,s,k) \in (\mathbb{Z}_{\geq 3}, \mathbb{Z}_{\geq 2}, \mathbb{Z}_{\geq 2})$ to satisfy
    (a) $10n^2(k-1)/(\bar{X}-4) < 1/2$,
    (b) $n^2\|A\| \leq X/\bar{X}$, and
    (c) $(n-1)^{(n-1)/2}\|A\|^{n-1} \leq 2X^{2^k-2}/\bar{X}$.
    # For example, choose (d,s,k) to be lexicographically minimal.
    # Choose $\bar{t}$ uniformly and randomly from $[2, \bar{X}-3]$;
    $t := \bar{t}(1 + \bar{X} + \cdots + \bar{X}^{s-1})$;
(3) # If the call to `Segment` fails then exit early with fail.
    $E :=$ `Segment`$[(X,t,s)](A,k)$;
    if E is the zero matrix **then**
        return true
    else
        return false
    fi

We now prove correctness. Phase 1 checks that $2 \perp \det A$, a condition that must hold if A is unimodular. Now consider phase 3. Condition (b) in phase 1 matches the precondition of Algorithm 26 (`Segment`). Assume that the call to

`Segment` does not fail. Then

$$\mathrm{Trunc}(A^{-1}, 2^k) = \mathrm{Trunc}(A^{-1}, 2^k - 2) + EX^{2^k-1}.$$

On the one hand, suppose A is unimodular. Then entries in A^{-1} are, up to sign, minors of A of dimension $n-1$. Hadamard's inequality gives $\|A^{-1}\| \leq (n-1)^{(n-1)/2}\|A\|^{n-1}$. Thus, condition (c) in phase 2 ensures that k is chosen so that E is the zero matrix (Corollary 8). This shows that true will be returned when the input matrix is unimodular. On the other hand, suppose E is the zero matrix. Then Theorem 24 gives $\mathrm{Res}(A, I, 2^k) = \mathrm{Left}(-A\,\mathrm{Left}(E, 1), 1)$ (that is, also the zero matrix). Considering Lemma 17, the expansion of A^{-1} must be finite in this case. This shows that the return value of true is always correct.

Now we make some remarks about the choice of (s, d, k) in phase (2). Roughly speaking, the exponent d, the bitlength of the guard coefficient $\bar{X}$, corresponds to the number of bits of precision that are sacrificed for the purpose of avoiding and detecting carry propagations. Condition (a) ensures that d is chosen large enough to afford sufficiently many choices for the random shift $\bar{t} \in [2, \bar{X} - 3]$, see Theorem 27. The exponent s should be chosen as small as possible to minimize the cost of the integer arithmetic. To obtain a good cost estimate the algorithm chooses (d, s, k) lexicographically minimal so that the probability of success is at least $1/2$. Then $\log X = \Theta(\log \|A\| + \log n)$ and $k = O(\log n)$. The next result now follows from Theorem 27.

Theorem 29 *Algorithm 28* (`UniCert`) *is correct and fails with probability less than* $1/2$. *The running time of algorithm is* $O((\log n)\mathsf{MM}(n, X) + \overline{\mathsf{MM}}(n, X))$ *bit operations, where* $\log X = \Theta(\log \|A\| + \log n)$.

Using the upper bounds $\mathsf{MM}(n, X) = O(\mathsf{MM}(n)\mathsf{B}(\log X + \log n))$ and $\overline{\mathsf{MM}}(n, X) = O((\log n)\mathsf{MM}(n)\mathsf{B}(\log X + \log n))$ gives the following.

Corollary 30 *Let* $A \in \mathbb{Z}^{n\times n}$ *be given. There exists a Las Vegas algorithm that assays if* A *is unimodular with an expected number of* $O((\log n)\mathsf{MM}(n)\mathsf{B}(\log ||A|| + \log n))$ *bit operations.*

8 The sparse inverse expansion

We are working over an (X, t)-adic shifted number system. Let $A \in \mathbb{Z}^{n\times n}$ be nonsingular, and suppose $X \perp \det A$. For $i \geq 1$, define

- $R^{(i)} = \mathrm{Res}(A, I, 2^i)$, and
- $M^{(i)} = \mathrm{Left}(\mathrm{Trunc}(A^{-1}, 2^i)\, R^{(i)}, 2^i)$.

Then

$$\begin{aligned}\mathrm{Trunc}(A^{-1}, 2^{i+1}) &= \mathrm{Trunc}(A^{-1}, 2^i) + \mathrm{Trunc}(A^{-1}R^{(i)}, 2^i)X^{2^i} \\ &= \mathrm{Trunc}(A^{-1}, 2^i) + \mathrm{Trunc}(A^{-1}, 2^i)R^{(i)}X^{2^i} - M^{(i)}X^{2^{i+1}}\end{aligned}$$

This gives the following.

Theorem 31 $\mathrm{Trunc}(A^{-1}, 2^{i+1}) = \mathrm{Trunc}(A^{-1}, 2^i)(I + R^{(i)}X^{2^i}) - M^{(i)}X^{2^{i+1}}$.

As an illustration of Theorem 31, let $x = 10$ and consider the computation of $\mathrm{mod}(7^{-1}, x^{16}) = 3 + 4x + 1x^2 + 7x^3 + 5x^4 + 8x^5 + 2x^6 + \cdots + 1x^{14} + 7x^{15}$. Repeated application of the theorem gives:

$$\begin{aligned}\mathrm{mod}(7^{-1}, x^{16}) &= \mathrm{mod}(7^{-1}, x^8)(1 - 3x^8) + 2x^{16} \\ &= ((\mathrm{mod}(7^{-1}, x^4)(1 - 5x^4) + 4x^8)(1 - 3x^8) + 2x^{16} \\ &= ((43(1 - 3x^2) + 2x^4)(1 - 5x^4) + 4x^8)(1 - 3x^8) + 2x^{16} \\ &= 7142857142857143\end{aligned}$$

The second to last expression, an example of the sparse inverse expansion, expresses the sixteen digit number $\mathrm{mod}(7^{-1}, x^{2^4})$ as an arithmetic expression involving the two digit number $\mathrm{mod}(7^{-1}, x^2) = 43$ and the six single digit numbers $-3, -2, -5, -4, -3, -2$. In general, $\mathrm{Trunc}(A^{-1}, 2^k)$ can be expressed as an arithmetic expression involving $\mathrm{Trunc}(A^{-1}, 2)$ and the $2(k-1)$ matrices $R^{(i)}$ and $M^{(i)}$, $1 \le i \le k-1$.

Let $B \in \mathbb{Z}^{m \times n}$. Theorem 31 gives the following scheme, $k \ge 1$:

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

For computing $\mathrm{Trunc}(BA^{-1}, 2^k)$, $k \ge 2$, we can set the loop to go up to $k-2$ and return $\mathrm{Trunc}(C(I + R^{(k-1)}X^{2^{k-1}}), 2^k)$ instead. Also, we can optimize the computation slightly by computing all intermediate quantities modulo X^{2^k}.

This gives the following scheme, $k \geq 1$:

$$\mathrm{Trunc}(BA^{-1}, 2^k) := \left[\begin{array}{l} C := B\,\mathrm{Trunc}(A^{-1}, 2); \\ \textbf{for } i \textbf{ from } 1 \textbf{ to } k-2 \textbf{ do} \\ \quad \bar{C} := \mathrm{Trunc}(C, 2^k - 2^i); \\ \quad \bar{B} := \mathrm{Trunc}(B, 2^k - 2^{i+1}); \\ \quad C := \mathrm{Trunc}(C + \bar{C}R^{(i)}X^{2^i} - \bar{B}M^{(i)}X^{2^{i+1}}, 2^k) \\ \textbf{od}; \\ \bar{C} := \mathrm{Trunc}(C, 2^k - 2^{k-1}); \\ \textbf{return } \mathrm{Trunc}(C + \bar{C}R^{(k-1)}X^{2^{k-1}}, 2^k) \end{array}\right.$$

Algorithm 32 (`RSeries`) follows the above scheme exactly, but the matrices $R^{(i)}$ and $M^{(i)}$ are computed as required on the fly.

Algorithm 32 `RSeries`$[(X, t, s)](A, B, k)$
Remark: $(X, t) = (\bar{X}^s, \bar{t}(1 + \bar{X} + \cdots + \bar{X}^{s-1}))$
Input: $A \in \mathbb{Z}^{n\times n}$, $B \in \mathbb{Z}^{m\times n}$ with $\mathrm{Trunc}(B, 2^k) = B$, $k \geq 2$.
Output: Failure, or $\mathrm{Trunc}(BA^{-1}, 2^k)$.
Condition: $X \perp \det A$ and $n^2\|A\| \leq X/\bar{X}$.

(1) $L := \mathrm{Trunc}(A^{-1}, 1)$
$H := \mathrm{Trunc}(\mathtt{MUL}(L, \mathrm{Left}(I - \mathtt{MUL}(A, L), 1)), 1)$;
$E := L + HX$;
$C := \mathrm{Trunc}(\mathtt{MUL}(B, L) + \mathtt{MUL}(B, H)X, 2^k)$;
(2) # If any call to `SPL` fails then exit early with fail.
for i **from** 1 **to** $k-2$ **do**
$(L, *) := \mathtt{SPL}(E, \mathtt{SPR}(A, L))$;
$R := \mathtt{SPR}(A, H)$;
$(H, M) := \mathtt{SPL}(E, R)$;
$E := L + HX$;
$\bar{C} := \mathrm{Trunc}(C, 2^k - 2^i)$;
$\bar{B} := \mathrm{Trunc}(B, 2^k - 2^{i+1})$;
$C := \mathrm{Trunc}(C + \mathtt{MUL}(\bar{C}, R)X^{2^i} - \mathtt{MUL}(\bar{B}, M)X^{2^{i+1}}, 2^k)$
od;
(3) $R := \mathtt{SPR}(A, H)$;
$\bar{C} := \mathrm{Trunc}(C, 2^k - 2^{k-1})$;
return $\mathrm{Trunc}(C + \mathtt{MUL}(\bar{C}, R)X^{2^{k-1}}, 2^k)$

The algorithm is almost identical to Algorithm 26 (`Segment`). The proof of correctness is analogous. The only difference here is that we keep track of the intermediate quantities R and M and use them to update C.

As for complexity, the algorithm performs the same computations as Algorithm 26 (`Segment`), except for the lines involving B or C (the last line in each phase). We can derive a slightly better bound for the cost of these lines if we assume that the input B is provided as an (X,t)-adic expansion and give the output in (X,t)-adic form. Note that the computation of $\bar{C}$ and $\bar{B}$ is free in this case. We need to bound the cost of all the calls to `MUL`. By Corollaries 7 and 19, all the quantities R and M computed by the algorithm satisfy $\|R\|, \|M\| \leq n\|A\|$, which by the precondition of the algorithm is $< X$. Consider the computation of $\texttt{MUL}(B, L)$ in phase 1. Suppose $B = B_0 + B_1 X + \cdots + B_{2^k-1} X^{2^k-1}$. Then compute

$$\left[\begin{array}{c} D_0 \\ \hline D_1 \\ \hline \vdots \\ \hline D_{2^k-1} \end{array}\right] = \left[\begin{array}{c} B_0 \\ \hline B_1 \\ \hline \vdots \\ \hline B_{2^k-1} \end{array}\right] L$$

at a cost bounded by $\lceil 2^k m/n \rceil \mathsf{MM}(n, X)$ bit operations. Since we are working modulo X^{2^k} it will suffice to compute the first 2^k coefficients of the (X,t)-adic expansion $\texttt{MUL}(B, L) = F_0 + F_1 X + F_2 X^2 + \cdots$. These are computed from $D_0, D_1, \ldots, D_{2^k-1}$ as follows.

$R :=$ the $m \times n$ zero matrix;
for i **from** 0 **to** $2^k - 1$ **do**
  $R := R + D_i$;
  $F_i := \text{Trunc}(R, 1)$;
  $R := \text{Left}(R, 1)$
od

At the start of each loop iteration $R = \text{Left}(\text{Trunc}(B, i)L, i)$, which shows that $\|R\| \leq n\|L\|$ throughout (Corollary 7). Thus, the code fragment above has cost bounded by $O(nm2^k \mathsf{M}(\log X + \log n))$, which by the definition of $\mathsf{MM}(n, X)$ is bounded by $O(\lceil m2^k/n \rceil \mathsf{MM}(n, X))$. The remaining matrix multiplications involving $\bar{C}$ and $\bar{B}$ are accomplished similarly. Note that multiplication by a power of X is for free if we assume we are working in (X,t)-adic representations.

Theorem 33 *Algorithm 32* (`RSeries`) *is correct. The cost of the algorithm is* $O(k\lceil m2^k/n \rceil \mathsf{MM}(n, X) + \overline{\mathsf{MM}}(n, X))$ *bit operations, assuming the input parameter B and output are given in (X,t)-adic form. Corresponding to a valid input* $[(X, *, s)](A, B, k)$*, there are fewer than* $10n^2(k-2)$ *choices for* $\bar{t} \in [2, \bar{X} - 3]$ *for which* $\texttt{RSeries}[(X, t, s)](A, B, k)$ *will fail.*

Considering the transpose situation gives the existence of an algorithm with the following specification.

Algorithm 34 `LSeries`$[(X,t,s)](A,B,k)$
Remark: $(X,t) = (\bar{X}^s, \bar{t}(1+\bar{X}+\cdots+\bar{X}^{s-1}))$
Input: $A \in \mathbb{Z}^{n\times n}$, $B \in \mathbb{Z}^{n\times m}$ with $\mathrm{Trunc}(B,2^k) = B$, $k \geq 2$.
Output: Failure, or $\mathrm{Trunc}(A^{-1}B, 2^k)$.
Condition: $X \perp \det A$ and $n^2\|A\| \leq X/\bar{X}$.

9 Reduction of linear system solving to matrix multiplication

In this section we show how to apply the algorithm of the previous section to get Las Vegas reductions to matrix multiplication for a number of linear algebra problems on integer matrices. Before we can apply Algorithm 34 (`LSeries`) we need to initialize a suitable $(X,*,s)$-adic number system. We will choose $\bar{X}$ to be a power of a prime p. The next lemma and corollary recall how to find a suitable p quickly using randomization.

Lemma 35 *Let $B \in \mathbb{Z}^{n\times m}$ be given, $n \geq m$. A prime p with $\log p = O(\log m + \log\log\|B\|)$ and such that the rank of $B \bmod p$ over $\mathbb{Z}/(p)$ is equal to the rank of B over $\mathbb{Z}$ with probability at least $1/2$ can be found with $O(m(\log m + \log\|B\|)(\log m + \log\log\|B\|))$ bit operations.*

PROOF. Let M be a nonzero minor of B of maximal dimension. Then $|M| \leq D$ where $D = m^{m/2}\|B\|^m$ (Hadamard's bound). Let $l = 6 + \lceil \ln\ln D \rceil$ and set Λ to be a set of $2\lceil \lceil \log D \rceil/(l-1) \rceil$ primes between 2^{l-1} and 2^l. Then fewer than half the primes in Λ divide M. In [17, Theorem 1.8], based on bounds by [36], it is shown that there are at least this many primes in this range, and that the construction of Λ can be accomplished with $O((\log D)(\log\log D))$ bit operations using the sieve of Eratosthenes (see [28, Section 4.5.4]). Choose p uniformly and randomly from Λ. For the cost estimate note that $\log D = O(m(\log m + \log\|B\|))$ and $\log\log D = O(\log m + \log\log\|B\|)$. □

Now suppose that B is known to have full column rank over $\mathbb{Z}$. Then for all but a finite number of primes p, $B \bmod p$ will have full column rank over $\mathbb{Z}/(p)$. Testing if $B \bmod p$ over $\mathbb{Z}/(p)$ has full column rank costs $O((n/m)\overline{\mathsf{MM}}(m,p))$ bit operations and is accomplished by computing, over $\mathbb{Z}/(p)$, an LSP decomposition [22] or an echelon transform [41, Chapter 2]. These algorithms also identify a minor of maximal rank. Repeatedly choosing primes p until $B \bmod p$ has full column rank over $\mathbb{Z}/(p)$ gives the following.

Corollary 36 *Let a full column rank $B \in \mathbb{Z}^{n\times m}$ be given. A prime p with $\log p = O(\log m + \log\log\|B\|)$ together with a permutation matrix P such that the principal $m \times m$ submatrix of $PB \bmod p$ is nonsingular over $\mathbb{Z}/(p)$ can be*

found with an expected number of $O(n(\log m)(\mathsf{MM}(m)/m)\mathsf{B}(\log\|B\|+\log m))$ *bit operations.*

Nonsingular rational system solving

Consider the problem of computing $A^{-1}b \in \mathbb{Q}^{n\times 1}$ for a given nonsingular $A \in \mathbb{Z}^{n\times n}$ and $b \in \mathbb{Z}^{n\times 1}$. Denominators of entries in $A^{-1}b$ will be divisors of $\det A$. Hadamard's bound gives $|\det A| \le n^{n/2}\|A\|^n$. Similarly, Cramer's rule gives $\|(\det A)A^{-1}b\| \le n^{n/2}\|A\|^{n-1}\|b\|$. The most effective methods for computing $A^{-1}b$ are based on X-adic lifting, see [31, Section 5] for a brief survey. The idea is to compute $A^{-1}b \bmod X^n \in \mathbb{Z}^{n\times 1}$ for $X^n > 2|\det A|\|(\det A)A^{-1}b\|$, and then recover $A^{-1}b \in \mathbb{Q}^{n\times 1}$ using rational reconstruction. The best previous methods have a cost in terms of bit operations that is cubic in n. Here we show how to use Algorithm 34 (`LSeries`) presented in the previous section to reduce the exponent of n to that of matrix multiplication.

Use Corollary 36 to find a prime $p = O(\log n + \log\log\|A\|)$ such that $p \perp \det A$. Let $k = \max(2, \lceil \log n \rceil)$ and choose $d \in \mathbb{Z}_{\ge 1}$ minimal so that $\bar{X} := p^d$ satisfies

$$10n^2(k-2)/(\bar{X}-4) < 1/2. \tag{8}$$

Now choose $s \in \mathbb{Z}_{\ge 2}$ minimal so that $X := \bar{X}^s$ satisfies

$$n^2\|A\| \le X/\bar{X} \quad \text{and} \quad X^n > 2\lfloor n^{n/2}\|A\|^{n-1}\|b\|\rfloor\lfloor n^{n/2}\|A\|^n\rfloor. \tag{9}$$

The first part of condition (9) is a precondition of Algorithm 34 (`LSeries`). Repeatedly choose $\bar{t} \in [2, \bar{X}-3]$ uniformly and randomly until $\texttt{LSeries}[(X,t,s)](A,b,k)$ does not return fail, $t = \bar{t}(1+\bar{X}+\cdots+\bar{X}^{s-1})$. By Theorem 33 and condition (8), any particular call to `LSeries` will fail with probability $< 1/2$. Thus, the expected number of calls to `LSeries` is < 2. Finally, reconstruct $A^{-1}b$ from $\mathrm{Trunc}(A^{-1}b, 2^k)$, the output of the successful call to $\texttt{LSeries}(A,b,k)$, using rational reconstruction.

Algorithm 34 (`LSeries`) assumes that input and output is given in X-adic representation. If $a = O(X^n)$ then the X-adic expansion of a can be computed from the binary expansion (or vice versa) with $O(\mathsf{M}(n\log X)\log n)$ bit operations [15, Theorem 9.17], which is bounded more simply by $O(\mathsf{B}(n\log X))$ bit operations. Thus, the binary expansion and rational reconstruction of $\mathrm{Trunc}(A^{-1}B, 2^k)$, which is given in X-adic form, can be computed with $O(n\,\mathsf{B}(n\log X))$ bit operations, which simplifies to $O(\mathsf{MM}(n)\mathsf{B}(\log X))$ if we make the assumption that $\mathsf{B}(t) = O(\mathsf{MM}(t)/t)$. We get the following as a corollary of Theorem 33. Note that condition (9), together with the minimality of s, gives that $\log X = \Theta(\log\|A\| + (\log\|b\|)/n + \log n)$.

Theorem 37 *There exists a Las Vegas algorithm that takes as input a nonsingular $A \in \mathbb{Z}^{n\times n}$ and $b \in \mathbb{Z}^{n\times 1}$, and returns as output the vector $A^{-1}b \in \mathbb{Q}^{n\times 1}$. The expected cost of the algorithm is $O((\log n)\mathsf{MM}(n)\mathsf{B}(d + \log n))$ bit operations, where d is a bound for both $\log \|A\|$ and $(\log \|b\|)/n$. This result assumes that $\mathsf{B}(t) = O(\mathsf{MM}(t)/t)$.*

Computing the largest invariant factor

The largest invariant factor of a nonsingular matrix $A \in \mathbb{Z}^{n\times n}$ is equal to the greatest common divisor of all minors of A of dimension $n-1$. The largest invariant factor can be computed with high probability as the least common multiple of the denominators of $A^{-1}b^{(1)}$ and $A^{-1}b^{(2)}$ for randomly chosen $b^{(i)}$, first observed in [33] for polynomial matrices. In [13, Theorem 2.1] it is shown that choosing entries in $b^{(i)}$ uniformly and randomly from $\{0, \ldots, M-1\}$ with $M = 6 + \lceil 2n(\log n + \log \|A\|)\rceil$ gives a probability of success at least $1/3$. We get the following as a corollary of Theorem 37

Theorem 38 (`LargestInvariantFactor`(A)) *There exists a Monte Carlo algorithm that takes as input a nonsingular $A \in \mathbb{Z}^{n\times n}$, and returns as output a factor of the largest invariant factor of A, equal to the largest invariant factor with probability at least $1/3$. The cost of the algorithm is $O((\log n)\mathsf{MM}(n)\mathsf{B}(\log n + \log \|A\|))$ bit operations. This result assumes that $\mathsf{B}(t) = O(\mathsf{MM}(t)/t)$.*

Certified dense linear system solving

Let $A \in \mathbb{Z}^{n\times m}$ and $b \in \mathbb{Z}^{n\times 1}$ be given. Suppose that $Ax = b$ admits a rational solution vector x. If d is the smallest positive integer such that dx is integral, and d is minimal among all solutions to the system, then x is a solution with *minimal denominator*.

The certified linear system solving problem [31] is to either prove that $Ax = b$ has no rational solution vector (that is, prove that the system is inconsistent over the field of rational numbers) or to compute a solution vector with minimal denominator. Algorithm *CertifiedSolver* [31, Page 506] reduces the problem to solving an expected constant number of nonsingular rational systems plus some additional work. We get the following as a corollary of [31, Proposition 44] and Theorem 37.

Theorem 39 *Let $A \in \mathbb{Z}^{n\times m}$ and $b \in \mathbb{Z}^{n\times 1}$ be given. There exists a Las Vegas algorithm that computes a solution to the certified linear system solving problem with input (A, b) with an expected number of $O(nm(\log r)(\mathsf{MM}(r)/r^2)\mathsf{B}(d+$*

$\log m))$ *bit operations, where* r *is the rank of* A *and* d *is a bound for both* $\log \|A\|$ *and* $(\log \|b\|)/r$*. This result assumes that* $\mathsf{B}(t) = O(\mathsf{MM}(t)/t)$*.*

We remark that if $\mathsf{MM}(r) = O(r^3)$ then the cost estimate of Theorem 39 becomes $O(nmr\,\mathsf{B}(d+\log m))$ bit operations, see [31, Corollary 45]. Thus, our incorporation of matrix multiplication (the reduction of nmr to $nm(\mathsf{MM}(r)/r^2)$) comes at the cost of introducing a factor of $\log r$.

Certified testing of matrix rank maximality

Testing if a given matrix $A \in \mathbb{Z}^{n\times m}$ has full column rank is equivalent to determining if $r := \operatorname{rank}(A) < m$. Our approach is to either find a prime p such that $A \bmod p$ over $\mathbb{Z}/(p)$ has rank m (in which case A is certified to have rank m over $\mathbb{Z}$) or to find a $b \in \mathbb{Z}^{1\times m}$ such that the system $xA = b$ is inconsistent over the field of rational numbers (in which case A is certified to have rank strictly less than m). Compute a prime p as in Lemma 35, then compute the rank $\bar{r}$ of $A \bmod p$ over $\mathbb{Z}/(p)$. We must have $\bar{r} \le r$, and with probability at least $1/2$ we have $\bar{r} = r$. If $\bar{r} = m$ then we're done. Otherwise, choose a random $b \in \{0,1\}^{1\times m}$ and try to solve the system $xA = b$ using the algorithm supporting Theorem 39. If $r < n$ then, by [31, Corollary 12], with probability at least $1/2$ the vector b does not lie in the row space of A over $\mathbb{Q}$, in which case the system will be determined to be inconsistent. If $xA = b$ is consistent then choose another prime and repeat. Since the probability of failure in either case ($r < m$ or $r = m$) is less than $1/2$, the expected number of repetitions is fewer than 2.

Theorem 40 *Let* $A \in \mathbb{Z}^{n\times m}$ *be given,* $n \ge m$*. There exists a Las Vegas algorithm that assays if* A *has rank* m *with an expected number of*

$$O(n(\log m)(\mathsf{MM}(m)/m)\mathsf{B}(\log \|A\| + \log m))$$

bit operations. This result assumes that $\mathsf{B}(t) = O(\mathsf{MM}(t)/t)$*.*

10 High-order lifting

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

$$A^{-1}B = * + *X + \cdots + \overbrace{*X^h + \cdots + *X^{h+k-1}}^{HX^h} + *\,X^{h+k} + \cdots \tag{10}$$

If $h = 0$ we can use Algorithm 34 (`LSeries`) to compute H. In high-order lifting, what is important is that h be larger than some specified bound l. The particular value of h is not important, only that $h > l$. The cost of the algorithm is linear in $\log l$. This is important because in typical applications $l \gg k$.

Algorithm 41 `HighOrderLift`$[(X, t, s)](A, B, l, k)$
Remark: $(X, t) = (\bar{X}^s, \bar{t}(1 + \bar{X} + \cdots + \bar{X}^{s-1}))$
Input: $A \in \mathbb{Z}^{n\times n}$, $B \in \mathbb{Z}^{n\times m}$, $l \geq 2$, k a power of two.
Output: Failure, or Left(Trunc$(A^{-1}B, h + k), h)$ for some $h > l$.
Condition: $X \perp \det A$ and $n^2\|A\| \leq X/\bar{X}$.

(1) # If the call to `LSeries` fails then exit early with fail.
$\bar{k} :=$ the smallest integer ≥ 2 such that $\|B\| \leq X^{2^{\bar{k}}}/\bar{X}$;
$\bar{D} :=$ Left(`LSeries`$(A, B, \bar{k}), 2^{\bar{k}} - 1)$;
$\bar{R} :=$ `SPR`$(A, \bar{D})$;
(2) # If the call to `Segment` or `SPL` fails then exit early with fail.
$\bar{l} :=$ the smallest integer ≥ 2 such that $2^{\bar{l}} + 2^{\bar{k}} > l$;
$E :=$ `Segment`$(A, \bar{l})$;
$(D, *) :=$ `SPL`$(E, \bar{R})$;
$R :=$ `SPR`(A, D);
(3) # If the call to `LSeries` fails then exit early with fail.
$H :=$ `LSeries`$(A, R, \log k)$;
return H

Given l, the algorithm here chooses $h := 2^{\bar{l}} + 2^{\bar{k}}$. The purpose of phase 1 is to reduce a possible large magnitude right hand side B to a small magnitude residue $\bar{R}$. In phase 1 we choose $\bar{k}$ to ensure the preconditions for Subroutine 25 (`SPR`) are satisfied. Note that $2^{\bar{k}} < 2\log_X(\|B\|\bar{X}) = O((\log\|B\|)/(\log X))$. After phase 1 finishes, $\bar{R} = \mathrm{Res}(A, B, 2^{\bar{k}})$ and $\|\bar{R}\| \leq n\|A\|$ (Corollary 19). In phase 2 we choose $\bar{l}$ to satisfy $2^{\bar{l}} + 2^{\bar{k}} > l$. After phase 2 finishes, $R = \mathrm{Res}(A, \bar{R}, 2^{\bar{l}})$, which is equal to $\mathrm{Res}(A, B, 2^{\bar{l}} + 2^{\bar{k}})$. Finally, the high-order lift is computed in phase 3.

The costs of phases 1, 2 and 3 are dominated by the calls to `LSeries`$(A, B, \bar{k})$, `Segment`$(A, \bar{l})$ and `LSeries`$(A, R, \log k)$, respectively. By Theorems 27 and 33 these calls will have cost bounded by $O((\log n)\mathsf{MM}(n, X) + \overline{\mathsf{MM}}(n, X))$ bit operations if all of

$$\bar{k}\lceil m2^{\bar{k}}/n\rceil, \qquad \bar{l}, \quad \text{and} \quad (\log k)\lceil mk/n\rceil$$

are $O(\log n)$; we can ensure this bound by making some assumptions on the input paramaters. First, assume that $m \times k = O(n)$. Then $\log k = O(\log n)$ and $m = O(n)$. Second, assume that $m \times (\log\|B\|)/(\log X) = O(n)$. Then $m \times 2^{\bar{k}} = O(n)$ and $\bar{k} = O(\log n)$. Third, assume that $\log l = O(\log n)$. We get the following result.

Theorem 42 *Algorithm 41* (`HighOrderLift`) *is correct. If* $\log l = O(\log n)$ *and both* $m \times k$ *and* $m \times (\log \|B\|)/(\log X)$ *are* $O(n)$*, then the cost of the algorithm is* $O((\log n)\mathsf{MM}(n, X) + \overline{\mathsf{MM}}(n, X))$ *bit operations, assuming the input parameter* B *and the output are given in* (X, t)*-adic form.*

The number of calls to `SPL` depends on the input parameters l, k and $\|B\|$. If these parameters satisfy $\log l, \log k = O(\log n)$ and $\log_X \|B\| = O(n)$, then we may easily derive the estimate $O(\log n)$ for the number of calls, but for actual applications of the algorithm (for example, in the next section) we will need an explicit bound. Using

$$\bar{k} = \max(2, \lceil \log(\log_X \|\bar{X}B\|) \rceil) \quad \text{and} \quad \bar{l} = \max(2, \lceil \log(l - 2^{\bar{k}} + 1) \rceil) \tag{11}$$

gives the following.

Theorem 43 *Corresponding to a valid input* $[(X, *, s)](A, B, l, k)$ *to Algorithm 41* (`HighOrderLift`)*, there are fewer than* $10n^2(\bar{k} + \bar{l} + \log k)$ *choices for* $\bar{t} \in [2, \bar{X} - 3]$ *for which* `HighOrderLift`$[(X, t, s)](A, B, l, k)$ *will fail,* $\bar{k}$ *and* $\bar{l}$ *as in (11).*

11 Integrality certification

Let $A \in \mathbb{Z}^{n \times n}$ be nonsingular, $\det A \perp X$. Let $B \in \mathbb{Z}^{n \times m}$ and $T \in \mathbb{Z}^{m \times m}$. This section presents an algorithm to assay if $A^{-1}BT$ is integral.

For any $h \geq 0$, post multiplying both sides of

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

by T gives the following observation.

Lemma 44 $A^{-1}BT$ *is integral if and only if* $A^{-1}\mathrm{Res}(A, B, h)T$ *is integral.*

The next result is an extension of Corollary 19, which gave a condition on h for $\|\mathrm{Res}(A, B, h)\|$ to be small (that is, independent of the size of $\|B\|$).

Lemma 45 *Assume* $A^{-1}BT$ *is integral and* h *satisfies* $\|A^{-1}BT\| \leq 2X^h/\bar{X}$*. Then* $A^{-1}\mathrm{Res}(A, B, h)T$ *is integral and* $\|A^{-1}\mathrm{Res}(A, B, h)T\| \leq m\|T\|$*.*

PROOF. Notice that

$$A^{-1}BT = \overbrace{\mathrm{Trunc}(A^{-1}B, h)T}^{\|\cdot\| < m\|T\|X^h} + A^{-1}\mathrm{Res}(A, B, h)TX^h. \tag{12}$$

By the assumption on h we have

$$A^{-1}BT = \text{Trunc}(\text{Trunc}(A^{-1}B, h)T, h). \tag{13}$$

Subtract (13) from (12) and divide both sides by X^h (an exact division) to get

$$0 = \overbrace{\text{Left}(\text{Trunc}(A^{-1}B, h)T, h)}^{\|\cdot\| \le m\|T\|} + A^{-1}\text{Res}(A, B, h)T.$$

The magnitude bound in the last formula follows from Corollary 7. □

The next equation defines the quantities S and C, and is obtained by applying $\text{Trunc}(\cdot, h+k)$ to both sides of (12).

$$\overbrace{\text{Trunc}(A^{-1}BT, h+k)}^{S} = \text{Trunc}(\overbrace{\text{Trunc}(A^{-1}B, h)T}^{\|\cdot\| < m\|T\|X^h} + \overbrace{\text{Trunc}(A^{-1}\text{Res}(A, B, h)T, k)}^{C} X^h, h+k) \tag{14}$$

Theorem 46 *Assume h satisfies $\|A^{-1}BT\|, \|BT\| \le 2X^h/\bar{X}$ and k satisfies $2nm\|T\|\|A\| \le 2X^k/\bar{X}$. Then $A^{-1}BT$ is integral if and only if $\|C\| \le m\|T\|$.*

PROOF. (If:) It follows from the definition of S (left hand side of (14)) that $\text{Trunc}(AS, h+k) = \text{Trunc}(BT, h+k)$. If both $\|AS\|$ and $\|BT\|$ are $\le 2X^{h+k}/\bar{X}$ then $AS = BT$ (Corollary 8) and it follows that $S = A^{-1}BT$, in which case $A^{-1}BT$ is integral. The assumption on h gives the bound for $\|BT\|$. Now assume $\|C\| \le m\|T\|$. Then $\|S\| < 2m\|T\|X^h$ (cf. (14)), giving the bound $\|AS\| < 2nm\|T\|\|A\|X^h$, which by the assumption on k is $\le 2X^{h+k}/\bar{X}$.
(Only If:) Follows from Lemma 45. □

If $A^{-1}BT$ is integral the algorithm returns C, a *left integrality certificate for $A^{-1}B$ with respect to T*.

Algorithm 47 `LIntCert`$[(X, t, s)](A, B, T)$
Remark: $(X, t) = (\bar{X}^s, \bar{t}(1 + \bar{X} + \cdots + \bar{X}^{s-1}))$
Input: $A \in \mathbb{Z}^{n\times n}$, $B \in \mathbb{Z}^{n\times m}$, $T \in \mathbb{Z}^{m\times m}$.
Output: Failure, a left integrality certificate for $A^{-1}B$ with respect to T if $A^{-1}BT$ is over $\mathbb{Z}$, false otherwise.
Condition: $X \perp \det A$ and $n^2\|A\| \le X/\bar{X}$.

(1) # If the call to `HighOrderLift` fails then exit early with fail.
$h :=$ smallest integer such that $mn(n-1)^{(n-1)/2}\|A\|^{n-1}\|B\|\|T\| \le 2X^h/\bar{X}$;

$k :=$ smallest power of two such that $2nm\|T\|\|A\| \leq 2X^k/\bar{X}$;
$H := \texttt{HighOrderLift}(A, B, h, k)$;
(2) $C :=$ Trunc($\texttt{MUL}(H, T), k$);
if $\|C\| \leq m\|T\|$ **then**
  return C
else
  return false
fi

The operation $\texttt{MUL}(H, T)$ can be computed as the product of an $n \times km$ by a $km \times km$ block Toeplitz matrix

$$\left[\begin{array}{c|c|c|c} H_0 & H_1 & \cdots & H_{k-1} \end{array}\right] \left[\begin{array}{c|c|c|c} T_0 & T_1 & \cdots & T_{k-1} \\ \hline & T_0 & \cdots & T_{k-2} \\ \hline & & \ddots & \vdots \\ \hline & & & T_0 \end{array}\right]$$

and then adjusting to get back the (X, t)-adic representation of C, see the discussion after Algorithm 32. In this way, no integer arithmetic with large operands is required.

Now consider the choice of h and k in phase 1. Considering the restrictions on m, $\|B\|$ and $\|T\|$ in the statement of the following theorem, and using the condition that $n^2\|A\| \leq X/\bar{X}$, gives that h is $O(n)$ and k is $O(\log_X \|T\|)$. Thus, all the conditions of Theorem 42 are met and we may apply that theorem to bound the cost of the call to `HighOrderLift`.

Theorem 48 *Algorithm 47* (`LIntCert`) *is correct. If all of m, $m\times(\log\|B\|)/(\log X)$ and $m\times(\log\|T\|)/(\log X)$ are $O(n)$, then the cost of the algorithm is $O((\log n)\mathsf{MM}(n, X)+\overline{\mathsf{MM}}(n, X))$ bit operations, assuming the input parameters B and T and the output are given in (X, t)-adic form.*

The following helper method computes an upper bound on the number of bad choices for the shift $\bar{t} \in [2, \bar{X} - 3]$ — choices for which the call to call to `HighOrderLift` will fail.

$$\texttt{Bad}(A, B, T) := \left[\begin{array}{l} k := \lceil \log(nm\|T\|\|A\|\bar{X}) \rceil; \\ l := \lceil \log_X(mn(n-1)^{(n-1)/2}\|A\|^{n-1}\|B\|\|T\|\bar{X}/2) \rceil; \\ \bar{k} := \max(2, \lceil \log(\log_X \|\bar{X}B\|) \rceil); \\ \bar{l} := \max(2, \lceil \log(l - 2^{\bar{k}} + 1) \rceil); \\ \textbf{return } 10n^2(\bar{k} + \bar{l} + \log k) \end{array}\right.$$

If the assumptions of Theorem 48 are satisfied then $\texttt{Bad}(A, B, T)$ is $O(n^2 \log n)$. To compute an integrality certificate using Algorithm 47 requires knowing a suitable $(X, *, s)$-adic number system. Use Corollary 36 to find a prime $p = O(\log n + \log\log \|A\|)$ such that $p \perp \det A$. Set $\bar{X} := p^d$ and $X := \bar{X}^s$, where $(d, s) \in (\mathbb{Z}_{\geq 1}, \mathbb{Z}_{\geq 2})$ is chosen lexicographically minimal to satisfy the following conditions: $n^2\|A\| \leq X/\bar{X}$ and $\texttt{Bad}(A, B, T)/(X - 4) < 1/2$. Then, for a random choice of $\bar{t} \in [2, \bar{X} - 3]$, the call $\texttt{LIntCert}[(X, t, s)](A, B, T)$ will fail with probability less than $1/2$, $t := \bar{t}(1 + \bar{X} + \cdots + \bar{X}^{s-1})$.

The conversation between X-adic and binary representation can be accomplished in the allotted time if we assume that $\mathsf{B}(t) = O(\mathsf{MM}(t)/t)$.

Theorem 49 ($\texttt{LeftIntegralityCertificate}(A, B, T)$) *There exists a Las Vegas algorithm that takes as input*

- *nonsingular* $A \in \mathbb{Z}^{n\times n}$,
- $B \in \mathbb{Z}^{n\times m}$, *and*
- $T \in \mathbb{Z}^{m\times m}$,

and returns as output

- *false, if* $A^{-1}BT$ *is not integral, or*
- *a left integrality certificate* C *for* $A^{-1}B$ *with respect to* T.

If m *is* $O(n)$ *and both* $m \times (\log \|B\|)/n$ *and* $m \times (\log \|T\|)/n$ *are* $O(\log n + \log \|A\|)$, *then the expected cost of the algorithm is* $O((\log n)\mathsf{MM}(n)\mathsf{B}(\log n + \log \|A\|))$ *bit operations. This result assumes that* $\mathsf{B}(t) = O(\mathsf{MM}(t)/t)$.

Note that a right integrality certificate for BA^{-1} with respect to T is equal to the transpose of a left integrality certificate of Transpose(BA^{-1}) with respect to Transpose(T).

12 Preconditioners for the determinant

Our algorithm to compute the determinant requires some preconditioning of the input matrix. This section is reasonably self-contained, but further background material can be found in [32,41,43]. We first recall some definitions about the Hermite and Smith canonical forms of matrices.

The notation StackMatrix(A_1, A_2) is defined by

$$\text{StackMatrix}(A_1, A_2) = \left[\frac{A_1}{A_2}\right].$$

A matrix A is a *left multiple* of B if $A = *B$ for a matrix $*$ over $\mathbb{Z}$. *Right multiple* is defined analogously. A matrix $G \in \mathbb{Z}^{m\times m}$ is a row basis for a full column rank $A \in \mathbb{Z}^{n\times m}$ if A and G are left multiples of each other. Corresponding to every full column rank $A \in \mathbb{Z}^{n\times m}$ is a unimodular (invertible over $\mathbb{Z}$) matrix $U \in \mathbb{Z}^{n\times n}$ such that

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

with all entries in H nonnegative, and off-diagonal entries h_{*j} strictly smaller than the diagonal entry h_j in the same column. The principal nonsingular submatrix H is the unique *Hermite row basis* of A. The product $h_1h_2\cdots h_k$ of the first k diagonal entries is equal to the gcd of all $k \times k$ minors of the first k columns of A, $1 \le k \le n$. If $n = m$ and A is nonsingular, then $h_1h_2\cdots h_n = \det H = |\det A|$.

Our first result, Theorem 50, is inspired by and based on [13, Section 6]. To clarify our starting point from [13], and indicate what our extension is, we begin by giving an example of a special case of the theorem. Suppose $B \in \mathbb{Z}^{n\times(n-10)}$ has entries chosen uniformly and randomly from $\{0, 1, 2, \ldots, n-1\}$. Then [13, Section 6] shows that with probability greater than 7/8 the matrix B has full column rank $n - 10$ over $\mathbb{Z}$ and, moreover, that for every prime p the matrix B mod p over the field $\mathbb{Z}/(p)$ has rank at least $n - 11$ (that is, at most one less than full column rank.[2])

We extend this result in two ways. First, suppose $C \in \mathbb{Z}^{5\times(n-10)}$ also has entries chosen uniformly and randomly from $\{0, 1, 2, \ldots, n-1\}$. Then we show that with probability at least 4/5 the matrix $F := \text{StackMatrix}(B, C) \in \mathbb{Z}^{(n+5)\times(n-10)}$ has Hermite basis I_{n-10}, which is equivalent to saying that for every p the matrix F mod p over $\mathbb{Z}/(p)$ has full column rank. Second, consider a rectangular matrix $A \in \mathbb{Z}^{n\times m}$, $n \ge m + 15$, with full column rank and Hermite row basis H. Theorem 50 states that we can extend A to a nearly square

[2] The technique used was to give a lower bound on the probability that the analogous result held for the submatrix of B comprised of the first i columns, considering $i = 1, 2, \ldots, n-10$ in succession, and basing the estimate for $i \ge 2$ conditionally on the estimate for $i - 1$.

matrix (only fifteen fewer columns than rows) that has the same Hermite basis of A but augmented with the identity. In other words, if the entries in B and C are well chosen, then

$$\left[\begin{array}{c|c} A & B \\ \hline & C \end{array}\right] \in \mathbb{Z}^{(n+5)\times(n-10)}$$

has Hermite row basis

$$\left[\begin{array}{c|c} H & \\ \hline & I_{n-m-10} \end{array}\right] \in \mathbb{Z}^{(n-10)\times(n-10)}.$$

In the proof we use the fact that AH^{-1} is an integral matrix with Hermite row basis I_m.

Theorem 50 *For $n \geq m + 15$, let $A \in \mathbb{Z}^{n\times m}$ have full column rank and Hermite row basis H. If entries in $B \in \mathbb{Z}^{n\times(n-m-10)}$ and $C \in \mathbb{Z}^{5\times(n-m-10)}$ are chosen uniformly and randomly from $\{0, 1, \ldots, \lambda - 1\}$ with $\lambda = \max(\|A\|, n)$, then the Hermite row basis of*

$$F := \left[\begin{array}{c|c} AH^{-1} & B \\ \hline & C \end{array}\right] \in \mathbb{Z}^{(n+5)\times(n-10)}$$

is equal to I_{n-10} with probability at least $4/5$.

PROOF. If the Hermite basis of F differs from I then there is a prime p so that the rank of F mod p over the field $\mathbb{Z}/(p)$ of integers modulo p drops below $n - 10$. Following [13] we define two events. Let Dep denote the event that $[AH^{-1} \mid B] \in \mathbb{Z}^{n\times(n-10)}$ does not have full column rank $n-10$. Let MDep denote the event that there exists at least one prime p such that $[AH^{-1} \mid B]$ mod p has rank at most $n - 12$ over $\mathbb{Z}/(p)$. It follows from [13] [3] that $\mathsf{Dep} \vee \mathsf{MDep}$ holds with probability less than $1/8$.

Under the assumption that $\neg(\mathsf{Dep} \vee \mathsf{MDep})$ holds, our goal is to bound the probability that there exits a prime p such that F mod p does not have full column rank $n - 10$ over the field $\mathbb{Z}/(p)$ of integers modulo p.

If $\neg\mathsf{MDep}$ is satisfied, then corresponding to each prime p there exists a submatrix R of $[AH^{-1} \mid B]$ of dimension $(n - 11) \times (n - 11)$ such that p does not

[3] We obtain this by first substituting $i = n - 10$ and $\lambda \geq n$ into the bound for $P[\mathsf{MDep}_i]$ given directly before Theorem 6.2, and then simplifying using the estimate $n \geq 15$.

divide $\det R$. Moreover, since AH^{-1} has Hermite row basis I_m, there exists such an R involving all m columns of AH^{-1} and all but one column of B. Without loss of generality (up to a permutation of the rows and columns of F involving B) suppose that R is the principal $(n-11)\times(n-11)$ submatrix of $[AH^{-1} \mid B]$. Consider any choice of the entries in all but the last column c of C, and recall that entries in C are selected independently. Then a necessary condition for $F \bmod p$ to have rank less than $n-10$ is that the following submatrix of F has rank less than $n-10$ over $\mathbb{Z}/(p)$.

$$\left[\begin{array}{c|c} R & *_2 \\ \hline *_1 & c \end{array}\right]$$

The Schur complement of this matrix with respect to R is $c - *_1 R^{-1} *_2$. Thus, we need to bound the probability that $c - *_1 R^{-1} *_2 \bmod p$ is zero over $\mathbb{Z}/(p)$. Following the technique of [13, Section 6] we consider primes $p < \lambda$ and primes $p \geq \lambda$ separately.

If $p < \lambda$, the probability that entries in $c \in \mathbb{Z}^{5\times 1}$ are chosen so that $c - *_1 R^{-1} *_2 \bmod p$ is the zero vector over $\mathbb{Z}/(p)$ is at most $((1+p/\lambda)/p)^5$, since the likelihood that a given entry of c assumes any fixed value, mod p, is bounded by $\lceil \lambda/p \rceil (1/\lambda) \leq (1+p/\lambda)/p$. Summing over all primes $p < \lambda$, and using the fact that $\lambda \geq n \geq 15$, gives $\sum_p ((1+p/\lambda)/p)^5 < \sum_{p\in\{2,3,5,7,11,13\}} ((1+p/15)/p)^5 + \sum_{i\geq 17} (2/i)^5$, which is less than 0.0712. Thus, if $\neg$MDep holds, then the probability that there exists a prime $p < \lambda$ such that $F \bmod p$ has rank less than $n-10$ over $\mathbb{Z}/(p)$ is less than 0.0707.

Now, if p is a prime that is greater than or equal to λ, then the probability that the rank of $F \bmod p$ over $\mathbb{Z}/(p)$ is less than $n-10$ is bounded by $(1/\lambda)^5$, since the likelihood that a given entry of c assumes any fixed value, mod p, is either 0 or $1/\lambda$. Since $\neg$Dep holds there exists a nonsingular $(n-10)\times(n-10)$ submatrix $[\bar{A}H^{-1} \mid \bar{B}]$ of $[AH^{-1} \mid B]$. The determinant d of this submatrix is equal to the determinant of $[\bar{A} \mid \bar{B}]$ divided by $\det H$. Since $\|[\bar{A} \mid \bar{B}]\| \leq \lambda$, the magnitude of d is bounded by $(n-10)!\lambda^{n-10} < \lambda^{2n}$, and the number of primes $p \geq \lambda$ that can divide d is bounded by $\log_\lambda \lambda^{2n} = 2n$. Thus, when neither the events Dep or MDep arise, the probability that there exists a prime $p \geq \lambda$ such that $F \bmod p$ has rank less than $n-10$ over $\mathbb{Z}/(p)$ is at most $2n(1/\lambda)^5 \leq 2n(1/n)^5$, which is less than $2/50625$ for all $n \geq 15$.

Since $1/8 + 0.0707 + 2/50625 < 1/5$, the result follows. $\square$

Recall the definition of the Smith form: corresponding to any matrix $A \in \mathbb{Z}^{n\times m}$ are unimodular matrices $U \in \mathbb{Z}^{n\times n}$ and $V \in \mathbb{Z}^{m\times m}$ such that

$$UAV = \mathrm{Smith}(A) = \mathrm{Diag}(\mathrm{PrincipalSmith}(A), 0),$$

where PrincipalSmith$(A) = \mathrm{Diag}(s_1, s_2, \ldots, s_r)$ is unique, each s_i positive and s_i dividing s_{i+1} for $1 \leq i \leq r-1$, r the rank of A. The s_i are called the invariant factors of A. The product $s_1 s_2 \ldots, s_k$ of the first k invariant factors is equal to the gcd of all $k \times k$ minors of A. Noting that every minor of A is bounded in magnitude by $m!\|A\|^m$ gives the following fact which will be used in the proof of the subsequent lemma.

Fact 51 *There are fewer than $2m$ primes $p \geq \max(\|A\|, m, 2)$ that divide s_r.*

Our next result is similar to [13, Section 3], where the authors give a relationship between the invariant factors of A and the invariant factors of $A + B$, where B is a well chosen rank k perturbation. In the next lemma, let $\mathrm{Diag}(s_1, s_2, \ldots, s_m)$ denote the Smith form of full column rank $A \in \mathbb{Z}^{n \times m}$, and let $\mathrm{Diag}(\sigma_1, \sigma_2, \ldots, \sigma_m)$ denote the principal Smith form of StackMatrix(A, B), for some given B over $\mathbb{Z}$. For a given k, $1 \leq k \leq m$, the lemma states that for a well chosen $B \in \mathbb{Z}^{O(k) \times m}$, $\mathrm{Diag}(1, \ldots, 1, s_1, s_2, \ldots, s_{m-k})$ will be a multiple of $\mathrm{Diag}(\sigma_1, \sigma_2, \ldots, \sigma_m)$.

Lemma 52 *Fix k, $1 \leq k \leq m$, and let $e \geq 10 + 2\log k$. If $B \in \mathbb{Z}^{(k+e) \times m}$ has entries chosen uniformly and randomly from $\{0, 1, \ldots, \lambda - 1\}$ with $\lambda = \max(\|A\|, m, 2)$, then $\sigma_i = 1$ for $1 \leq i \leq k$ and σ_i divides s_{i-k} for $k+1 \leq i \leq m$, with probability at least $1/(4k)$.*

Before proving Lemma 52 we give an example of a special case to illustrate the main ideas of the proof. Suppose $A \in \mathbb{Z}^{m \times m}$ is in Smith form, say $A = \mathrm{Diag}(S_1, S_2)$ where S_2 has dimension $k \times k$. Let e and B be chosen as in the lemma. Decompose $B = \left[\begin{array}{c|c} B_1 & B_2 \end{array}\right]$ where B_2 has dimension $(k+e) \times k$. The lemma states that $\mathrm{Diag}(I_k, S_1)$ will be a multiple of the principal Smith form of

$$\left[\begin{array}{c|c} S_1 & \\ \hline & S_2 \\ \hline B_1 & B_2 \end{array}\right]$$

with probability at least $1/(4k)$. Indeed, a sufficient condition for success is that the principal Smith form of StackMatrix(S_2, B_2) be equal to I_k, for if so, then there exists a sequence of unimodular row and column transformations such that

$$\left[\begin{array}{c|c} S_1 & \\ \hline & S_2 \\ \hline B_1 & B_2 \end{array}\right] \rightarrow \left[\begin{array}{c|c} S_1 & \\ \hline * & I \\ \hline * & \end{array}\right] \rightarrow \left[\begin{array}{c|c} I & \\ \hline & S_1 \\ \hline & * \end{array}\right]$$

A sufficient condition for StackMatrix(S_2, B_2) to have principal Smith form I_k is that that $B_2 \bmod p$ have full column rank k over $\mathbb{Z}/(p)$ for every prime p which divides s_m. Fact 51 will be used to bound the number of primes dividing

s_m. Then, using a counting argument similar to [13, Section 3] and as used in the proof of Theorem 50, we consider primes $p < \lambda$ and $p \geq \lambda$ separately, summing a bound on the probability of failure for a given prime over all the primes to arrive at an overall bound on the probability of failure. The following proof generalizes the above argument to the case where A may be rectangular and not in Smith form.

PROOF. (Of Lemma 52). Let U and V be unimodular matrices such that UAV is in Smith form. Decompose $V = [V_1 \mid V_2]$ where V_2 has dimension $m \times k$. Similarly, decompose the Smith form S of A as StackMatrix(Diag$(S_1, S_2), 0)$ where S_2 has dimension $k \times k$. Then

$$\left[\begin{array}{c|c} U & \\ \hline & I \end{array}\right] \left[\begin{array}{c} A \\ \hline B \end{array}\right] \left[\begin{array}{c|c} V_1 & V_2 \end{array}\right] = \left[\begin{array}{c|c} S_1 & \\ \hline & S_2 \\ \hline 0 & 0 \\ \hline\hline BV_1 & BV_2 \end{array}\right].$$

Now, if the principal Smith form of StackMatrix(S_2, BV_2) is I_k, then there exists a sequence of unimodular row and column transformations such that

$$\left[\begin{array}{c|c} S_1 & \\ \hline & S_2 \\ \hline BV_1 & BV_2 \end{array}\right] \rightarrow \left[\begin{array}{c|c} S_1 & \\ \hline * & I \\ \hline * & \end{array}\right] \rightarrow \left[\begin{array}{c|c} I & \\ \hline & S_1 \\ \hline & * \end{array}\right].$$

The result will follow, since diagonal entries in the principal Smith form of StackMatrix$(S_1, *)$ necessarily divide the corresponding entries of S_1. A sufficient condition for StackMatrix(S_2, BV_2) to have principal Smith form I is that, for every prime p that divides s_m, BV_2 mod p has full column rank over the field $\mathbb{Z}/(p)$.

Let $N \in \mathbb{Z}^{(m-k)\times m}$ be a left kernel of V_2 over $\mathbb{Z}$. Recall the definition of a left kernel: N has full row rank, NV_2 is the zero matrix, and the Hermite column basis of N is I. Since V_2 is a subset of columns of a unimodular matrix, V_2 has Hermite row basis I and it follows that for any prime p the rows of N mod p comprise a left nullspace for V_2 mod p over the field $\mathbb{Z}/(p)$. It is easy to show (see for example [31, Lemma 15]) that $BV_2 \in \mathbb{Z}^{(k+e)\times k}$ has full column rank over $\mathbb{Z}/(p)$ if and only if StackMatrix$(N, B) \in \mathbb{Z}^{(m+e)\times m}$ has rank m over $\mathbb{Z}/(p)$. Because N is a left kernel, there exists an $(m-k)\times(m-k)$ submatrix of N that has determinant relatively prime to p. Without loss of generality (up to a column permutation of StackMatrix(N, B)) assume that the principal $(m-k) \times (m-k)$ submatrix of N has determinant relatively

prime to p. Decompose $N = [N_1 \,|\, N_2]$ and $B = [B_1, \,|\, B_2]$ where N_1 and B_1 have $m - k$ columns. Then the Schur complement of

$$\left[\begin{array}{c|c} N_1 & N_2 \\ \hline B_1 & B_2 \end{array}\right]$$

with respect to N_1 is $B_2 - B_1 N_1^{-1} N_2$. Now, let elements of B_1 be chosen, and recall that elements in B_2 are chosen independently. Our goal is to bound the probability that $B_2 - B_1 N_1^{-1} N_2 \bmod p$ has rank less than k over $\mathbb{Z}/(p)$. We consider primes $p < \lambda$ and $p \geq \lambda$ separately.

Let $p \geq \lambda$. Using a similar argument [4] as in the proof of Theorem 50 we can show that the probability that $B_2 - B_1 N_1^{-1} N_2 \bmod p$ has rank less than k over $\mathbb{Z}/(p)$ is bounded by $\sum_{i=1}^{k} (1/\lambda)^{k+e-(i-1)} \leq (1/\lambda)^{e+1} \sum_{i=0}^{k-1} (1/\lambda)^i < (1/\lambda)^{e+1} \sum_{i=0}^{\infty} (1/2)^i = 2(1/\lambda)^{e+1}$. Since there are fewer than $2m$ primes $p \geq \lambda$ that divide s_m (Fact 51), the sum of this bound over all such primes is $\leq 4m(1/\lambda)^{e+1}$, which is $\leq 4(1/\lambda)^e$ using the condition $\lambda \geq m$. Thus, the probability that there exists a prime $p \geq \lambda$ such that StackMatrix(N, B) has rank less than n over $\mathbb{Z}/(p)$ is bounded by $4(1/2)^e$.

Now consider primes $p < \lambda$. Similar to the above, we get that the probability that there exists a prime $p < \lambda$ such that StackMatrix(N, B) has rank less than k over $\mathbb{Z}/(p)$ is bounded by

$$\begin{aligned}
\sum_{p<\lambda} \sum_{i=1}^{k} \left(\frac{1 + p/\lambda}{p}\right)^{k+e-(i-1)} &< \sum_{p<\lambda} \left(\left(\frac{1 + p/\lambda}{p}\right)^{e+1} \sum_{i=0}^{\infty} \left(\frac{1 + p/\lambda}{p}\right)^{i} \right) \\
&\leq \sum_{p<\lambda} \left(\left(\frac{1 + p/\lambda}{p}\right)^{e+1} \sum_{i=0}^{\infty} (2/3)^i \right) \\
&< 3 \sum_{p<\lambda} \left(\frac{1 + p/\lambda}{p}\right)^{e+1} \\
&< 3(2/3)^{e-1} \sum_{i=2}^{\infty} (2/i)^2 \\
&< 9(2/3)^{e-1}
\end{aligned}$$

Summing the error bound for primes $\geq \lambda$ with the bound for primes $< \lambda$ gives $4(1/2)^e + 9(2/3)^{e-1}$. Finally, if $e \geq 10 + 2 \log k$ then

[4] By summing, for $i = 1 \ldots k$, a bound on the probability that column i of B_2 is chosen such that column i of $B_2 - B_1 N_1^{-1} N_2$ is a linear combination over $\mathbb{Z}/(p)$ of the previous $i - 1$ columns.

$$\begin{aligned}4(1/2)^e + 9(2/3)^{e-1} &= 4(1/2)^{10}(1/k^2) + 9(2/3)^9(1/k^{2\log 3/2}) \\ &\leq (4(1/2)^{10} + 9(2/3)^9)(1/k) \\ &< 0.239(1/k) \\ &< 1/(4k)\end{aligned}$$

The result follows. □

For the remainder of this section let $A \in \mathbb{Z}^{n\times n}$ be nonsingular with Smith form $\mathrm{diag}(s_1, s_2, \ldots, s_n)$. As in [13], we are going to exploit the structure of the Smith form. The next lemma follows from the fact that $s_1 s_2 \cdots s_n = |\det A|$ and $s_i \mid s_{i+1}$ for $1 \leq i \leq n-1$.

Lemma 53 *If λ is such that $|\det A| \leq \lambda^{2n}$, then then $s_n \leq \lambda^{2n}$ and $s_{n-2^i+1} \leq (\lambda^{2n})^{1/2^i}$ for $i = 1, 2, \ldots, \lfloor \log n \rfloor$.*

We will need one additional fact.

Fact 54 *Let s be the largest invariant factor of a nonsingular matrix A. Then sI is a multiple of the Hermite row basis of A.*

Assume that $n = 2^{t+1} - 1$ for some integer t, and let $A \in \mathbb{Z}^{n\times n}$ be nonsingular with Smith form $\mathrm{Diag}(s_1, s_2, \ldots, s_n)$. For $i = 0, 1, \ldots, t$, choose entries in $B_i \in \mathbb{Z}^{(2^i+10+2i)\times n}$ uniformly and randomly from $\{0, 1, 2, \ldots, \lambda - 1\}$, where $\lambda = \max(\|A\|, n, 2)$. Then $|\det A| \leq \lambda^{2n}$ and Lemma 53 applies. Let the Hermite row basis of

$$C := \left[\begin{array}{c|ccc} A & & & \\ \hline B_t & I & & \\ \vdots & & \ddots & \\ B_1 & & & I \\ B_0 & & & \end{array}\right] \quad \text{be} \quad \left[\begin{array}{c|cccc} H_{t+1} & * & \cdots & * & * \\ \hline & H_t & \cdots & * & * \\ & & \ddots & & \vdots \\ & & & H_1 & * \\ & & & & H_0 \end{array}\right]. \tag{15}$$

Let us identify A_0 with A. Following the approach of [41, Lemma 6.2], we can transform C to a triangular row basis via a sequence of $t+1$ unimodular

transformations, of which the first two look like:

$$\left[\begin{array}{c|ccccc} A_0 & & & & & \\ \hline B_t & I & & & & \\ \vdots & & \ddots & & & \\ B_2 & & & I & & \\ B_1 & & & & I & \\ B_0 & & & & & I \end{array}\right] \rightarrow \left[\begin{array}{c|ccccc} A_1 & & & & & * \\ \hline B_t & I & & & & \\ \vdots & & \ddots & & & \\ B_2 & & & I & & \\ B_1 & & & & I & \\ & & & & & H_0 \end{array}\right] \rightarrow \left[\begin{array}{c|ccccc} A_2 & & & & * & * \\ \hline B_t & I & & & & \\ \vdots & & \ddots & & & \\ B_2 & & & I & & \\ & & & & H_1 & * \\ & & & & & H_0 \end{array}\right]$$

Note that the Hermite row basis of

$$\left[\begin{array}{c|c} A_i & \\ \hline B_i & I \end{array}\right] \quad \text{is} \quad \left[\begin{array}{c|c} A_{i+1} & * \\ \hline & H_i \end{array}\right].$$

Fact 54 states that the largest invariant factor of A_i will be a multiple of H_i.

An application of Lemma 52 with $k = 2^0$ gives that $\mathrm{Diag}(1, s_1, s_2, \ldots, s_{n-1})$ is a multiple of the principal Smith form of StackMatrix(A_0, B_0), and thus s_{n-1} is a multiple of the largest invariant factor of StackMatrix(A_0, B_0) with probability at least $3/4$. The next theorem follows by applying Lemma 52 for $k = 2^0, 2^1, \ldots, 2^t$ with StackMatrix$(A_{\log k}, B_{\log k})$. The key observation is that StackMatrix$(A_{i+1}, 0)$ has the same Smith form as StackMatrix(A_i, B_i) since these matrices are left equivalent. Summing the probabilities of failure of the preconditioning gives $\sum_{i=0}^{t}(1/4)2^{-i} < (1/4)\sum_{i=0}^{\infty} 2^{-i} \leq 1/2$. Note that $\sum_{l=0}^{i} 2^l = 2^{i+1} - 1$. For convenience define $s_0 := 1$.

Theorem 55 *If each B_i has entries chosen uniformly and randomly from $\{0, 1, \ldots, \lambda - 1\}$ where $\lambda = \max(\|A\|, n, 2)$, then $s_{n-2^{i+1}+1}$ is a multiple of the largest invariant of* StackMatrix(A_i, B_i) *for all $i = 0, 1, \ldots, t$ simultaneously, with probability at least $1/2$.*

13 Certified computation of the determinant

Our algorithm for the determinant will call upon almost all of the algorithms presented in the previous sections. In addition, we need to extend the results of [43, Sections 12–15] for matrices over $k[x]$ to the case of integer matrices.

Smith of trailing Hermite basis

Let $A \in \mathbb{Z}^{n \times n}$ be nonsingular. For some m, $1 \leq m \leq n$, decompose the Hermite row basis of A as

$$H = \left[\begin{array}{c|c} H_1 & * \\ \hline & H_2 \end{array}\right], \tag{16}$$

where H_2 has dimension $m \times m$. If we are given an $s \in \mathbb{Z}_{\geq 1}$ such that sH_2^{-1} is integral, then following [43, Algorithm 9] we can compute the Smith form S_2 of H_2 as follows. Let B be the last m rows of I_n and compute a right integrality certificate C for BA^{-1} with respect to sI_m. Compute $D := \text{PrincipalSmith}([C \mid sI_m])$ and set $S_2 := \text{Smith}((sI_m)D^{-1})$.

The cost of computing C and D depends on m and the bitlength of s. We are going to assume that these parameters are balanced: $m \times (\log s)/n = O(\log n + \log \|A\|)$ or, equivalently, that $\log s = O((n/m) \times (\log n + \log \|A\|))$. Then Theorem 49 bounds the expected cost of computing C by $O((\log n)\mathsf{MM}(n)\mathsf{B}(\log n + \log \|A\|))$ bit operations. The Smith form D can be computed by working over the ring $\mathbb{Z}/(s)$ with $O((n/m)\mathsf{MM}(m))$ ring operations ([41, Chapter 7]). The cost of a single operation in $\mathbb{Z}/(s)$ is bounded by $\mathsf{B}(\log s)$ bit operations. Using $\mathsf{B}(ab) = O(\mathsf{B}(a)\mathsf{B}(b))$ and our assumption on $\log s$ gives that $\mathsf{B}(\log s) = O(\mathsf{B}(n/m)\mathsf{B}(\log n + \log \|A\|))$.

Theorem 56 (`SmithOfTrailingHermite`(A, m, s)) *There exists a Las Vegas algorithm that takes as input*

- $A \in \mathbb{Z}^{n \times n}$, *nonsingular,*
- $m \in \mathbb{Z}$, $1 \leq m \leq n$, *and*
- $s \in \mathbb{Z}$, *nonzero,*

and returns as output

- *false, if sI_m is not a multiple of the trailing $m \times m$ submatrix H_2 of the Hermite row basis of A, see (16), otherwise,*
- *the Smith form S_2 of H_2.*

If $m \times (\log s)/n$ is $O(\log n + \log \|A\|)$ then the algorithm uses an expected number of $O((\log n)\mathsf{MM}(n)\mathsf{B}(\log \|A\| + \log n))$ bit operations. This result assumes that $\mathsf{MM}(a)\mathsf{B}(b) = O(\mathsf{MM}(ab)/b)$.

Let $A \in \mathbb{Z}^{n\times n}$ be nonsingular with Hermite row basis

$$\left[\begin{array}{cccc|c} h_1 & h_{1,2} & \cdots & h_{1,n-1} & h_{1,n} \\ & h_2 & \cdots & h_{2,n-1} & h_{2,n} \\ & & \ddots & \vdots & \vdots \\ & & & h_{n-1} & h_{n-1,n} \\ \hline & & & & h_n \end{array}\right]. \tag{17}$$

In [43, Section 15] we described algorithm `DetReduction`(A) that computes a new matrix $B \in \mathbb{Z}^{n\times n}$ that has Hermite row basis

$$\left[\begin{array}{cccc|c} h_1 & h_{1,2} & \cdots & h_{1,n-1} & \\ & h_2 & \cdots & h_{2,n-1} & \\ & & \ddots & \vdots & \\ & & & h_{n-1} & \\ \hline & & & & 1 \end{array}\right]. \tag{18}$$

As a side effect the algorithm also computes the trailing diagonal entry h_n. The algorithm described in [43] was for the case $k[x]$ but extends directly to the case $\mathbb{Z}$, see the worked example given in [43, Section 15]. The algorithm computes solutions to two nonsingular rational systems involving A, plus does some additional work like an extended gcd computation that can be accomplished with $O(n\,\mathsf{B}(n(\log\|A\| + \log n)))$ bit operations. The matrix B produced will be identical to A except for possibly the last column. An inspection of the algorithm reveals that entries in the last column will be bounded in magnitude by $n^2\|A\|$.

Let $P \in \mathbb{Z}^{n\times n}$ be the permutation matrix that rotates the columns to the right by one (that is, such that column $(j \bmod n) + 1$ of BP is equal to column j of B). Now, if B has Hermite row basis as in (18), then BP will have Hermite row basis

$$\left[\begin{array}{c|ccc} 1 & & & \\ \hline & h_1 & h_{1,2} & \cdots & h_{1,n-1} \\ & & h_2 & \cdots & h_{2,n-1} \\ & & & \ddots & \vdots \\ & & & & h_{n-1} \end{array}\right]. \tag{19}$$

For a given k, $1 \leq k \leq n$, decompose A and the Hermite row basis of A as

$$A = \left[\begin{array}{c|c} A_{11} & A_{12} \\ \hline A_{21} & A_{22} \end{array}\right] \text{ and } H = \left[\begin{array}{c|c} H_{11} & H_{12} \\ \hline & H_{22} \end{array}\right] \tag{20}$$

where A_{22} and H_{22} are $k \times k$, and consider executing the following code fragment, where P is the permutation matrix described above.

$B := A$;
for i **from** 1 **to** k **do**
  $B :=$ `DetReduction`(B);
  $B := BP$
od

On output

$$B = \left[\begin{array}{c|c} * & A_{11} \\ \hline * & A_{21} \end{array}\right] \text{ with Hermite basis } \left[\begin{array}{c|c} I & \\ \hline & H_{11} \end{array}\right]. \tag{21}$$

This gives the following result.

Theorem 57 (`IteratedDetRedution`(A, k)) *There exists a Las Vegas algorithm that takes as input*

- $A \in \mathbb{Z}^{n \times n}$, *nonsingular, and*
- $k \in \mathbb{Z}$, $1 \leq k \leq n$,

and returns as output

- *the determinant of H_{22}, where H_{22} is the trailing $k \times k$ submatrix of the Hermite row basis of A, and*
- *a $B \in \mathbb{Z}^{n \times n}$ with $\|B\| \leq n^{2k}\|A\|$, and such that the Hermite row basis of B has the shape shown in (21), where the Hermite row basis of A is as in (20).*

The algorithm uses an expected number of $O(k(\log n)\mathsf{MM}(n)\mathsf{B}(\log \|A\| + k \log n))$ bit operations. This result assumes $\mathsf{B}(t) = O(\mathsf{MM}(t)/t)$.

The bound for $\|B\|$ in Theorem 57 is very pessimistic. In our application of algorithm `IteratedDetReduction` the parameter k will be $O(1)$, so the bound suffices for our purposes.

Extension to nonsingular matrix

Let $A \in \mathbb{Z}^{n \times m}$ have full column rank. Our goal is to construct a nonsingular matrix $B \in \mathbb{Z}^{(n+5) \times (n+5)}$ that has Hermite row basis equal to $\mathrm{Diag}(I, H)$,

where H is the Hermite row basis of A. If $n = m$ we may simply choose $B := \texttt{Diag}(I_5, A)$, so assume that $n > m$. We consider two cases: $n \leq m + 15$ and $n > m + 15$.

Suppose $n \leq m + 15$. Use the method supporting Corollary 36 to find a set $\{i_1, i_2, \ldots, i_m\}$ of row indices such that these rows of A are linearly independent. Let $\bar{A}$ be equal to A augmented with columns $\{1, 2, \ldots, n\} - \{i_1, i_2, \ldots, i_m\}$ of I_n. Then $\bar{A}$ is nonsingular, and

$$B := \texttt{Diag}(I_5, \texttt{IteratedDetReduction}(\bar{A}, n - m))$$

will be as desired.

Now suppose $n > m + 15$. Then extend A to a nearly square matrix (fifteen fewer columns than rows)

$$E := \left[\begin{array}{c|c} A & * \\ \hline & * \end{array}\right] \in \mathbb{Z}^{(n+5)\times(n-10)}$$

where entries in the blocks labelled $*$ are chosen uniformly and randomly from $\{0, 1, \ldots, \lambda - 1\}$, where $\lambda := \max(n, \|A\|)$. Then the Hermite row basis of E is equal to $\text{Diag}(H, I)$ with probability at least $4/5$ (Theorem 50). Check that E has full column rank using the method of Theorem 40 and if not choose a different E. Assume henceforth that E has full column rank. Extend E to a nonsingular matrix $\bar{E} \in \mathbb{Z}^{(n+5)\times(n+5)}$ by augmenting with a subset of fifteen columns of I_{n+5}, as described above. Let $B := \texttt{IteratedDetReduction}(\bar{E}, 15)$. Finally, try computing a right integrality certificate for the last $n - m - 10$ rows of B^{-1} with respect to I_{n-m-10} to check that the trailing $(n - m - 10) \times (n - m - 10)$ submatrix of the Hermite row basis of B is equal to I_{n-m-10}. If this check returns false then repeat the construction of E and try again. Otherwise, rotate the columns of B to the right by $n - m - 10$.

Theorem 58 ($\texttt{ExtensionToNonsingular}(A)$) *There exists a Las Vegas algorithm that takes as input*

- $A \in \mathbb{Z}^{n\times m}$ *with rank* m*,*

and returns as output

- *a nonsingular* $B \in \mathbb{Z}^{(n+5)\times(n+5)}$ *with* $\|B\| \leq n^{30} \max(n, \|A\|)$*, and such that the Hermite row basis of* B *has the shape*

$$\left[\begin{array}{c|c} I & \\ \hline & H \end{array}\right],$$

  where H *is the Hermite row basis of* A*.*

The algorithm uses an expected number of $O(\mathsf{MM}(n)(\log n)\mathsf{B}(\log \|A\| + \log n))$ *bit operations. This result assumes* $\mathsf{B}(t) = O(\mathsf{MM}(t)/t)$.

The determinant algorithm

Let $A \in \mathbb{Z}^{n\times n}$. Assume, at the cost of assaying if A is singular using the method of Theorem 40, that $\det A \neq 0$. Assume, up to embedding A into a larger matrix $\mathrm{Diag}(I, A)$, that $n = 2^{t+1} - 1$ for some integer t.

For $i = 0, 1, \ldots, t$, choose entries in $B_i \in \mathbb{Z}^{(2^i+10+2i)\times n}$ uniformly and randomly from $\{0, 1, 2, \ldots, \lambda - 1\}$, where $\lambda := \max(\|A\|, n, 2)$. Let $m := \sum_{i=0}^{t}(2^i + 10 + 2i) = n + O((\log n)^2)$, and let C be the $(n + m) \times (n + m)$ matrix shown in (15) on page 39. Then $|\det C| = |\det A| \leq \lambda^{2n}$. We will compute $\det H_i$ (see (15)) for $i = 0, 1, 2, \ldots, t$ in succession, terminating early and reporting fail if an $i \in \{0, 1, \ldots, t\}$ is found such that the trailing entry in $\mathtt{Smith}(H_i)$ has magnitude larger than $(\lambda^{2n})^{1/2^i}$. By Theorem 55, failure will be reported with probability less than $1/2$.

Let $C[1 \ldots n + m, 1 \ldots k]$ denote the submatrix of C comprised of the first m columns. Initialize $k := n + m$ and $d := 1$ and execute the following:

for i **from** 0 **to** t **do**
  repeat
    $R := \mathtt{ExtensionToNonsingular}(C[1 \ldots n + m, 1 \ldots k])$;
    $s := \mathtt{LargestInvariantFactor}(R)$;
    if $s > (\lambda^{2n})^{1/2^i}$ then **return** fail **fi**;
    $S := \mathtt{SmithOfTrailingHermite}(R, 2^i + 10 + 2i, s)$
  until $S \neq$ fail;
  $d := d \times \det(S)$;
  $k := k - (2^i + 10 + 2i)$
od;

If fail is returned, then construct a new C and try again. If the loop completes, then R has Hermite row basis $\mathrm{Diag}(I, H_{t+1})$, see (15). Check that $\mathtt{UniCert}(R)$ returns true to ensure that $H_{t+1} = I$. If not, construct a new C and try again. If $\mathtt{UniCert}(R)$ does return true then $d = |\det A|$. Compute the determinant of A modulo a small prime in order to determine if d needs to be negated.

Each of the $O(\log n)$ calls to algorithm `SmithOfTrailingHermite` has $(2^i + 10 + 2i) \times (\log s)/n = O(\log \|A\| + \log n)$.

Theorem 59 *Let* $A \in \mathbb{Z}^{n\times n}$. *There exists a Las Vegas algorithm that computes the determinant of* A *using an expected number of* $O((\log n)^2\, \mathsf{MM}(n)\mathsf{B}(\log \|A\| +$

$\log n))$ *bit operations. This result assumes* $\mathsf{MM}(a)\mathsf{B}(b) = O(\mathsf{MM}(ab)/b)$.

14 Conclusions

Consider the following problems on an input matrix $A \in \mathbb{Z}^{n\times n}$ and, in the case of problem LinSys, a $b \in \mathbb{Z}^{n\times 1}$ with $\|b\| = \|A\|^{O(n)}$.

Problem	Compute the
LinSys	vector $A^{-1}b$ for A nonsingular
Det	determinant
MinPoly	minimal polynomial
CharPoly	characteristic polynomial
Frobenius	Frobenius canonical form
Smith	Smith canonical form

Given an algorithm that can multiply two $n \times n$ matrices in $O(n^\omega)$ scalar operations, we have given a Las Vegas algorithms for problem LinSys (Theorem 37) and for problem Det (Theorem 59) that use an expected number of $O^\sim(n^\omega \log \|A\|)$ bit operations.

To the best of our knowledge, algorithms with cost $O^\sim(n^\omega \log \|A\|)$ bit operations are not yet known for the problems MinPoly, CharPoly, Frobenius and Smith. The currently best know cost estimate for MinPoly is $O^\sim(n^{2.697263} \log \|A\|)$ bit operations (Monte Carlo), given in [27]. This result assumes the best known value for ω (= 2.375477 [9]) and uses also the fast rectangular matrix multiplication techniques in [8]. Without using fast matrix techniques the cost estimate derived in [27] for MinPoly is $O^\sim(n^{3.2} \log \|A\|)$ bit operations. This result for MinPoly extends to the other problems using some known reductions. We recall these now.

For two problems P_1 and P_2, let us write $P_1 \leq P_2$ if a Monte Carlo algorithm for P_2 with running time $O^\sim(n^\eta \log \|A\|)$ bit operations ($\eta \geq \omega$) gives us a Monte Carlo algorithm for P_1 with running time $O^\sim(n^\eta \log \|A\|)$ bit operations. Then MinPoly = Frobenius = CharPoly. The reduction of Frobenius to either CharPoly or MinPoly uses the observation that the entire Frobenius form F over $\mathbb{Z}$ can be reconstructed from either the minimal or characteristic polynomial together a single image (a signature) of F mod p over $\mathbb{Z}/(p)$ for well chosen prime p, the image being computed in $O^\sim(n^\omega \log\log \|A\|)$ bit operations using any of the algorithms in [12,18,41]. A reduction Smith $\leq$

MinPoly is given in [19]. The reductions Smith $\leq$ MinPoly and Frobenius $\leq$ MinPoly are used in [27], see [27, Section 7] for more details.

Our algorithms for LinSys and Det are based on the high-order lifting and integrality certifications techniques of [43], developed for polynomial matrices. The shifted number system allowed us to extend these techniques to the integer case. Actually, the low level algorithms in the current paper are based also on a new idea: the sparse inverse expansion. On the one hand, the algorithm for computing $A^{-1}b$ in the polynomial case [43, Section 9] precomputed $O(\log n)$ high-order segments of the expansion of the inverse. Precomputation was required because the segments were used in the reverse order of their computation. On the other hand, the sparse inverse expansion introduced in Section 8 applies the segments in the order of their computation, thus allowing them to be computed on the fly. We may derive that the intermediate space requirement of our algorithms for LinSys and Det is bounded by $O(n^2(\log \|A\| + \log n))$ bits, which is a factor of at most $O(\log n)$ more than the space required to write down the input matrix. We remark that the sparse inverse expansion is applicable in the polynomial case also and will reduce the intermediate space requirement of the linear solving algorithm in [43, Section 9] by a factor of $O(\log n)$, or down to $O(n^2 \deg A)$ field elements.

For a nonsingular $A \in \mathbb{Z}^{n\times n}$ our algorithm for Det in Section 13 can probably be adapted to get a Las Vegas algorithm for SmithForm. If A is singular, however, a difficult arises in that a Las Vegas Smith form algorithm must necessarily certify the rank of A (since this is one of the invariants revealed by the form). We currently don't know how to extend our techniques to certify the rank (for example, when the rank is about $n/2$). To the best of our knowledge, the current best cost estimate for rank certification is $O^{\sim}(n^{2.697263} \log \|A\|)$ bit operations, also obtained using the MinPoly algorithm of [27] and the reduction of rank certification to MinPoly given in [37].

In the future we will report on implementations the high-order lifting technique described in this paper. In [5,6] the Integer Matrix Library (IML) is described. On a modern processor [5], IML computes the exact solution of a nonsingular system of dimension 500, 2000 and 10,000 (with single decimal digit entries) in about one second, one minute and one hour, respectively. The performance of IML is achieved in part by using the highly optimized and portable ATLAS/BLAS software library for numerical linear algebra [45]. Large integer arithmetic is performed using GMP [20]. Currently, IML's nonsingular solver is based on the classic linear lifting algorithm as described in [11,30]. IML also includes an implementation of a new algorithm for finding minimal denominator solutions to integer input systems of arbitrary shape, rank profile and entry size. In the future we plan to add functionality for unimodularity

[5] Intel Itanium 2 @ 1.3GHz

certification and determinant computation based on the techniques described in this paper.

Acknowledgements

Corrections and comments from the two anonymous referees have substantially improved the presentation.

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] A. V. Aho, J. E. Hopcroft, and J. D. Ullman. *The Design and Analysis of Computer Algorithms*. Addison-Wesley, 1974.

[3] H. Brönnimann and M. Yvinec. Efficient exact evaluation of signs of determinants. *Algorithmica*, 21:21–56, 2000.

[4] P. Bürgisser, M. Clausen, and M. A. Shokrollahi. *Algebraic Complexity Theory*, volume 315 of *Grundlehren der mathematischen Wissenschaften*. Springer-Verlag, 1996.

[5] Z. Chen. A BLAS based C library for exact linear algebra on integer matrices. Master's thesis, School of Computer Science, University of Waterloo, 2005.

[6] Z. Chen and A. Storjohann. A BLAS based C library for exact linear algebra on integer matrices. In M. Kauers, editor, *Proc. Int'l. Symp. on Symbolic and Algebraic Computation: ISSAC '05*. ACM Press, New York, 2005.

[7] K. L. Clarkson. Safe and efficient determinant evaluation. In *Proc. 33rd Annual Symp. Foundations of Comp. Sci.*, pages 387–395, Los Alamitos, California, 1992. IEEE Computer Society Press.

[8] D. Coppersmith. Rectangular matrix multiplication revisited. *J. Complexity*, 13:42–49, 1997.

[9] D. Coppersmith and S. Winograd. Matrix multiplication via arithmetic progressions. *Journal of Symbolic Computation*, 9:251–280, 1990.

[10] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein. *Introduction to Algorithms*. MIT Press and McGraw-Hill, 2 edition, 2001.

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

[12] W. Eberly. Asymptotically efficient algorithms for the Frobenius form. Technical report, Department of Computer Science, University of Calgary, 2000.

[13] 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.

[14] I. Z. Emiris and V. Y. Pan. Improved algorithms for computing determinants and resultants. *Journal of Complexity*, 2004. To appear.

[15] J. von zur Gathen and J. Gerhard. *Modern Computer Algebra*. Cambridge University Press, 2 edition, 2003.

[16] K. O. Geddes, S. R. Czapor, and G. Labahn. *Algorithms for Computer Algebra*. Kluwer, Boston, MA, 1992.

[17] M. Giesbrecht. *Nearly Optimal Algorithms for Canonical Matrix Forms*. PhD thesis, University of Toronto, 1993.

[18] M. Giesbrecht. Nearly optimal algorithms for canonical matrix forms. *SIAM Journal of Computing*, 24:948–969, 1995.

[19] M. Giesbrecht. Fast computation of the Smith form of a sparse integer matrix. *Computational Complexity*, 10(1):41–69, 11 2001.

[20] T. Granlund. The GNU multiple precision arithmetic library, 2004. Edition 4.1.4. `http://www.swox.com/gmp`.

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

[22] O. Ibarra, S. Moran, and R. Hui. A generalization of the fast LUP matrix decomposition algorithm and applications. *Journal of Algorithms*, 3:45–56, 1982.

[23] 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.

[24] E. Kaltofen. An output-sensitive variant of the baby steps/giant steps determinant algorithm. In T. Mora, editor, *Proc. Int'l. Symp. on Symbolic and Algebraic Computation: ISSAC '02*, pages 138–144. ACM Press, New York, 2002.

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

[26] E. Kaltofen and G. Villard. Computing the sign or the value of the determinant of an integer matrix, a complexity survey. *J. Computational Applied Math.*, 162(1):133–146, Jan. 2004. Special issue: Proceedings of the International Conference on Linear Algebra and Arithmetic 2001, held in Rabat, Morocco, 28–31 May 2001, S. El Hajji, N. Revol, P. Van Dooren (guest eds.).

[27] E. Kaltofen and G. Villard. On the complexity of computing determinants. *Computational Complexity*, 13(3–4):91–130, 2004.

[28] D. E. Knuth. *The Art of Computer Programming, Vol.2, Seminumerical Algorithms*. Addison–Wesley (Reading MA), 2 edition, 1981.

[29] W. Krandick and J. Johnson. Efficient multiprecision floating point multiplication with optimal directional rounding. In *11th IEEE Symposium on Computer Arithmetic*, pages 228–233. IEEE Computer Society Press, 1993.

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

[31] T. Mulders and A. Storjohann. Certified dense linear system solving. *Journal of Symbolic Computation*, 37(4):485–510, 2004.

[32] M. Newman. *Integral Matrices*. Academic Press, 1972.

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

[34] V. Y. Pan. Randomized acceleration of fundamental matrix computations. In *Proc. STACS 2002*, volume 2285, pages 215–226, Heidelberg, Germany, 2002. Springer-Verlag.

[35] V. Y. Pan and X. Wang. On rational number reconstruction and approximation. *SIAM Journal of Computing*, 33(2):502–503, 2004.

[36] J. B. Rosser and L. Schoenfeld. Approximate formulas for some functions of prime numbers. *Ill. J. Math.*, 6:64—94, 1962.

[37] D. Saunders, A. Storjohann, and G. Villard. Matrix rank certification. *Electronic Journal of Linear Algebra*, 11:16–23, 2004.

[38] A. Schönhage. Schnelle Berechnung von Kettenbruchentwicklungen. *Acta Informatica*, 1:139–144, 1971.

[39] A. Schönhage and V. Strassen. Schnelle Multiplikation grosser Zahlen. *Computing*, 7:281–292, 1971.

[40] A. Storjohann. Near optimal algorithms for computing Smith normal forms of integer matrices. In Y. N. Lakshman, editor, *Proc. Int'l. Symp. on Symbolic and Algebraic Computation: ISSAC '96*, pages 267–274. ACM Press, New York, 1996.

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

[42] A. Storjohann. High–order lifting. Extended Abstract. In T. Mora, editor, *Proc. Int'l. Symp. on Symbolic and Algebraic Computation: ISSAC '02*, pages 246–254. ACM Press, New York, 2002.

[43] A. Storjohann. High–order lifting and integrality certification. *Journal of Symbolic Computation*, 36(3–4):613–648, 2003. Extended abstract in [42].

[44] V. Strassen. Gaussian elimination is not optimal. *Numerische Mathematik*, 13:354–356, 1969.

[45] R. C. Whaley, A. Petitet, and J. J. Dongarra. Automated empirical optimization of software and the ATLAS project. *Parallel Computing*, 27(1–2), 2001.