Integer Matrix Rank Certification

Arne Storjohann
astorjoh@uwaterloo.ca
David R. Cheriton School of Computer Science
University of Waterloo, Ontario, Canada N2L 3G1

ABSTRACT

Let $M = \left[\begin{smallmatrix} A & B \\ C & D \end{smallmatrix}\right]$ be a $2n \times 2n$ integer matrix with the principal block A square and nonsingular. An algorithm is presented to determine if the Schur complement $D - CA^{-1}B$ is equal to the zero matrix in $O\tilde{}(n^\omega \log ||M||)$ bit operations. Here, ω is the exponent of matrix multiplication and $||M||$ denotes the largest entry in absolute value. The algorithm is randomized of the Las Vegas type, and either returns the correct answer ("yes" or "no"), or returns fail with probability less than $1/2$. This gives a Las Vegas algorithm for computing the rank r of an $n \times m$ integer matrix A in an expected number of $O\tilde{}(nmr^{\omega-2} \log ||A||)$ bit operations.

Categories and Subject Descriptors: F.2 [Theory of computation]: Analysis of algorithms and problem Complexity; I.1 [Computing Methodology]: Symbolic and algebraic manipulation: Algorithms

General Terms: Algorithms

1. INTRODUCTION

Consider a $2n \times 2n$ integer matrix

$$M = \left[\begin{array}{c|c} A & B \\ \hline C & D \end{array}\right], \tag{1}$$

where all the blocks are square of dimension n and A is nonsingular. The rank of M is equal to n if and only if the Schur complement $D - CA^{-1}B$ is equal to the zero matrix.

$$\left[\begin{array}{c|c} I_n & \\ \hline -CA^{-1} & I_n \end{array}\right] \left[\begin{array}{c|c} A & B \\ \hline C & D \end{array}\right] = \left[\begin{array}{c|c} A & B \\ \hline & D - CA^{-1}B \end{array}\right]$$

Entries of $(D - CA^{-1}B) \det A$ are minors of M of dimension $n+1$. The definition of the determinant gives the bound $||(D - CA^{-1}B) \det A|| \leq (n+1)! \, ||M||^{n+1} \leq ((n+1)||M||)^{n+1}$, where $||\cdot||$ denotes the maximum entry in absolute value. So, the numbers in $D - CA^{-1}B$ can have size (bitlength) more than n times that of numbers in M. The proof of [6, Proposition 2.3] details a deterministic algorithm with cost $O\tilde{}(n \times n^3 \log ||M||)$ bit operations to certify that $D - CA^{-1}B$ is the zero matrix. This can be reduced to $O\tilde{}(n \times n^\omega \log ||M||)$ where ω is the exponent of matrix multiplication. The "$n\times$" in these complexity results arise because of the aforementioned bound for the size of entries in $D - CA^{-1}B$.

A well known, and much faster probabilistic approach to test if $D = CA^{-1}B$ is to choose a single random prime p and test if $D \equiv CA^{-1}B \bmod p$. Primes are sufficiently dense that it suffices to have $\log p \in O(\log n + \operatorname{loglog} ||M||)$ to ensure a high probability of success (see, for example, [13, Section 2.1]). This gives a Monte Carlo algorithm with cost $O\tilde{}(n^\omega \operatorname{loglog} ||M|| + n^2 \log ||M||)$ bit operations. If the algorithm reports ("no,", $D \neq CA^{-1}B$) this is guaranteed to be correct. But if the algorithm reports ("yes," $D = CA^{-1}B$) this might be incorrect. The probability of returning an incorrect answer can be made arbitrarily small by repeating the algorithm, or, as is more usual, by choosing the prime p from a larger set.

This paper is concerned with the problem of computing the rank in a certified fashion. We give a Las Vegas algorithm that has cost $O\tilde{}(n^\omega \log ||M||)$ bit operations to determine if $\operatorname{rank}(M) = n$ or $\operatorname{rank}(M) > n$. This matches, up to logarithmic factors, the cost of multiplying together two integer matrices with the same dimension and same size of entries as M. Using the best known exponent for ω this becomes $O\tilde{}(n^{2.376} \log ||M||)$ bit operations. To the best of our knowledge, the previously fastest Las Vegas algorithm for this problem has cost $O\tilde{}(n^{2.697263} \log ||M||)$ bit operations, obtained by combining the minimal polynomial algorithm of [9] with the techniques in [12].

Our interest in the rank certification problem is motivated by the recent progress made in understanding the bit complexity of linear algebra problems on integer matrices. We refer to [9, 15] for surveys. In particular, the determinant of an $n \times n$ nonsingular matrix A, and the solution of a linear system of equations $Ax = b$, can be computed in a Las Vegas fashion with $O\tilde{}(n^\omega \log ||A||)$ bit operations [15]. A further example is certification of linear system inconsistency in case A has unknown rank. The school method to certify that $Ax = b$ is inconsistent is to compare the rank of A with that of the augmented system $\left[\begin{array}{c|c} A & b \end{array}\right]$. Fast methods to certify inconsistency have been developed that specifically avoid the need to certify the rank [5]. Adapted to the setting of dense integer matrices [10], these methods lead to an $O\tilde{}(n^\omega \log ||A||)$ bit operations Las Vegas algorithm that either computes a solution to the linear system over $\mathbb{Q}$ or proves that the rank of the augmented system is one more than the rank of A, but the actual certified rank of A itself

ISSAC'09, July 28–31, 2009, Seoul, Republic of Korea.

is not computed.

Although certified rank computation can be avoided in some cases, many important problems on integer matrices, such as computing a basis for the nullspace, computing a reduced lattice basis from a generating set [11], or transforming the input matrix to a canonical form, have the property that the rank of the matrix is revealed. For example, consider the problem of computing the Smith form, a diagonal canonical form under unimodular pre- and post-multiplication that looks like $\mathrm{Diag}(s_1, s_2, \ldots, s_r, 0, \ldots, 0)$, where $1 \leq s_1|s_2|\cdots|s_r \neq 0$ (see, for example, [4, 6, 13]). Since r is the rank of the input matrix, a Las Vegas reduction of Smith form computation to integer matrix multiplication must certify correctness of r in the same time.

We can illustrate the key ideas of our approach using the following scalar example.

$$\left[\begin{array}{c|c} A & B \\ \hline C & D \end{array}\right] = \left[\begin{array}{c|c} 21 & 14 \\ \hline 6 & 4 \end{array}\right]$$

We can express 21^{-1} as an infinite 10-adic expansion as follows: $21^{-1} = 1+8\cdot10+3\cdot10^2+2\cdot10^3+\cdots$. For this example, $6 \cdot 21^{-1} \cdot 14 = 4$, so pre- and post-multiplying the infinite 10-adic expansion of 21^{-1} by 6 and 14 will annihilate the expansion except for the leftover term 4. As explained above, it will be sufficient to assay if $6 \cdot 21^{-1} \cdot 14 \equiv 4$ up to some sufficiently high order. Suppose, for the purposes of this example, that our goal is to assay if $6 \cdot 21^{-1} \cdot 14 \equiv 4 \bmod 10^{32}$. For integers a and $m > 0$, let $\mathrm{rem}(a, m)$ denote the unique integer in the range $[0, m-1]$ that is congruent to a modulo m. If we pre- and post-multiply $\mathrm{rem}(21^{-1}, 10^{32})$ by 6 and 14 we obtain

$$\begin{aligned} & 6\,\mathrm{rem}(21^{-1}, 10^{32})\,14 \\ &= 6 \cdot 80952380952380952380952380952381 \cdot 14 \\ &= 68000000000000000000000000000000004 \\ &= 4 + 68 \cdot 10^{32}. \end{aligned} \tag{2}$$

The modulo 10^{32} adjustment term $68 \cdot 10^{32}$ appears in (2) because we used the truncated expansion $\mathrm{rem}(21^{-1}, 10^{32})$.

We now show how to reduce the single problem of order 32 as described above to two problems of order only 16. Split the expansion of $\mathrm{rem}(21^{-1}, 10^{32})$ into its high and low order parts as follows: $\mathrm{rem}(21^{-1}, 10^{32}) = L + H \cdot 10^{16}$.

$$\begin{aligned} & \mathrm{rem}(21^{-1}, 10^{32}) \\ &= \overbrace{8095238095238095}^{H}\underbrace{2380}_{E}\overbrace{952380952381}^{L}. \end{aligned} \tag{3}$$

The high-order lifting techniques in [15] allow us to compute the first few leading digits E of L efficiently, without computing the entire expansion of L. Consider pre- and post-multiplying L and H separately by 6 and 14.

$$\begin{aligned} & 6\,\mathrm{rem}(21^{-1}, 10^{32})\,14 \\ &= 6 \cdot \overbrace{8095238095238095}^{H}\overbrace{2380952380952381}^{L} \cdot 14 \\ &= (6 \cdot L \cdot 14) + (6 \cdot H \cdot 10^{16} \cdot 14) \\ &= (4 + 20 \cdot 10^{16}) + (-20 \cdot 10^{16} + 68 \cdot 10^{32}) \end{aligned} \tag{4}$$

Equation (4) illustrates a necessary condition for $CA^{-1}B = D$ to hold: the modulo 10^{16} adjustment term arising from the pre- and post-multiplication of L must cancel out the leftover term arising from the pre- and post-multiplication of $H \cdot 10^{16}$. To compute the modulo 10^{16} adjustment term it will be sufficient to have only E, the first few leading bits of L, as shown in (3). In particular, note that

$$6 \cdot E \cdot 14 = 6 \cdot 2380 \cdot 14 = -80 + 20 \cdot 10^4.$$

We know that $H = \mathrm{rem}(21^{-1}R, 10^{16})$ where the so-called residue R is defined by $R := (I - AL)/10^{16} = -5$. Fortunately, R is guaranteed to have small bitlength and can also be computed efficiently using only E. If we let $\bar{B} := RB = -5 \cdot 14 = -70$ and $\bar{D} := 20$, then $CA^{-1}B \equiv D \bmod 10^{32}$ if and only if $CA^{-1}B \equiv D \bmod 10^{16}$ and $CA^{-1}\bar{B} \equiv \bar{D} \bmod 10^{16}$. Our original problem of checking if $6 \cdot 21^{-1} \cdot 14 \equiv 4 \bmod 10^{32}$ is thus equivalent to checking if $6 \cdot 21^{-1} \cdot \left[\begin{array}{c|c} 14 & -70 \end{array}\right] \equiv \left[\begin{array}{c|c} 4 & 20 \end{array}\right] \bmod 10^{16}$.

If our original problem is to check that $\mathrm{rem}(CA^{-1}B, 10^k) - D = 0$ where A, B, C and D are scalars and k is power of 2, then repeating the above order reduction process i times produces the equivalent problem of checking that

$$\mathrm{rem}(CA^{-1}B', 10^{k'}) - D' = 0, \tag{5}$$

with order $k' := k/2^i$, but where B' and D' are now row vectors of dimension 2^i. The second, and key idea of our algorithm, is to exploit the following well known property of the integers to achieve a compression of the problem dimension: if v is an integer row vector of arbitrary dimension, then v is the zero vector if and only if the scalar vv^T is zero, where v^T denotes the transpose of v. More generally, for an integer matrix M we have $\mathrm{rank}(M) = \mathrm{rank}(MM^T)$, a property that is exploited for rank certification in [12]. An equivalent problem to checking (5) is thus to check that the dot product of the vector on the left of (5) with itself is equal to zero. By using a high-order component of the expansion of A^{-1} and CA^{-1}, we show how to reduce this to three scalar problems of order k'.

We now discuss some related work for polynomials matrices. If we consider the matrix in (1) to have entries univariate polynomials from $\mathsf{K}[x]$ of degree d, K a field, the algorithm in [16] will determine in a Las Vegas fashion if $D - CA^{-1}B$ is the zero matrix in $O^{\tilde{}}(n^\omega d)$ field operations from K. The algorithm in [16] actually computes a complete left nullspace of M of total size $O^{\tilde{}}(n^2 d)$ coefficients from K, or about the same as required to write down M. For comparison, the nullspace $\left[\begin{array}{c|c} -CA^{-1}\det A & I_n \det A \end{array}\right]$ based on the Schur complement can require $\Omega(n^3 d)$ field elements to write down. The rank certification algorithm for integer matrices we present in this paper does not compute a nullspace of the matrix: we only assay if $D - CA^{-1}B$ is the zero matrix.

The rest of this paper is organized as follows. Section 2 recalls some notation and results from [15] about X-adic expansions. Section 3 gives a detailed explanation of the rank certificate algorithm, referring to the various results in Section 4–6 that prove correctness of the algorithm. A Las Vegas algorithm for solving the problem discussed at the start of this introduction is given in Section 7. Finally, Section 8 extends the algorithm to get a Las Vegas algorithm for computing the rank of an integer matrix in $O^{\tilde{}}(nmr^{\omega-2}\log||A||)$ bit operations.

Following [3, 15], we give cost estimates in terms of multiplication times $\mathsf{M}(t)$ for integers and $\mathsf{MM}(n)$ for matrices. The algorithm of [14] allows $\mathsf{M}(t) = O(t(\log t)(\log\log t))$, while the asymptotically fastest known method for matrix

multiplication [2] allows $\mathsf{MM}(n) = O(n^{2.376})$. We place no restrictions on MM: our results remain valid if $\omega = 2$.

2. PRELIMINARIES

Fix an integer radix $X > 1$ and an integer shift $0 \le t \le X-1$. Every rational number a with denominator relatively prime to X can be expressed as a unique and possibly infinite expansion

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

where each integer coefficient a_i lies in the range $[-t, X-1-t]$. Operation Trunc truncates an X-adic expansion:

$$\mathrm{Trunc}(a,k) := a_0 + a_1 X + a_2 X^2 + a_3 X^3 + \cdots + a_{k-1} X^{k-1}$$

Operation Left corresponds to a division by a power of X:

$$\mathrm{Left}(a,k) := a_k + a_{k+1} X + a_{k+2} X^2 + a_{k+3} X^3 + \cdots.$$

We will often use the decomposition $a = \mathrm{Trunc}(a,k) + \mathrm{Left}(a,k) X^k$. Rational matrices also admit (X,t)-adic expansions. If $A \in \mathbb{Z}^{n\times n}$ is nonsingular with $\det A$ relatively prime to X (Notation: $\det A \perp X$) we can write $A^{-1} = *+*X+*X^2+\cdots$ where each $*$ lies in $[-t, X-1-t]^{n\times n}$. Operation Trunc and Left extend by elementwise application. The first step of the algorithm in this paper is to compute some high-order components of A^{-1}:

$$\begin{aligned} A^{-1} &= \overbrace{*+*X}^{E^{(1)}} + \overbrace{*X^2+*X^3}^{E^{(2)}X^2} + *X^4 \\ &\quad + *X^5 + \overbrace{*X^6+*X^7}^{E^{(3)}X^6} + *X^8 + *X^9 + \cdots \end{aligned}$$

We will also require the analogous coefficients of the expansion of CA^{-1}. These high-order components can be accomplished using the high-order lifting techniques of [15].

THEOREM 1. *There exists an algorithm that takes as input a prime p with $p \perp \det A$ and $\log p$ bounded by $O(\log n + \mathrm{loglog}\,||A||)$, together with a bound β such that $||A||, ||C|| \le \beta$, and returns as output*

- *an (X,t)-adic shifted number system for which X satisfies $\log X \in O(\log n + \log||A||)$ and the property that $\mathrm{Trunc}(a,k) = a$ for all $a \in \mathbb{Z}$ with*
$$|a|/X^{k-1} \le 12 n^5 \beta^2 \qquad (6)$$
- *the high-order inverse components*
$$E^{(i)} := \mathrm{Left}(\mathrm{Trunc}(A^{-1}, 2^i), 2^i - 2)$$
and
$$R^{(i)} := \mathrm{Left}(\mathrm{Trunc}(CA^{-1}, 2^i), 2^i - 2)$$
for $1 \le i \le \log_2 k$, where k is the smallest power of two such that $X^k > ((n+1)\beta)^{n+1}$.

The cost of the algorithm is $O((\log n)\mathsf{MM}(n)\mathsf{M}(\log n + \log\beta))$ bit operations. The algorithm is randomized and either returns a correct result or reports failure, the latter with probability $< 1/2$.

We end this section by recalling some material from [15]. Let $B \in \mathbb{Z}^{n\times *}$ be an integer matrix of arbitrary column dimension. Then

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

where the residue $\mathrm{Res}(A,B,k)$ is defined by

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

Note that the division by X^k is exact. If $||B||$ is small enough, the above formula to compute the residue may be simplified.

LEMMA 2. *If $B = \mathrm{Trunc}(B,k)$ then*

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

In particular, for any $i > 0$ we have

$$\mathrm{Res}(A,I,2^i) = \mathrm{Left}(-A\,E^{(i)}, 2).$$

The next two size bounds will be indispensable for the proofs of results in the subsequent sections. Lemma 3 is [15, Corollary 7].

LEMMA 3. $||\mathrm{Left}(A\,\mathrm{Trunc}(*,k),k)|| \le n||A||$.

In the statement of the following lemma, the $*$ notation in $\mathbb{Z}^{*\times n}$ and $\mathbb{Z}^{n\times *}$ indicates an arbitrary (not necessarily the same) dimension. The $*$ in $\mathrm{Trunc}(*,k)$ indicates an arbitrary $n \times n$ integer matrix.

LEMMA 4. *If $F \in \mathbb{Z}^{*\times n}$ and $G \in \mathbb{Z}^{n\times *}$ then*

$$||\mathrm{Left}(F\,\mathrm{Trunc}(*,k)G,k)|| \le n||G|| + n^2||F||||G||.$$

PROOF. Let $L_1 := \mathrm{Trunc}(F\,\mathrm{Trunc}(*,k),k)$ and $H_1 := \mathrm{Left}(F\,\mathrm{Trunc}(*,k),k)$ and decompose $F\,\mathrm{Trunc}(*,k) = L_1 + H_1 X^k$. We have thus reduced our problem to bounding $||\mathrm{Left}((L_1 + H_1X^k)G,k)$. Decompose $L_1G = L_2 + H_2X^k$. Then $\mathrm{Left}((L_1+H_1X^k)G,k) = H_2 + H_1G$. By Lemma 3 we have $||H_1|| \le n||F||$ and $||H_2|| \le n||G||$. Using $||H_1G|| \le n||H_1||||G||$ gives $||H_2 + H_1G|| \le n^2||F||||G|| + n||G||$. □

3. OUTLINE OF THE ALGORITHM

Fix matrices $A, C \in \mathbb{Z}^{n\times n}$ with A nonsingular. Given $(B,D) \in (\mathbb{Z}^{n\times n}, \mathbb{Z}^{n\times n})$, our goal is to determine if $CA^{-1}B = D$. We will work in an X-adic number system with X chosen to have size (bitlength) slightly larger than the size of entries in B and D. If $CA^{-1}B = D$, then simultaneously pre-multiplying A^{-1} by C and post-multiplying by B has the effect of annihilating the possibly infinite X-adic expansion of A^{-1}, leaving behind only a leftover term D which satisfies $\mathrm{Trunc}(D,1) = D$. It will be sufficient to determine if $\mathrm{Trunc}(CA^{-1}B,k) = D$ for large enough k, which initially is about n. We may assume that k is a power of two. Consider pre- and post-multiplying the truncated expansion $\mathrm{Trunc}(A^{-1},k)$ by C and B. If $CA^{-1}B = D$ then we must have

$$C\,\mathrm{Trunc}(A^{-1},k)B = D + TX^k.$$

Because we truncated the expansion there will be a modulo X^k adjustment TX^k. From our choice of X, which will be sufficiently larger than $||C||$ and $||D||$, this adustment will satisfy $\mathrm{Trunc}(T,1) = T$ (this will follow from Lemma 4 and (6)).

Reduction to two problems of half the order

We can decompose A^{-1} as follows.

$$\begin{aligned} \mathrm{Trunc}(A^{-1},k) &= \mathrm{Trunc}(A^{-1},k/2) \\ &\quad + \mathrm{Trunc}(A^{-1}R,k/2)X^{k/2} \qquad (7) \end{aligned}$$

where $R := \mathrm{Res}(A, I, k/2)$. Suppose that $CA^{-1}B = D$. If we pre- and post-multiply $\mathrm{Trunc}(A^{-1}, k/2)$ by C and B, we will obtain

$$C\,\mathrm{Trunc}(A^{-1}, k/2)B = D - \bar{D}X^{k/2}, \tag{8}$$

where $-\bar{D}X^{k/2}$ is the modulo $X^{k/2}$ adjustment. If we pre- and post-multiply the high order part $\mathrm{Trunc}(A^{-1}R, k/2)$ by C and B we must obtain

$$C\,\mathrm{Trunc}(A^{-1}R, k/2)X^{k/2}B = \bar{D}X^{k/2} + *X^k. \tag{9}$$

Comparing with (7), we see that the sum of the left hand sides of equations (8) and (9) is equal to $C\,\mathrm{Trunc}(A^{-1}, k)B$. Because we assumed that $CA^{-1}B = D$, the modulo $X^{k/2}$ adjustment in (8) must be the negation of the leftover term $\bar{D}X^{k/2}$ in (9). (In other words, a necessary but not sufficient condition for $CA^{-1}B = D$ to hold is that the modulo $X^{k/2}$ adjustment in (8) cancels out the leftover term in (9).) Our original problem of determining if $\mathrm{Trunc}(CA^{-1}B, k) = D$ is thus equivalent to the following pair of problems: does $\mathrm{Trunc}(CA^{-1}B, k/2) = D$ and $\mathrm{Trunc}(CA^{-1}RB, k/2) = \bar{D}$? Let $\bar{B} := RB$. We can combine these two problems into a single problem of double the column dimension:

$$\mathrm{Trunc}(CA^{-1}\left[\begin{array}{c|c} B & \bar{B} \end{array}\right], k/2) \stackrel{?}{=} \left[\begin{array}{c|c} D & \bar{D} \end{array}\right].$$

To accomplish the problem reduction we need to compute $\bar{D}$ shown in (8). Although we don't have the entire expansion $\mathrm{Trunc}(A^{-1}, k/2)$, we do have the high-order component E.

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

On the one hand, if $\mathrm{Trunc}(C\,\mathrm{Trunc}(A^{-1}, k/2)B, k/2)$ is sufficiently small (e.g., $||\cdot|| < X$) then $\mathrm{Trunc}(CEB, 2)$ will also be small (Lemma 7). On the other hand, if $\mathrm{Trunc}(CEB, 2)$ is sufficiently small (which we can easily check by direct computation) then we can compute $\bar{D}$ as $\bar{D} := \mathrm{Left}(CEB, 2)$ (Theorem 8). If we determine that $\mathrm{Trunc}(CEB, 2)$ is not small enough we report ("no,", $\mathrm{Trunc}(CA^{-1}B, k) \neq D$).

Reduction of the bitlength

We have seen above how to reduce a single problem of given order to an equivalent problem of half the order but with twice the column dimension. One issue is that the numbers in the new problem might be more than twice the size (bitlength) of numbers in the original problem (e.g., only $||\cdot|| < X^2$ instead of $||\cdot|| < X$ may apply). Let (B, D) be a problem instance for which we want to reduce the bitlength. We have the decomposition

$$\begin{aligned}\mathrm{Trunc}(A^{-1}B, k) &= \mathrm{Trunc}(A^{-1}B, 2) \\ &\quad + \mathrm{Trunc}(A^{-1}R, k-2)X^2 \end{aligned} \tag{10}$$

where $R := \mathrm{Res}(A, B, 2)$. Throughout the algorithm D will be small enough so that if $CA^{-1}B = D$ then we must also have $\mathrm{Trunc}(CA^{-1}B, 2) = D$ (Lemma 5). Assume the check $\mathrm{Trunc}(CA^{-1}B, 2) = D$ holds. (If not we report ("no,", $\mathrm{Trunc}(CA^{-1}B, k) \neq D$).) We then have

$$C\,\mathrm{Trunc}(A^{-1}B, 2)B = D - D'X^2,$$

where $-D'X^2$ is the modulo X^2 adjustment. If it is indeed the case that $\mathrm{Trunc}(CA^{-1}B, k) = D$, then pre- and post-multiplying $\mathrm{Trunc}(A^{-1}R, k-2)$ by C and D must give

$$C\,\mathrm{Trunc}(A^{-1}R, k-2)X^2 = D'X^2 + *X^k.$$

We have already checked that $\mathrm{Trunc}(CA^{-1}B, 2) = D$, so we may conclude that $\mathrm{Trunc}(CA^{-1}B, k) = D$ if and only if $\mathrm{Trunc}(CA^{-1}R, k-2) = D'$ (Theorem 6). Note that the decomposition in (10) is based on $A^{-1}B$ instead of A^{-1} as in (7). Basing the construction on $A^{-1}B$ lets us prove good bounds on the size of numbers in B' and D' (also in Theorem 6).

Compression of column dimension

Using the two recipes described above we can transform one certification problem of order k and bitlength $< \log_2 X$ to two certification problems of order $k/2$ and bitlength $< \log_2 X$. The cost of the order reduction and bitlength reduction is $O(\mathsf{MM}(n)\mathsf{M}(\log X))$ bit operations. Repeating this process recursively gives an algorithm for Schur complement zero certification that has running time

$$T_{\mathrm{bad}}(k) = 2T_{\mathrm{bad}}(k/2) + \Theta(\mathsf{MM}(n)\mathsf{M}(\log X))$$

bit operations. Unfortunately, the solution to this recurrence is $T_{\mathrm{bad}}(n) = \Theta(n\mathsf{MM}(n)\mathsf{M}(\log X))$. We now explain the key ingredient of the algorithm which allows to compress a certificate problem with column dimension $6n$ down to a problem with column dimension only $3n$. This leads to an overall running time which satisfies the recurrence

$$T_{\mathrm{good}}(k) = T_{\mathrm{good}}(k/2) + \Theta(\mathsf{MM}(n)\mathsf{M}(\log X)).$$

Note that $T_{\mathrm{good}}(n) = O((\log n)\mathsf{MM}(n)\mathsf{M}(\log X))$ as desired.

Let (B, D) be a multi-problem instance, for example with $(B, D) \in (\mathbb{Z}^{n\times 6n}, \mathbb{Z}^{n\times 6n})$ of column dimension 6. Then the $n \times 6n$ matrix $\mathrm{Trunc}(CA^{-1}B, k) - D$ is the zero matrix if and only if the $n \times n$ symmetric and positive semi-definite matrix

$$(\mathrm{Trunc}(CA^{-1}B, k) - D)(\mathrm{Trunc}(CA^{-1}B, k) - D)^T$$

is the zero matrix. Expanding out gives an expression with four terms $P_1 - P_2 - P_3 + P_4$, where

$$\begin{aligned} P_1 &:= \mathrm{Trunc}(CA^{-1}B, k)\,\mathrm{Trunc}(B^T(A^{-1})^TC^T, k) \\ P_2 &:= \mathrm{Trunc}(CA^{-1}B, k)\,D^T \\ P_3 &:= D\,\mathrm{Trunc}(B^T(A^{-1})^TC^T, k) \\ P_4 &:= DD^T \end{aligned}$$

Thus, $\mathrm{Trunc}(CA^{-1}B, k) = D$ if and only if $P_1 = P_2 = P_3 = P_4 = DD^T$. Actually, $P_2 = DD^T$ if and only if $P_3 = DD^T$ since $P_3 = P_2^T$ and $(DD^T)^T = DD^T$. We have reduced our problem to determining if $P_1 = DD^T$ and $P_2 = DD^T$. First consider P_2. We have precomputed the high order part R of $\mathrm{Trunc}(CA^{-1}, k)$.

$$\mathrm{Trunc}(CA^{-1}, k) = * + *X + \cdots + *X^{k-3} + \overbrace{*X^{k-2} + *X^{k-1}}^{RX^{k-2}}.$$

If $\mathrm{Trunc}(CA^{-1}B, k) = D$ then $\mathrm{Trunc}(RB, 2)$ is small enough so that $\mathrm{Left}(\mathrm{Trunc}(RB, 2), 1)$ is the zero matrix (Lemma 9). Assume that $\mathrm{Left}(\mathrm{Trunc}(RB, 2), 1)$ is indeed the zero matrix. (If not, we return ("no,", $\mathrm{Trunc}(CA^{-1}B, k) \neq D$).) By part 3 of Lemma 10, our check that $\mathrm{Left}(\mathrm{Trunc}(RB, 2), 1)$ is the zero matrix implies that $\mathrm{Trunc}(CA^{-1}B, k)$ is small enough to ensure that

$$\mathrm{Trunc}(CA^{-1}BD^T, k) = \mathrm{Trunc}(CA^{-1}B, k)D^T.$$

We conclude that $P_2 = \mathrm{Trunc}(CA^{-1}BD^T, k)$. The important observation here is that BD^T is $n \times n$. One of our

subproblems will be to verify that $\text{Trunc}(CA^{-1}(BD^T), k) = DD^T$.

Now consider P_1. To apply a similar technique as for P_2, we construct the following decomposition.

$$\text{Trunc}(B^T(A^{-1})^T C^T, k) = B^T \text{Trunc}((A^{-1})^T C^T, k) - P^T X^k$$

Lemma 10 shows that we can construct the matrix P^T efficiently using R because $\text{Left}(\text{Trunc}(RB, 2), 1)$ is the zero matrix. Substituting the above into $P_1 - DD^T$ gives that $P_1 = DD^T$ if and only if both of the following hold:

$$\text{Trunc}(CA^{-1}B, k)B^T = DB^T \tag{11}$$

and

$$\begin{aligned} &\text{Left}(DB^T \text{Trunc}((A^{-1})^T C^T, k), k) \\ &= \text{Trunc}(CA^{-1}B, k)P^T. \end{aligned} \tag{12}$$

Consider (11). Because $\text{Trunc}(CA^{-1}B, k)$ is small enough, we have $\text{Trunc}(CA^{-1}B, k)B^T = \text{Trunc}(CA^{-1}BB^T, k)$. Similarly, we can prove that the left hand side of (12) is equal to DP^T (part 2 of Lemma 10) and the right hand side is equal to $\text{Trunc}(CA^{-1}BP^T, k)$.

Our original problem is reduced to verifying that

$$\text{Trunc}(CA^{-1}\left[BB^T | BD^T | BP^T\right], k) = \left[DB^T | DD^T | DP^T\right],$$

which has column dimension only $3n$.

4. BITLENGTH REDUCTION

In this section we are starting with the following problem.

$$\text{Start:} \left[\begin{array}{l} \textbf{Input: } B, D \in \mathbb{Z}^{n\times m},\ k \geq 2 \\ \textbf{Condition: } ||B|| \leq 6n^3\beta^2,\ ||D|| \leq 6n^4\beta^2 \\ \textbf{Question: } \text{Does } \text{Trunc}(CA^{-1}B, k) = D? \end{array}\right. \tag{13}$$

Our goal is to produce an equivalent problem that has better bounds on the magnitude of entries.

$$\text{Target:} \left[\begin{array}{l} \textbf{Input: } B', D' \in \mathbb{Z}^{n\times m} \\ \textbf{Condition: } ||B'|| \leq n\beta,\ ||D'|| \leq n\beta \\ \textbf{Question: } \text{Does } \text{Trunc}(CA^{-1}B', k-2) = D'? \end{array}\right.$$

The algorithm to accomplish this is given below.

`ReduceBitlength`(B, D)
if $\text{Trunc}(CA^{-1}B, 2) \neq D$ **then**
  return "no"
else
  $B' := \text{Left}(-A\,\text{Trunc}(A^{-1}B, 2)$;
  $D' := -\text{Left}(C\,\text{Trunc}(A^{-1}B, 2), 2)$;
  return (B', D')
fi

The rest of this section proves correctness of the algorithm.

LEMMA 5. *If* $\text{Trunc}(CA^{-1}B, k) = D$ *then we have that* $\text{Trunc}(CA^{-1}B, 2) = D$.

PROOF. The bound for $||D||$ in (13) satisfies (6) so that $\text{Trunc}(D, 1) = D$. It follows that $\text{Trunc}(D, 2) = D$ also. □

The following theorem assumes $k \geq 2$ and the bounds for $||B||$ and $||D||$ indicated in (13). The assumption (implicit throughout the paper) that $||A||, ||C|| \leq \beta$ is also used in the proof. The matrices B' and D' are as in Algorithm `ReduceBitlength` given above.

THEOREM 6. *If* $\text{Trunc}(CA^{-1}B, 2) = D$ *then*

$$\text{Trunc}(CA^{-1}B, k) = D \Longleftrightarrow \text{Trunc}(CA^{-1}B', k-2) = D'.$$

Moreover, $||B'|| \leq n\beta$ *and* $||D'|| \leq n\beta$.

PROOF. The bound for $||B||$ in (13) satisfies (6) so that $\text{Trunc}(B, 1) = B$. It follows that $\text{Trunc}(B, 2) = B$ also. By Lemma 2

$$A^{-1}B = \text{Trunc}(A^{-1}B, 2) + A^{-1}B'X^2$$

It follows that

$$\begin{aligned} CA^{-1}B &= C\,\text{Trunc}(A^{-1}B, 2) + CA^{-1}B'X^2 \\ &= D - D'X^2 + CA^{-1}B'X^2 \end{aligned}$$

The theorem follows from the last equation. The bounds for $||B'||$ and $||D'||$ follow from Lemma 3 using the assumption $||C||, ||A|| \leq \beta$. □

5. ORDER REDUCTION

In this section we are starting with the following problem.

$$\text{Start:} \left[\begin{array}{l} \textbf{Input: } B, D \in \mathbb{Z}^{n\times m},\ k > 2 \text{ a power of 2} \\ \textbf{Condition: } ||B|| \leq n\beta,\ ||D|| \leq n\beta \\ \textbf{Question: } \text{Does } \text{Trunc}(CAB, k) = D? \end{array}\right. \tag{14}$$

Our goal is to produce an equivalent problem of only half the order (but with twice the column dimension).

$$\text{Target:} \left[\begin{array}{l} \textbf{Input: } B', D' \in \mathbb{Z}^{n\times 2m} \\ \textbf{Condition: } ||B'|| \leq n^3\beta^2,\ ||D'|| \leq n^2\beta + n^3\beta^2 \\ \textbf{Question: } \text{Does } \text{Trunc}(CAB', k/2) = D'? \end{array}\right.$$

The algorithm to accomplish the problem transformation, given below, makes use of the single high-order inverse component

$$E := \text{Left}(\text{Trunc}(A^{-1}, k/2), k/2 - 2).$$

`ReduceToHalfOrder`(B, D, E)
if $\text{Left}(\text{Trunc}(CEB, 2), 1)$ is not the zero matrix
  then return "no"
else
  $B' := \left[\ B \mid \text{Left}(-AE, 2)B\ \right]$;
  $D' := \left[\ D \mid -\text{Left}(CEB, 2)\ \right]$;
  return (B', D')
fi

The rest of this section proves correctness of the algorithm.

LEMMA 7. *If* $\text{Trunc}(CA^{-1}B, k) = D$ *then we have that* $\text{Left}(\text{Trunc}(CEB, 2), 1)$ *is the zero matrix.*

PROOF. First consider $k = 4$. Then $E = \text{Trunc}(A^{-1}, 2)$ and $\text{Left}(\text{Trunc}(CEB, 2), 1) = \text{Left}(D, 1)$, which is zero in light of (6) and the bound for $||D||$ in (14). Now consider the case $k \geq 8$. We have $\text{Trunc}(A^{-1}, k/2) = \text{Trunc}(A^{-1}, k/2 - 2) + EX^{k/2-2}$, giving

$$\begin{aligned} D &= \text{Trunc}(CA^{-1}B, k/2) \\ &= \text{Trunc}(C\,\text{Trunc}(A^{-1}, k/2-2)B \\ &\qquad + CEBX^{k/2-2}, k/2). \end{aligned} \tag{15}$$

We have already established that $\text{Trunc}(D, 1) = D$, so we also have $\text{Trunc}(D, k/2 - 2) = D$, which leads to

$$\begin{aligned} &C\,\text{Trunc}(A^{-1}, k/2-2)B = \\ &\quad D + \text{Left}(C\,\text{Trunc}(A^{-1}, k/2-2)B, k/2-2)X^{k/2-2}. \end{aligned} \tag{16}$$

Substituting (16) into (15) reveals that

$$\begin{aligned}&\mathrm{Trunc}(CEB,2) = \\ &\quad -\mathrm{Left}(C\,\mathrm{Trunc}(A^{-1},k/2-2)B,k/2-2). \quad (17)\end{aligned}$$

We will now use $||B|| \le n\beta$ from (14) together with the assumption that $||A|| \le \beta$. Using Lemma 4 to bound the magnitude of the right hand side of (17) gives

$$||\mathrm{Trunc}(CEB,2)|| \le n||B|| + n^2||A||||B|| \le n^2\beta + n^3\beta^2,$$

which satisfies (6) so that $\mathrm{Trunc}(CEB,1) = CEB$. The result follows. □

The following theorem assumes $k > 2$ and the bounds for $||B||$ and $||D||$ in (14). The assumption $||A||, ||C|| \le \beta$ is also used. B' and D' are as in Algorithm `ReduceToHalfOrder` given above.

THEOREM 8. *If* $\mathrm{Left}(\mathrm{Trunc}(CEB,2),1)$ *is zero, then*

$$\mathrm{Trunc}(CA^{-1}B,k) = D \iff \mathrm{Trunc}(CA^{-1}B',k/2) = D'.$$

Moreover, $||B'|| \le n^3\beta^2$ *and* $||D'|| \le n^2\beta + n^3\beta^2$.

PROOF. Lemma 2 gives

$$A^{-1} = \mathrm{Trunc}(A^{-1},k/2) + A^{-1}\mathrm{Left}(-AE,2)X^{k/2}.$$

Premultiplying by C and postmultiplying by B reveals that $\mathrm{Trunc}(CA^{-1}B,k) = D$ if and only if $\mathrm{Trunc}(CA^{-1}B,k/2) = D$ and

$$\begin{aligned}&\mathrm{Trunc}(CA^{-1}\mathrm{Left}(-AE,2)B,k/2) = \\ &\qquad -\mathrm{Left}(C\,\mathrm{Trunc}(A^{-1},k/2)B,k/2).\end{aligned}$$

It remains to establish that the right hand side of the last equation is equal to $-\mathrm{Left}(CEB,2)$. To this end, let $L = \mathrm{Trunc}(A^{-1},k/2-2)$ and consider the decomposition

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

Then

$$\begin{aligned}&C\,\mathrm{Trunc}(A^{-1},k/2)B \\ &\quad = CLB + CEBX^{k/2-2} \\ &\quad = \underbrace{CLB + \mathrm{Trunc}(CEB,2)X^{k/2-2}}_{||\cdot|| \le \gamma} + \mathrm{Left}(CEB,2)X^{k/2},\end{aligned}$$

where $\gamma \le (n||B|| + n^2||C||||B|| + X)X^{k/2-2}$; this bound for γ follows from Lemma 4 and the assumption that

$$\mathrm{Left}(\mathrm{Trunc}(CEB,2),1)$$

is the zero matrix (which implies $\mathrm{Trunc}(CEB,2) \le X$). Using $||B|| \le n\beta$ and $||C|| \le \beta$ gives $n||B|| + n^2||C||||B|| + X \le n^2\beta + n^4\beta^2 + X$, which upon division by X satisfies (6). It follows that

$$\begin{aligned}&\mathrm{Trunc}(CLB + \mathrm{Trunc}(CEB,2)X^{k/2-2},k/2) = \\ &\qquad CLB + \mathrm{Trunc}(CEB,2)X^{k/2-2}.\end{aligned}$$

Finally, the claimed bounds for $||B'||$ and $||D'||$ follow from Lemmas 3 and 4 respectively, using $||A||, ||C|| \le \beta$ and the bound for $||B||$ in (14). □

6. DIMENSION REDUCTION

In this section we are starting with the following problem with column dimension up to $6n$.

$$\text{Start:}\left[\begin{array}{l}\textbf{Input: } B, D \in \mathbb{Z}^{n\times\overset{\le 6n}{\square m}},\ k \ge 4 \text{ a power of 2}\\ \textbf{Condition: } ||B|| \le n\beta,\ ||D|| \le n\beta\\ \textbf{Question: } \text{Does } \mathrm{Trunc}(CAB,k) = D?\end{array}\right. \quad (18)$$

Our goal is to produce an equivalent problem that has column dimension $3n$.

$$\text{Target:}\left[\begin{array}{l}\textbf{Input: } B', D' \in \mathbb{Z}^{n\times 3n}\\ \textbf{Condition: } ||B'|| \le 6n^3\beta^2,\ ||D'|| \le 6n^4\beta^2\\ \textbf{Question: } \text{Does } \mathrm{Trunc}(CAB',k) = D'?\end{array}\right. \quad (19)$$

The algorithm to accomplish the problem transformation, given below, makes use of the single high-order component

$$R := \mathrm{Left}(\mathrm{Trunc}(CA^{-1},k),k-2).$$

`ReduceColumnDimension`(B,D,R)
if $\mathrm{Left}(\mathrm{Trunc}(RB,2),1)$ is not the zero matrix
  then return "no"
else
  $P := \mathrm{Left}(\mathrm{Trunc}(RB,2),2)$;
  $B' := \left[\begin{array}{c|c|c} BD^T & BB^T & BP^T \end{array}\right]$;
  $D' := \left[\begin{array}{c|c|c} DD^T & DB^T & DP^T \end{array}\right]$;
  return (B',D')
fi

The rest of this section proves correctness of the algorithm. For convenience, let H stand for A^{-1}.

LEMMA 9. *If* $\mathrm{Trunc}(CHB,k) = D$ *then*

$$\mathrm{Left}(\mathrm{Trunc}(RB,2),1)$$

is the zero matrix.

PROOF. Decompose $\mathrm{Trunc}(CH,k) = \mathrm{Trunc}(CH,k-2) + RX^{k-2}$. Then

$$D = \mathrm{Trunc}(\mathrm{Trunc}(CH,k-2)B + RBX^{k-2},k).$$

The bound for $||D||$ in (18) satisfies (6) so $\mathrm{Trunc}(D,1) = D$. It follows that $\mathrm{Trunc}(D,k-2) = D$ also, so we must have

$$\mathrm{Trunc}(RB,2) = \mathrm{Left}(\mathrm{Trunc}(CH,k-2)B,k-2).$$

Lemma 3 bounds the right hand side of the last equation by $n||B|| = n^2\beta$, which satisfies (6) so that $\mathrm{Trunc}(RB,1) = RB$. The result follows. □

LEMMA 10. *Define* $P := \mathrm{Left}(\mathrm{Trunc}(CH,k)B,k)$. *If*

$$\mathrm{Left}(\mathrm{Trunc}(RB,2),1)$$

is the zero matrix then

1. $P = \mathrm{Left}(RB,2)$ *with* $||P|| \le n^2\beta$
2. $\mathrm{Left}(\mathrm{Trunc}(CH,k)BD^T,k) = PD^T$.
3. $\mathrm{Trunc}(CHB,k)F^T = \mathrm{Trunc}(CHBF^T,k)$ *for any matrix* $F \in \mathbb{Z}^{n\times 6n}$ *with* $||F|| \le n^2\beta$.

PROOF. Let $L = \mathrm{Trunc}(CH,k-2)$ and consider the decomposition $\mathrm{Trunc}(CH,k) = L + RX^{k-2}$. Then

$$\begin{aligned}&\mathrm{Trunc}(CH,k)B \\ &\quad = LB + RBX^{k-2} \\ &\quad = \underbrace{LB + \mathrm{Trunc}(RB,2)X^{k-2}}_{||\cdot|| \le (n||B|| + X)X^{k-2}} + \mathrm{Left}(RB,2)X^k\end{aligned}$$

The $||\cdot||$ bound follows from Lemma 3 and the assumption that $\text{Left}(\text{Trunc}(RB,2),1)$ is the zero matrix. Using $||B|| \leq n\beta$ and dividing by X shows the bound satisfies (6) so that

$$\text{Trunc}(LB + \text{Trunc}(RB,2)X^{k-2}, k) = LB + \text{Trunc}(RB,2)X^{k-2}.$$

The first claim of the lemma follows.

Now consider the second claim.

$$\begin{aligned}
&\text{Trunc}(CH,k)BD^T \\
&= LBD^T + RBD^TX^{k-2} \\
&= \underbrace{LBD^T + \text{Trunc}(RB,2)D^TX^{k-2}}_{||\cdot|| \leq \gamma} + \text{Left}(RB,2)D^TX^k
\end{aligned}$$

where $\gamma \leq (n||BD^T|| + 6nX||D^T||)X^{k-2}$. Next note that $n||BD^T|| + nX||D^T|| \leq 6n^4\beta^2 + 6n^2X\beta$, which when divided by X satisfies (6), so

$$\text{Trunc}(LBD^T + \text{Trunc}(RB,2)D^TX^{k-2}, k) = LBD^T + \text{Trunc}(RB,2)D^TX^{k-2}.$$

The second claim follows.

Now consider the third claim. Our proof of the first claim established that $||\text{Trunc}(CHB,k)|| \leq (n||B|| + X)X^{k-2}$. It follows that $||\text{Trunc}(CHB,k)F^T|| \leq 6n^3(n^2\beta + X)\beta X^{k-2}$, which when divided by X^{k-1} satisfies (6). □

THEOREM 11. *Let P be as in Lemma 10. If*

$$\text{Left}(\text{Trunc}(CEB,2),1)$$

is the zero matrix, then $\text{Trunc}(CHB,k) = D$ *if and only if both of the following hold:*

1. $\text{Trunc}(CHBD^T,k) = DD^T$
2. $\text{Trunc}(CHBB^T,k)\text{Trunc}(HTC^T,k) - \text{Trunc}(CHBR^T)X^k = DD^T$

PROOF. $\text{Trunc}(CHB,k) - D$ is the zero matrix if and only if

$$(\text{Trunc}(CHB,k) - D)(\text{Trunc}(CHB,k) - D)^T \quad (20)$$

is the zero matrix. Expanding and comparing terms reveals that (20) is zero if and only if

$$\text{Trunc}(CHB,k)D^T = DD^T \quad (21)$$

and

$$\text{Trunc}(CHB,k)\text{Trunc}(B^TH^TC^T,k) = DD^T \quad (22)$$

By part 3 of Lemma 10, we may substitute

$$\text{Trunc}(CHB,k)D^T = \text{Trunc}(CHBD^T,k)$$

into (21), giving the first condition in the theorem.

Next we reformulate (22) to an equivalent identity. We have

$$\text{Trunc}(CHB,k) = \text{Trunc}(CH,k)B - PX^k,$$

the transpose of which is

$$\text{Trunc}(B^TH^TC^T,k) = B^T\text{Trunc}(H^TC^T,k) - P^TX^k. \quad (23)$$

Substitute (23) into (22) to obtain the equivalent identity

$$\text{Trunc}(CHB,k)B^T\text{Trunc}(H^TC^T,k) - \text{Trunc}(CHB,k)P^TX^k = DD^T. \quad (24)$$

Together with $||B|| \leq n\beta$ from (18), Lemma 3 gives $||R|| \leq n^2\beta$. The second condition of the theorem now follows from two applications of part 3 of Lemma 10 to equation (24). □

THEOREM 12. *Let P be as in Lemma 10. If*

$$\text{Left}(\text{Trunc}(RB,2),1)$$

is the zero matrix, then $\text{Trunc}(CHB,k) = D$ *if and only if all of the following hold*

$$\text{Trunc}(CHBD^T,k) = DD^T \quad (25)$$

$$\text{Trunc}(CHBB^T,k) = DB^T \quad (26)$$

$$\text{Trunc}(CHBP^T,k) = DP^T \quad (27)$$

PROOF. The "only if" direction is clear. Now suppose all of (25), (26) and (27) hold. Then (25) implies that the first condition of Theorem 11 holds. Substituting (26) and (27) into the left hand side of the second condition of Theorem (11) gives

$$DB^T\text{Trunc}(H^TC^T,k) - DP^TX^k \quad (28)$$

Now note that

$$\begin{aligned}
&DB^T\text{Trunc}(H^TC^T,k) \\
&= \text{Trunc}(DB^T\text{Trunc}(H^TC^T,k),k) \\
&\quad + \text{Left}(DB^T\text{Trunc}(H^TC^T,k),k)X^k \\
&= DD^T + \text{Left}(DB^T\text{Trunc}(H^TC^T,k),k)X^k \quad (29) \\
&= DD^T + \text{Left}(DP^T,k)X^k \quad (30) \\
&= DD^T + DP^TX^k \quad (31)
\end{aligned}$$

Here, line (29) follows from (25), and (30) from part 2 of Lemma 10. Finally, substituting (31) into (28) gives DD^T, which matches the right hand side of the second condition of Theorem 11. □

7. ALGORITHM

Algorithm `SchurCert` is shown in Figure 1. Lemmas 5, 7 and 9 ensure that if $CA^{-1}B = D$ then none of calls to the subroutines `ReduceBitlength`, `ReduceToHalfOrder` and `ReduceColumnDimension` will return "no,", respectively. Theorems 6, 8 and 12 ensure the equivalence of all the problems (i.e., they all have the same answer). The cost analysis of the algorithm was detailed in Section 3. We obtain the following result.

THEOREM 14. *Algorithm* `SchurCert` *works as stated. The running time is* $O((\log n)\mathsf{MM}(n)\mathsf{M}(\log n + \log||A||))$ *bit operations.*

8. A LAS VEGAS RANK ALGORITHM

Let $A \in \mathbb{Z}^{n\times m}$. Assume without loss of generality that $n \leq m$. Minors of A are bounded in magnitude by $(n||A||)^n$. Choose a random prime p with $\log p \in \Theta(\log n + \text{loglog}\,||A||)$ so that the rank of A mod p over $\mathbb{Z}/(p)$ is equal to the rank of A over $\mathbb{Z}$ with probability at least $1/4$. For details of choosing such a prime see for example [3]. Compute A mod p using nm modular reductions. This costs $O(nm\mathsf{M}(\log||A||))$ bit operations, which will be dominated by the other steps. (Note that if $p > ||A||$ the modular reduction is free.)

ALGORITHM 13. `SchurCert`(A, B, C, D)
Input: $A, B, C, D \in \mathbb{Z}^{n\times n}$, $p \in \mathbb{Z}_{>0}$ with $p \perp \det A$
Output: "yes" if $CA^{-1}B{=}D$, "no" otherwise, or fail
Condition: $\log p \in O(\log n + \text{loglog } A)$
Note: Fail is returned with probability $< 1/2$.

1. $\beta := \max(||A||, ||B||, ||C||, ||D||)$;
   Compute the following as per Theorem 1:
   • An (X, t)-adic shifted number system.
   • k, the minimal power of 2 such that
   $X^k > ((n+1)\beta)^{n+1}$,
   together with
   $E^{(i)} := \text{Left}(\text{Trunc}(A^{-1}, 2^i), 2^i - 2)$
   and
   $R^{(i)} := \text{Left}(\text{Trunc}(CA^{-1}, 2^i), 2^i - 2)$
   for $1 \le i \le \log_2 k$.
   # Note: If the above fails then return fail.
2. **for** $i := (\log_2 k) - 1$ **downto** 2 **do**
   $(B, D) :=$ `ReduceToHalfOrder`$(B, D, E^{(i)})$;
   $(B, D) :=$ `ReduceBitlength`(B, D);
   $(B, D) :=$ `ReduceColumnDimension`$(B, D, R^{(i)})$;
   $(B, D) :=$ `ReduceBitlength`(B, D)
   od;
   if $\text{Trunc}(CA^{-1}B, 2) = D$ **then return** "yes"
   else return "no" **fi**

Figure 1: Algorithm `SchurCert`

Next compute the rank $\bar{r}$ of $A \bmod p$ using a rank sensitive variation of `LSP` decomposition [7] developed in [8]. The cost is $O((\log \bar{r})nm(\mathsf{MM}(\bar{r})/\bar{r}^2))$ arithmetic operations modulo $\mathbb{Z}/(p)$ [8, Theorem 3.3]. We remark that the extra $\log \bar{r}$ will not occur if $n^{2+\epsilon} \in O(\mathsf{MM}(r))$ for some $\epsilon > 0$, see [8] for details. All the arithmetic operations are additions, subtractions and multiplications, each with a cost of $O(\mathsf{M}(\log p))$ bit operations, plus $\bar{r}$ modular inverse, each costing $O(\mathsf{M}(\log p) \text{loglog } p)$ bit operations. The bit cost of the `LSP` decomposition is thus

$$O((\log \bar{r})nm(\mathsf{MM}(\bar{r})/\bar{r}^2)\mathsf{M}(\log n + \text{loglog } ||A||)).$$

The `LSP` decomposition also gives a row and column permutation of A such that we can write

$$A = \left[\begin{array}{c|c} \bar{A} & B \\ \hline C & D \end{array}\right]$$

with $\bar{A}$ nonsingular.

Finally, partition D into at most $\lceil (n-\bar{r})/\bar{r} \rceil \lceil (m-\bar{r})/\bar{r} \rceil$ blocks of dimension bounded by $\bar{r}$, and for each block use algorithm `SchurCert` to test that the Schur complement of D is the zero matrix. Note that the high-order components of $\bar{A}^{-1}$ and $C\bar{A}^{-1}$ need to be computed only once. One subtlety to be aware of is that the shifted number system from [15] will require choosing an X that satisfies $\log X \in O(\log n + \log ||A||)$, not just $\log X \in O(\log r + \log ||A||)$. This is because C has row dimension $\Theta(n)$, implying the need to certify correctness of $\Theta((\log r)nr)$ X-adic coefficients.

Since $\bar{r} \le r$ and $\log X \in O(\log n + \log ||A||)$, we get the following result.

THEOREM 15. *Given an $A \in \mathbb{Z}^{n\times m}$, there exists a Las Vegas probabilistic algorithm that computes the rank r of A using an expected number of*

$$O((\log r)nm(\mathsf{MM}(r)/r^2)\mathsf{M}(\log \min(n, m) + \log ||A||))$$

bit operations.

9. REFERENCES

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

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

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

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

[5] M. Giesbrecht, A. Lobo, and B. D. Saunders. Certifying inconsistency of sparse linear systems. In O. Gloor, editor, *Proc. Int'l. Symp. on Symbolic and Algebraic Computation: ISSAC '98*, pages 113—119. ACM Press, New York, 1998.

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

[7] 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.

[8] C.-P. Jeannerod. LSP Matrix Decomposition Revisited. Technical Report Research Report 2006-28, École normale supérieure de Lyon, LIP, September 1996.

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

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

[11] M. Pohst. A modification of the LLL reduction algorithm. *Journal of Symbolic Computation*, 4(1):123–127, 1987.

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

[13] D. Saunders and Z. Wan. Smith normal form of dense integer matrices, fast algorithms into practice. In J. Gutierrez, editor, *Proc. Int'l. Symp. on Symbolic and Algebraic Computation: ISSAC '04*, pages 274–281. ACM Press, New York, 2004.

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

[15] A. Storjohann. The shifted number system for fast linear algebra on integer matrices. *Journal of Complexity*, 21(4):609–650, 2005. Festschrift for the 70th Birthday of Arnold Schönhage.

[16] A. Storjohann and G. Villard. Computing the rank and a small nullspace basis of a polynomial matrix. In M. Kauers, editor, *Proc. Int'l. Symp. on Symbolic and Algebraic Computation: ISSAC '05*, pages 309–316. ACM Press, New York, 2005.