Report

A fast+practical+deterministic algorithm for triangularizing integer matrices

Author(s):
Storjohann, Arne

Publication Date:
1996

Permanent Link:

https://doi.org/10.3929/ethz-a-006651735 →

Technical Report No. 255, Department of Computer Science, ETH Zurich, December 1996

A Fast+Practical+Deterministic Algorithm for Triangularizing Integer Matrices*

Arne Storjohann
Institut für Wissenschaftliches Rechnen
ETH Zürich, Switzerland
storjoha@inf.ethz.ch

Abstract

This paper presents a new algorithm for computing the row reduced echelon form triangularization H of an $n \times m$ integer input matrix A. The cost of the algorithm is $O(nmr^2 \log^2 r||A|| + r^4 \log^3 r||A||)$ bit operations where r is the rank of A and $||A|| = \max_{ij} |A_{ij}|$. This complexity result assumes standard (quadratic) integer arithmetic but still matches, in the paramaters n, m and r, the best bit complexity we can reasonably hope for under the assumption of standard matrix arithmetic. A unimodular transforming matrix U which satisfies $UA = H$ is also computed within the same running time. As a direct application of our triangularization algorithm we give a fast algorithm for solving a system $A\vec{x} = \vec{b}$ of linear Diophantine equations. The algorithms presented here are both fast *and* practical. They are easily implemented, handle the case of input matrices having arbitrary shape and rank profile, and allow integer arithmetic to be performed in a residue number system.

1 Introduction

It follows from Hermite [Her51] that any $n \times m$ rank m integer matrix A can be transformed using a sequence of integer row operations to the unique upper triangular matrix

$$H = UA = \begin{bmatrix} h_1 & \bar{h}_{12} & \bar{h}_{13} & & \bar{h}_{1m} \\ & h_2 & \bar{h}_{23} & \cdots & \bar{h}_{2m} \\ & & h_3 & & \bar{h}_{3m} \\ & & & \ddots & \vdots \\ & & & & h_m \\ & & & & \\ & & & & \\ & & & & \end{bmatrix}$$

where h_j is positive for $1 \leq j \leq m$ and $\bar{h}_{ij}$ satisfies $0 \leq \bar{h}_{ij} < h_j$ for $1 \leq i < j \leq m$ and where the matrix U is unimodular (i.e. U has determinant ± 1). H is called the Hermite normal form (HNF) of A and U is called a transforming matrix. Computing HNFs of integer matrices is useful in many applications such as solving systems of linear

*This work has been partially supported by grants from the Swiss Federal Office for Education and Sciences in conjunction with partial support by ESPRIT LTR Project no. 20244 — ALCOM–IT.

Diophantine equations and computing related normal forms such as the Smith normal form [Sto96a].

Many algorithms for triangularizing integer matrices have been given previously. A partial list is [Bod56, Bla66a, Hu69, Bra71, BP74, KB79, CC82, DKT87, Ili89a, Dom89, HM91, HM94, SL96]. Early algorithms [Bod56, Bla66a, Hu69, Bra71, BP74] are not known to admit polynomial time running bounds — the main problem being the potential for rapid growth in the bit length of intermediate integer coefficients in the matrix being triangularized. The first polynomial time HNF algorithm was given by Kannan & Bachem [KB79] and later improved by Chou & Collins [CC82]. More recently, a new heuristic algorithm with good practical performance (experimentally justified) has been given by Havas & Majewski [HM94].

A new class of HNF algorithms has been presented by Domich, Kannan & Trotter [DKT87] (see also Iliopoulos [Ili89a]) which use modular arithmetic to avoid the problem of intermediate expression swell. This class of algorithms — which we call the mod d approach — perform all intermediate integer arithmetic modulo d where d is the absolute value of the determinant of a square nonsingular input matrix. An extension of the mod d approach to rectangular matrices is given by Hafner & McCurley [HM91] who also apply fast matrix multiplication techniques to the problem of triangularizing an integer matrix. A fast matrix multiplication decomposition for computing the complete HNF is given by Storjohann & Labahn [SL96]. All the normal form algorithms in [DKT87, Ili89a, Dom89, HM91, SL96, Sto96b] are based on the mod d approach.

A serious problem with the mod d approach is that d may be large. For an $n \times n$ input matrix A a worst case bound for d is $d \leq (n^{1/2}||A||)^n$ where $||A||$ denotes the largest magnitude entry of A. This bound — known as Hadamard's bound — is tight. This means that a mod d algorithm performs arithmetic with integers bounded in length by a relatively large $O(n \log n||A||)$ bits. For integers of this size it is desirable to perform intermediate computations in a residue number system modulo a basis of word size primes. A partial solution is given by Domich [Dom89] who proposes an extension of the mod d approach which works in a residue number system with moduli $(d_1, d_2, \ldots, d_k)$ where $d = d_1 d_2 \cdots d_k$ and the d_i's are pairwise relatively prime. The d_i's are obtained by computing a partial factorization of the modulus d. In many practical cases, though, no improvement is possible. Consider, for example, the case $d = 2^{1000}$ or $d = 3^{14} \cdot c_{2438}$ where c_{2438} is a number with 2438 decimal digits that resists factorization.

In this paper we give a new (i.e. non mod d) algorithm for computing the HNF of an $n \times m$ input matrix A. The algorithm has significant advantages over previous methods.

- The algorithm is fast. Unlike previous methods, the algorithm proposed here allows arithmetic to be performed in a residue number system modulo a basis of word size primes. This leads to a very good deterministic complexity bound using standard (quadratic) integer arithmetic. Previous algorithms [DKT87, Ili89a, Dom89, HM91, SL96] admit the complexity bound we prove for the algorithm presented here only under the assumption of asymptotically fast (but currently impractical) pseudo-linear integer arithmetic.

- The algorithm directly handles the case of input matrices having arbitrary shape and rank profile. This is crucial since many input matrices arising in practice are non square and without full column rank. Most previous HNF algorithms which are backed up by a complexity analysis have been presented for the case of square non-singular matrices (cf. [DKT87, Dom89, Ili89a]) or for full column rank matrices (cf.

[HM91]) but do not admit obvious (efficient) generalizations to the case of matrices with arbitrary shape and rank profile[1].

- The algorithm returns an $n \times n$ unimodular transforming matrix U which satisfies $UA = H$. The matrix U will have integer entries bounded in length by $O(m \log m||A||)$ bits. The only other deterministic algorithm which provides an efficient method for computing a U in the case of rectangular input matrices is the asymptotically fast algorithm proposed by Storjohann & Labahn [SL96] — when assuming pseudo-linear integer multiplication the algorithm in [SL96] is near optimal[2]. Under the assumption of standard integer and matrix multiplication, though, the bit complexity of the algorithm in [SL96] is still at least a factor of $O(m)$ more than that of the algorithm we present here. Computing a transforming matrix is important in such applications as solving systems of linear Diophantine equations.

In addition to (or in lieu of) a single unimodular matrix U, our algorithm also returns a factorization $QC = U$ for U where Q and C can be written in block form as

$$Q = \left[\begin{array}{c|c} Q_1 & \\ \hline Q_2 & I_{n-r'} \end{array}\right] \qquad C = \left[\begin{array}{c|c} C_1 & C_2 \\ \hline & I_{n-r'} \end{array}\right]$$

where $r' = \min(r+1, n)$ and r is the rank of the input matrix A. The advantage of returning the matrices Q and C instead of (or in addition to) the single matrix $U = QC$ is that Q and C will each have about only nr nonzero entries whereas U, in general, may have n^2 nonzero entries. Entries in Q will be bounded in length by $O(r \log r||A||)$ bits and entries in Q will be very small and admit the length bound of $O(\log r + \log\log||A||)$ bits. The decomposition $U = QC$ has a much simpler structure, fewer nonzero entries, and entries of smaller magnitude than the transforming matrix U returned by the asymptotically fast algorithm (which assumes $r = m$) given previously [SL96]. One reason for computing a unimodular transforming matrix U is to be able to evaluate the transforming matrix at a vector, that is, compute $U\vec{b}$ for an integer vector $\vec{b}$. Using the decomposition $U = QC$ we can accomplish this with only $O(nr)$ arithmetic operations as opposed to $O(n^2)$ arithmetic operations when U is dense.

To give complexity results we use the parameters θ and ϵ. Two $n \times n$ matrices over a ring can be multiplied in $O(n^\theta)$ ring operations and two $\lceil t \rceil$ bit integers can be multiplied in $O(t^{1+\epsilon})$ bit operations. The asymptotically fast (but currently impractical) algorithm of Coppersmith-Winograd [CW90] and the pseudo-linear algorithm of Schönhage-Strassen [SS71] allow $\theta = 2.376$ and any fixed (but positive) ϵ respectively. All algorithms in this paper assume the standard, practical algorithms for integer and matrix multiplication which have $\epsilon = 1$ and $\theta = 3$.

The cost of our HNF algorithm is $O(nm^3 \log^2 m||A|| + m^4 \log^3 m||A||)$ bit operations. When the rank r of the input matrix is less than m, the algorithm computes the row reduced echelon form of A and the complexity bound reduces to $O(nmr^2 \log^2 r||A|| + r^4 \log^3 r||A||)$ bit operations. The previously fastest[3] algorithm of Hafner & McCurley [HM91] works only for the case of input matrix with full column rank and requires $O\tilde{}(nm^3 \log||A||)$ bit operations[4] under the assumption of asymptotically fast (but currently impractical) pseudo-linear integer arithmetic — our algorithm assumes standard,

[1] See Subsection 2.1 for a discussion of this issue

[2] The HNF with transforming matrix is computed within a logarithmic factor of the time required to by the fastest known deterministic algorithm to verify that the rank of the input matrix is indeed m.

[3] barring the use of fast matrix multiplication techniques as in [SL96]

[4] To summarize complexity results we use soft-"Oh" notation: for real valued functions f and g, $f = O\tilde{}(g)$ if and only if $f = O(g \cdot \log^c g)$ for some constant $c > 0$.

quadratic integer arithmetic but still matches this complexity result in the parameters n and m. Assuming standard integer arithmetic the complexity of the algorithm in [HM91] increases to $O(nm^4 \log^2 ||A||)$ bit operations.

In Section 5 we apply our echelon form algorithm to the problem of solving a system of linear Diophantine equations $A\vec{x} = \vec{b}$ where $A \in \mathbb{Z}^{n \times m}$ and $\vec{b} \in \mathbb{Z}^{n \times 1}$. To summarize complexity results let $s = n + m$ and $\log ||A'|| = \log ||A|| + \log ||\vec{b}||$. Combining Blankinship's [Bla66b] method with our triangularization algorithm leads to an algorithm which requires $O^{\sim}(s^4 \log^3 ||A'||)$ bit operations to return the general solution of the system (or to determine that no solution exists). This complexity result improves in the parameter s on the previously fastest (asymptotic) algorithm of Iliopolous [Ili89b] which requires $O^{\sim}(s^{4.376} \log ||A'||)$ bit operations using asymptotically fast integer and matrix multiplication. A direct application of the algorithm in [Ili89b] using standard arithmetic calls for $O^{\sim}(s^6 \log^2 ||A'||)$ bit operations.

The rest of this paper is organised as follows. In Section 2 we recall the definition of the row reduced echelon form (RREF) — an extension of the HNF to the case of matrices with arbitrary shape and rank profile — and briefly discuss the problem of computing a transforming matrix for the RREF. In Section 3 some subroutines for matrix triangularization are presented and in Section 4 we present the new RREF algorithm itself. Section 5 shows how to apply RREF algorithm to the problem of solving a system of linear Diophantine equations.

2 Echelon Forms for Integer Matrices

Integer matrices A and B are said to be row equivalent if $A = UB$ for some unimodular matrix U. A matrix U is unimodular if $\det U = \pm 1$. In particular, the unimodular matrices are precisely those that are invertible over the ring of integers. The important point here is that if $A = UB$ with U unimodular then $B = U^{-1}A$ where U^{-1} is also unimodular. It follows that row equivalence is an equivalence relation for matrices of similar dimension. The matrix U corresponds to a sequence of elementary row operations: interchanging two rows; negating a row; adding a multiple of one row to different row. Any integer matrix A can be transformed using only elementary row operations to an upper triangular matrix H that satisfies the following conditions:

($e1$) Let r be the rank of H. Then the first r rows of H are nonzero.

($e2$) For $1 \leq i \leq r$ let $H[i, j_i]$ be the first nonzero entry in row i. Then $j_1 < j_2 < \ldots < j_r$.

A matrix H which satisfies ($e1$) and ($e2$) is said to be in row echelon form. For example, the 6×8 rank 5 matrix

$$\begin{bmatrix} 8 & 12 & 5 & -55 & 116 & 1369 & 639416 & 2486 \\ 0 & 0 & 0 & 1 & 551 & 7309 & 2024618 & -1852 \\ 0 & 0 & 0 & 0 & -9 & -4843 & -50732 & -1565 \\ 0 & 0 & 0 & 0 & 0 & 77 & 287 & 4416 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 50 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \tag{1}$$

is in row echelon form with rank profile $[j_1, j_2, j_3, j_4, j_5] = [1, 4, 5, 6, 8]$. The row echelon provides a non unique triangularization of the input matrix A (although the j_i's and the entries $H[i, j_i]$ are unique up to sign). The entire triangularization can be made unique

by applying further elementary row operations so that H satisfies

($e3$) $H[i, j_i] > 0$ for $1 \leq i \leq r$.

($e4$) For $1 \leq k < i \leq r$, $H[i, j_i] > H[k, j_i] \geq 0$.

The matrix H which satisfies ($e1$), ($e2$), ($e3$) and ($e4$) is called the row reduced echelon form of A. For example, the matrix

$$\begin{bmatrix} 8 & 12 & 5 & 0 & 1 & 51 & 28983 & 42 \\ 0 & 0 & 0 & 1 & 2 & 20 & 3920 & 5 \\ 0 & 0 & 0 & 0 & 9 & 69 & 32938 & 23 \\ 0 & 0 & 0 & 0 & 0 & 77 & 287 & 16 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 50 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \tag{2}$$

is the row reduced echelon form of the matrix in (1). When we say the row reduced echelon form H in (2) is unique we mean the following: if a second matrix B is both in row reduced echelon form and left equivalent to H then B and H are necessarily identical. This definition of the row reduced echelon is taken from Havas & Majewski [HM94], but see also [KKS90]. The form originates with Hermite [Her51] who first gave a constructive algorithm to compute H for the case of a square nonsingular integer matrix A. Hermite's result can be extended to matrices of arbitrary shape and rank profile using alternative definitions (cf. [Mac56, New72]) but these may not be unique. A brief discussion of this issue — as well as a proof that the row reduced echelon form as defined here provides a canonical form for row equivalence — can be found in [KKS90, Theorem 2.1].

Note that we reserve the term Hermite normal form for those cases where the input matrix A has full column rank — in this case $j_i = i$ for $1 \leq i \leq m$. This is because most previous algorithms in the literature which have had their running times analysed have been presented as "Hermite normal form" algorithms, that is, for the case of square nonsingular matrices (cf. [DKT87, Dom89, Ili89a]) or for full column rank matrices (cf. [HM91]) but do not admit obvious (efficient) generalizations to the case of matrices with arbitrary shape and rank profile.

2.1 The Transforming Matrix Problem

The problem of triangularizing an integer matrices has been well studied — it is useful to distinguish three problems of increasing difficulty. The first is to simply triangularize the input matrix A, that is, compute a general (non unique) row echelon form T of A. The second is to compute the unique row reduced echelon form H of A. The third is to compute, in addition to H, a unimodular transforming matrix U such that $UA = H$. This third problem deserves some comment. When A is square nonsingular than U is unique and can be computed using standard methods as $U \leftarrow HA^{-1}$. Moreover, the entries in H and U admit good length bounds. In what follows, we write A^{adj} to denote the adjoint of an $n \times n$ matrix A. Recall that entries in A^{adj} are, up to sign, $(n-1) \times (n-1)$ minors of A.

Lemma 1 *Let A be an $n \times n$ integral matrix and let d_j bound the magnitudes of entries in column j of A. Then $|\det A| \leq n^{n/2} d_1 d_2 \cdots d_n$. More simply we have $|\det A| \leq (n^{1/2}||A||)^n$.*

Proof. This is Hadamard's bound. For a proof see Gantmacher [Gan60]. ∎

Theorem 2 *Let U, A and H be $n \times n$ nonsingular integral matrices which satisfy $UA = H$ where H is in Hermite normal form. Then*

$$||H|| \leq |\det A| \leq (n^{1/2}||A||)^n \quad \textit{and} \quad ||U|| \leq ||A^{\mathsf{adj}}|| \leq ((n-1)^{1/2}||A||)^{n-1}$$

Proof. The bound for $|\det A|$ and $||A^{\mathsf{adj}}||$ follows from Hadamard's bound since each entry of $||A^{\mathsf{adj}}||$ is an $(n-1) \times (n-1)$ minor (or negative thereof) of A. The bound for $||H||$ follows by noting that off-diagonal entries of H are bounded in magnitude by the diagonal entries, the product which equals $|\det A|$. Let $t_1, t_2, \ldots, t_k$ be those diagonal entries of H, in order of increasing magnitude and with repeats, which are greater than 1. If $k = 0$ then $H = I_n$ and $U = A^{\mathsf{adj}}$ and the result follows. Assume henceforth that $k > 0$ and let $d = |\det A|$. Then for $i = 1, 2, \ldots, k$ we have $t_i \leq d/2^{k-i}$ (since $d = t_1 t_2 \cdots t_k$) and row i of H contains at most k nonzero entries, $k-1$ of which are bounded in magnitude by $d/2^{k-1} - 1$ and one of which is bounded in magnitude by $d/2^{k-1}$. Noting that $U = HA^{\mathsf{adj}}(1/d)$ yields

$$\begin{aligned} ||U|| &= ||HA^{\mathsf{adj}}(1/d)|| \\ &\leq (d/2^{k-1} + (k-1)(d/2^{k-1} - 1))(1/d)||A^{\mathsf{adj}}|| \\ &\leq (k/2^{k-1})||A^{\mathsf{adj}}|| \\ &\leq ||A^{\mathsf{adj}}|| \end{aligned}$$

which proves the lemma. ∎

Now consider the case when A is rectangular. In this case the matrix U is not unique and most previous algorithms for computing H don't provide an efficient method for computing a U. In one sense this is no essential restriction; Hafner & McCurley [HM91] suggest a technique for reducing the full column rank rectangular case to the square nonsingular case. The requirement that the input matrix be full column rank can also be dropped. In fact, any algorithm for computing the Hermite normal form of a square nonsingular matrix can be adapted to compute the row reduced echelon form with premultiplier for an input matrix of arbitrary shape and rank profile. We briefly outline the technique here — similar to that used by Kaltofen, Krishnamoorthy and Saunders [KKS90] in the context of polynomial matrices. Consider the case of an $n \times m$ rank r input matrix A. Using standard methods (e.g. Gaussian elimination) compute the rank profile of A, that is, determine the rank r of A and j_i for $1 \leq i \leq r$. At the same time recover a permutation matrix P such that PA has first r rows linearly independent. Let

$$A' = \left[\begin{array}{c|c} A_1 & \\ \hline A_2 & I_{n-r} \end{array}\right]$$

be the $n \times n$ matrix with column j equal to column j_i of PA for $1 \leq j \leq r$. Then the principal $r \times r$ submatrix A_1 of A' is nonsingular and A' will be nonsingular. Compute the Hermite normal form H' of A'. Compute U' using standard methods as $U' \leftarrow H'A'^{-1}$. The row reduced echelon form H of A can now be computed as $H \leftarrow UA$ where $U = U'P$. The technique just described, although correct, has two serious problems. First, computing the Hermite normal form of the augmented $n \times n$ matrix A' will be asymptotically more expensive than computing the Hermite normal form of the original $n \times m$ input matrix A. Second, the unimodular transforming matrix U returned may be dense, with $O(n^2)$ nonzero entries. For comparison, the algorithm we present in this paper returns admits a running time which is linear in n and returns representation for a U which is sparse, with only $O(nr)$ nonzero entries.

3 Some Subroutines

In this section we present the key subroutine of our row reduced echelon form algorithm. In Subsection 3.1 we present a deterministic preconditioning algorithm that reduces the problem of computing the gcd of n integers to that of only two. Subsection 3.2 builds on the result of Subsection 3.1 and gives a "column reduction" subrouting for effecting a particular unimodular transformation on an $n \times 2$ input matrix. This subroutine will form the basis for our row reduced echelon form algorithm presented in Section 4. In particular, r applications of the column reduction subroutine will transform an $n \times m$ rank r integral matrix to row reduced echelon form.

We will require the following number theoretic result.

Theorem 3 (Storjohann [Sto96c]) *Let r be a bound on the number of distinct prime divisors of a positive integer N. Given integers a and b with $\gcd(a,b) = 1$, there exists a nonnegative integer c bounded in magnitude by $2r^{3/2}$ that satisfies $\gcd(a + cb, N) = 1$.*

The practical algorithm that we use to produce a c satisfying the requirements of Theorem 3 can be described as follows: compute $\gcd(a + tb, N)$ for $t = 0, 1, 2, \ldots$ in succesion and set c to be the minimum t with $\gcd(a + tb, N) = 1$. The number of distinct prime divisors r of N is bounded by $\lfloor \log_2 N \rfloor$ and each gcd computation costs $O(\log^2 N)$ bit operations leading directly to a worst case complexity of $O(\log^{3.5} N)$ bit operations. In practice this method behaves more like $O(\log^2 N)$ bit operations. This brute force algorithm works very well in practice for two reasons. First, the bound $2r^{3/2}$ guaranteed by Theorem 3 is worst case. Secondly, N is typically not the product of large number of small primes, that is, we usually have $r \ll \log_2 N$. In proving a deterministic complexity bound for the algorithm presented in Subsection 3.1, however, we will appeal to the following result.

Proposition 4 (Storjohann [Sto96c]) *Given a positive integer N and k pairs of numbers $(a_1, b_1), (a_2, b_2), \ldots, (a_k, b_k)$ with $\gcd(a_i, b_i) = 1$ for $1 \leq i \leq k$, there exists a deterministic algorithm that computes, in succession for $i = 1, 2, \ldots, k$, the smallest nonnegative integer c_i that satisfies $\gcd(a_i + c_i b_i, N) = 1$. If $|a_i|, |b_i| < N$ for $1 \leq i \leq k$, then the running time of the algorithm is bounded by $O(k \log^2 N + \log^3 N)$ bit operations.*

3.1 The "Conditioning" Subroutine

Let B be an $(k+2) \times 2$ rank 2 input matrix which can be written as

$$B = \begin{bmatrix} N & \bar{N} \\ a_0 & \bar{a}_0 \\ b_1 & \bar{b}_1 \\ \vdots & \vdots \\ b_k & \bar{b}_k \end{bmatrix} \tag{3}$$

with N positive. The goal of this section is to give an algorithm that computes a certain $(k+2) \times (k+2)$ unimodular "conditioning" matrix

$$C = \begin{bmatrix} 1 & & & & \\ & 1 & c_1 & \cdots & c_k \\ & & 1 & & \\ & & & \ddots & \\ & & & & 1 \end{bmatrix} \tag{4}$$

Premultiplying B by C will have the effect of adding certain multiples of rows $3, 4, \ldots, k$ to row 2 of B. In particular, the conditioned matrix CB is given by

$$\overset{\qquad\quad C}{\begin{bmatrix} 1 & & & & \\ & 1 & c_1 & \cdots & c_k \\ & & 1 & & \\ & & & \ddots & \\ & & & & 1 \end{bmatrix}} \overset{\quad B}{\begin{bmatrix} N & \bar{N} \\ a_0 & \bar{a}_0 \\ b_1 & \bar{b}_1 \\ \vdots & \vdots \\ b_k & \bar{b}_k \end{bmatrix}} = \overset{\quad CB}{\begin{bmatrix} N & \bar{N} \\ a_k & \bar{a}_k \\ b_1 & \bar{b}_1 \\ \vdots & \vdots \\ b_k & \bar{b}_k \end{bmatrix}} \quad \text{where} \quad \begin{array}{l} a_k = a_0 + c_1 b_1 + \cdots + c_k b_k \\ \bar{a}_k = \bar{a}_0 + c_1 \bar{b}_1 + \cdots + c_k \bar{b}_k \end{array} \tag{5}$$

and should satisfy the conditions stipulated by the following theorem.

Theorem 5 *Let B be a $(k+2)\times 2$ rank 2 integer input matrix which can be written as in (3) with N positive. There exists a deterministic algorithm which computes a $(k+2)\times(k+2)$ unimodular matrix C which can be written as in (4) and such that the matrix CB, written as in (5), satisfies the following conditions:*

1. *the principal 2×2 submatrix $\begin{bmatrix} N & \bar{N} \\ a_k & \bar{a}_k \end{bmatrix}$ is nonsingular, and*
2. *$\gcd(a_k, N) = \gcd(a_k, b_1, b_2, \ldots, b_k, N)$.*

Furthermore, at most $\lfloor \log_2 N \rfloor + 1$ of the c_i's will be nonzero and all of the c_i's will be bounded in magnitude by $\lfloor 2(\log_2 N)^{3/2} \rfloor + 1$. The cost of the algorithm is bounded by $O(k \log^2 ||B|| + \log^3 ||B||)$ bit operations.

Proof. We assume that $k > 0$ since if $k = 0$ then choosing C to be the 2×2 identity matrix will satisfy the requirements of the theorem. We claim that without loss of generality we may assume the input matrix to have principal 2×2 submatrix nonsingular. If this is not the case, that is, if $N\bar{a}_0 - \bar{N}a_0 = 0$, then it follows from the assumptions $N \neq 0$ and rank $B = 2$ that there there exist a smallest s with $N\bar{b}_s - \bar{N}b_s \neq 0$. Finding this s can be accomplished within $O(k \log^2 ||B||)$ bit operations by assaying the equality $N\bar{b}_i - \bar{N}b_i = 0$ for $i = 1, 2, \ldots, s \leq k$ in succesion. Adding row $2 + s$ to row 2 of B will ensure that the principal 2×2 submatrix of B is nonsingular. After computing the c_i's for the modified input matrix increment c_s by one to account for this intial row transformation. Note that we also need to increase by one both the upper bound we will prove for the number of nonzero c_i's and the magnitudes of the c_i's.

At this point we assume our input matrix B can be written as in (3) and satisfies $N\bar{a}_0 - \bar{N}a_0 \neq 0$. The conditioning algorithm works by computing c_l in succesion for $l = 1, 2, \ldots, k$. For convenience, define the intermediate values $a_l = a + c_1 b_1 + \cdots + c_l b_l$ and $\bar{a}_l = \bar{a} + c_1 \bar{b}_1 + \cdots + c_l \bar{b}_l$. At the end of stage $l - 1$ and start of stage l, the quantities $c_1, c_2, \ldots, c_{l-1}$ have already been computed and the following two conditions

$$\begin{array}{ll} \text{(c1)} & \gcd(a_i, N) = \gcd(a_0, b_1, b_2, \ldots, b_i, N) \\ \text{(c2)} & N\bar{a}_i - \bar{N}a_i \neq 0 \end{array}$$

are satisfied for $i = l - 1$. Note that for the initial case $i = 0$ conditions (c1) and (c2) are trivially satisfied. The goal at stage l is to compute a suitable c_l such that conditions (c1) and (c2) are satisfied for $i = l$. Set $g \leftarrow \gcd(a_{l-1}, b_1)$ and compute integers q_1 and $\hat{a}_{l-1}$ which satisfy $a_{l-1}/g = q_1 N + \hat{a}_{l-1}$ with $0 \leq \hat{a}_{l-1} < N$. Similarily, compute q_2 and $\hat{b}_l$

which satisfy $b_l/g = q_2 N + \hat{b}_l$. with $0 \leq \hat{b}_l < N$. The partially conditioned matrix at the end of stage $l-1$ and start of stage l is given by

$$C^{(l-1)}B = \begin{bmatrix} N & \bar{N} \\ a_{l-1} & \bar{a}_{l-1} \\ b_1 & \bar{b}_1 \\ \vdots & \vdots \\ b_l & \bar{b}_l \\ \vdots & \vdots \\ b_k & \bar{b}_k \end{bmatrix} = \begin{bmatrix} N & \bar{N} \\ g(q_1 N + \hat{a}_{l-1}) & \bar{a}_{l-1} \\ b_1 & \bar{b}_1 \\ \vdots & \vdots \\ g(q_2 N + \hat{b}_l) & \bar{b}_l \\ \vdots & \vdots \\ b_k & \bar{b}_k \end{bmatrix}$$

where $\hat{a}_{l-1}$ and $\hat{b}_l$ are relatively prime and bounded in magnitude by N. It follows from Theorem 3 that the smallest nonnegative t which satisfies

$$\gcd(\hat{a}_{l-1} + t\hat{b}_l, N) = 1 \tag{6}$$

will be bounded in magnitude by $2(\log_2 N)^{3/2}$. We claim that choosing $c_l \leftarrow t$ where t satisfies (6) will cause $(c1)$ to be satisfied for $i = l$. To see this, note that

$$\begin{aligned} \gcd(a_l, N) &= \gcd(a_{l-1} + tb_l, N) \\ &= \gcd(g(q_1 N + \hat{a}_{l-1}) + tg(q_2 N + \hat{b}_l), N) \\ &= \gcd(g(\hat{a}_{l-1} + t\hat{b}_l) + g(q_1 + q_2)N, N) \\ &= \gcd(g(\hat{a}_{l-1} + t\hat{b}_l), N) \\ &= \gcd(g, N) \\ &= \gcd(a_{l-1}, b_l, N) \\ &= \gcd(a_0, b_1, b_2, \ldots, b_l, N) \end{aligned}$$

where the last equality follows from the fact that (c1) is satisfied for $i = l-1$. There is one subtlety that we must take care for. At stage l there may exist some particular real number x such that a choice of $c_l = x$ causes condition (c2) to be not satisfied for $i = l$, that is, there may exist an x for which

$$\begin{vmatrix} N & \bar{N} \\ a_{l-1} + xb_l & \bar{a}_{l-1} + x\bar{b}_l \end{vmatrix} = 0. \tag{7}$$

In case $N\bar{b}_l - \bar{N}b_l = 0$ then there does not exist an x which satisfies (7). Otherwise, there exists a unique value

$$x = -\frac{N\bar{a}_{l-1} - \bar{N}a_{l-1}}{N\bar{b}_l - \bar{N}b_l} \tag{8}$$

which satisfies equation (7). The case where x of (8) exists and is a positive integer is handled by setting c_l to be the *negative* of the smallest nonnegative integer t which satisfies

$$\gcd(\hat{a_l} + t(-\hat{b_l}), N) = 1.$$

We now turn our attention to bounding the running time. Note that $\gcd(a_i, N)$ is a divisor of $\gcd(a_{i-1}, N)$ for $i = 1, 2, \ldots, k$ and if $\gcd(a_i, N) = \gcd(a_{i-1}, N)$ then c_i will have been chosen to be zero. There can be at most $\lfloor \log_2 N \rfloor$ distinct choices for i with $\gcd(a_i, N)$ a proper divisor of $\gcd(a_{i-1}, N)$. Here we are using the gross upper bound $\lfloor \log_2 N \rfloor$ for the number of prime divisors (not necessarily distinct) in the prime factorization of N.

This shows that at most $\lfloor \log_2 N \rfloor$ of the c_i's will be nonzero and is also sufficient to show that the a_i's are all bounded in length by $O(\log ||B||)$ bits. To establish the running time, note the computation of the a_i's, $\hat{a}_i$'s, $\hat{b}_i$'s and the "bad" values of (8) will be bounded by $O(k \log^2 ||B||)$ bit operations. Proposition 4 bounds the cost of computing the c_i's by $O(k \log^2 N + \log^3 N)$ bit operatons. ■

3.2 The "Column Reduction" Subroutine

Let B be an $n \times 2$ input matrix which can be written as

$$B = \begin{bmatrix} * & * \\ \vdots & \vdots \\ * & * \\ N & \bar{N} \\ a_0 & \bar{a}_0 \\ b_1 & \bar{b}_1 \\ \vdots & \vdots \\ b_k & \bar{b}_k \end{bmatrix} \tag{9}$$

with $k \geq 0$, N positive, and trailing $(k+2) \times 2$ submatrix of rank 2. The goal of this section is to give an algorithm which computes $n \times n$ unimodular matrices

$$C = \begin{bmatrix} 1 & & & & & & & & \\ & \ddots & & & & & & & \\ & & 1 & & & & & & \\ & & & 1 & * & * & \cdots & * & \\ & & & & 1 & & & & \\ & & & & & 1 & & & \\ & & & & & & \ddots & & \\ & & & & & & & & 1 \end{bmatrix} \quad \text{and} \quad Q = \begin{bmatrix} 1 & & & & * & * & & & \\ & \ddots & & & \vdots & \vdots & & & \\ & & 1 & & * & * & & & \\ & & & & * & * & & & \\ & & & & * & * & & & \\ & & & & * & * & 1 & & \\ & & & & \vdots & \vdots & & \ddots & \\ & & & & * & * & & & 1 \end{bmatrix} \tag{10}$$

where the only nonzero off-diagonal entries in C occur to the right of the $(n-k-1)$-th diagonal entry and the only off-diagonal entries in Q occuring in columns $n-k-1$ and $n-k$. Q and C should be such that the matrix QCB can be written as

$$QCB = \begin{bmatrix} * & * \\ \vdots & \vdots \\ * & * \\ t_1 & * \\ & t_2 \\ & * \\ & \vdots \\ & * \end{bmatrix} \tag{11}$$

and satisfies the conditions stipulated by the following theorem.

Theorem 6 *Let B be an $n \times 2$ integer input matrix which can be written as in (9) with $k \geq 0$, N positive, and trailing $(k+2) \times 2$ submatrix of rank 2. There exists a deterministic*

algorithm `ColumnReduction`(B, k) *that takes as input* B *and* k*, and returns as output* $n \times n$ *unimodular matrices* C *and* Q *which can be written as in (10) and such that the matrix* QCB*, written as in (11), satisfies the following conditions:*

1. *the entries* t_1 *and* t_2 *occur in rows* $n-k-1$ *and* $n-k$ *respecitvely and are positive,*
2. *entries above* t_1 *in column 1 are nonnegative and bounded in magnitude by* $t_1 - 1$*, and*
3. *entries above and below* t_2 *in column 2 are nonnegative and bounded in magnitude by* $t_2 - 1$*.*

Furthermore, $||Q|| \leq O(||B||)$*,* $||C|| \leq \lfloor 2(\log_2 N)^{3/2} \rfloor + 1$ *and at most* $\lfloor \log_2 N \rfloor + 1$ *of the off-diagonal entries in* C *will be nonzero. The cost of the algorithm is bounded by* $O(n \log^2 ||B|| + \log^3 ||B||)$ *bit operations.*

Proof. Apply the algorithm of Theorem 5 to the submatrix of B comprised of the last $k+2$ rows and embed the $(k+2) \times (k+2)$ output matrix into the $n \times n$ identity matrix to produce a suitable C which satisfies the requirements of the Theorem. We then have

$$B' = CB = \begin{bmatrix} * & * \\ \vdots & \vdots \\ * & * \\ N & \bar{N} \\ a_k & \bar{a}_k \\ b_1 & \bar{b}_1 \\ \vdots & \vdots \\ b_k & \bar{b}_k \end{bmatrix}$$

with $\gcd(a_k, N) = \gcd(a_k, b_1, b_2, \ldots, b_k, N)$ and submatrix comprised of rows $n-k-2$ and $n-k-1$ nonsingular. Compute a solution (m_1, m_2, t_1) to the extended gcd problem $m_1 N + m_2 a_0 = t_1$ with $t_1 = \gcd(N, a_0)$ and initialize Q to be the $n \times n$ identity matrix except with $Q_{n-k-1,n-k-1} = m_1$, $Q_{n-k-1,n-k} = m_2$, $Q_{n-k,n-k-1} = -sa_0/t1$ and $Q_{n-k,n-k} = sN/t_1$ where s is chosen to be either 1 or -1 so as to ensure that the quantity $t_2 = (-sa_0/t_1)\bar{N} + (sN/t_1)\bar{a}_k$ is positive. At this point we have

$$\overset{Q}{\begin{bmatrix} 1 & & & & & & & \\ & \ddots & & & & & & \\ & & 1 & & & & & \\ & & & * & * & & & \\ & & & * & * & & & \\ & & & & & 1 & & \\ & & & & & & \ddots & \\ & & & & & & & 1 \end{bmatrix}} \overset{CB}{\begin{bmatrix} * & * \\ \vdots & \vdots \\ * & * \\ N & \bar{N} \\ a_k & \bar{a}_k \\ b_1 & \bar{b}_1 \\ \vdots & \vdots \\ b_k & \bar{b}_k \end{bmatrix}} = \overset{B'}{\begin{bmatrix} * & * \\ \vdots & \vdots \\ * & * \\ t_1 & * \\ & t_2 \\ b_1 & \bar{b}_1 \\ \vdots & \vdots \\ b_k & \bar{b}_k \end{bmatrix}}$$

where Q is unimodular. Using elementary row operations (applied to Q as well as B') first reduce the entries above and below entry t_1 in column 1 of B' modulo t_1 then reduce entries above and below entry t_2. The output matrix Q will satisfy the requirements of the Theorem. Theorem 5 bounds the cost of producing C by $O(n \log^2 ||B|| + \log^3 ||B||)$ bit operations. The cost of producing Q from CB is easily seen to be bounded by $O(n \log^2 ||B||)$ bit operations since $||CB|| = O(||B||)$. ■

4 The Row Reduced Echelon Form Algorithm

In this Section we present our algorithm for computing the row reduced echelon form with premultiplier for an $n' \times m'$ input matrix A'. As promised, the algorithm directly handles the case of an input matrix with arbitrary shape and rank profile. A number of special cases occur when the rank r' of A' is either 0, n and/or m. To handle all of these cases in one fell swoop, we embed the original input matrix A' into a larger $n \times m$ matrix A that can be written in block form as

$$A = \left[\begin{array}{c|c|c} I_1 & & \\ \hline & A' & \\ \hline & & I_1 \end{array}\right], \tag{12}$$

where A' is $(n-2) \times (m-2)$. In other words, the first row and column of A are all zero except for a 1 the leading entry, and trailing row and column are all zero except for a 1 in the trailing entry. Such an input matrix A will have rank profile $[j_1, j_2, \ldots, j_r]$ with $r \geq 2$, $j_1 = 1$ and $j_r = m$. The algorithm computes matrices Q, C and T which satisfy $QCA = T$ with Q and C unimodular and T having first $m-1$ columns in row reduced echelon form. We show later how to recover the row reduced echelon form with transforming matrix for A' from Q, C and T. For clarity, we present the algorithm in a form that takes as input the rank profile $[j_1, j_2, \ldots, j_r]$ — we show later how to compute the rank profile during as the reduction proceeds.

Algorithm: `RowReducedEchelonForm`
Input: An $n \times m$ integral matrix A with rank profile $[j_1, j_2, \ldots, j_r]$ which can be written as in (12).
Output: Matrices Q, C and T satisfying $QCA = T$ with Q and C unimodular and T having first $m-1$ columns in row reduced echelon form.

(1) [Initialization:]
  $Q \leftarrow I_n$;
  $C \leftarrow I_n$;
  $T \leftarrow A$;
  Remark: Let $(Q^{(0)}, C^{(0)}, T^{(0)})$ denote the state of (Q, C, T) at this point.
(2) [Reduction:]
  for $k = 1$ to $r-1$ do
    $B_k \leftarrow$ the $n \times 2$ matrix $[\mathrm{col}(T, j_k) \mid \mathrm{col}(T, j_{k+1})]$
    $(Q_k, C_k) \leftarrow$ `ColumnReduction`$(B_k, n-k-1)$;
    $Q \leftarrow Q_k C_k Q C_k^{-1}$;
    $C \leftarrow C_k C$;
    $T \leftarrow Q_k C_k T$;
    Remark: Let $(Q^{(k)}, C^{(k)}, T^{(k)})$ denote the state of (Q, C, T) at this point.
(4) [Output:]
  return (Q, C, T) and quit;

The remainder of this section is divided into four subsections. In Subsection 4.1 we prove that Algorithm `RowReducedEchelonForm` is correct. In Subsection 4.2 we prove that

intermediate integers occurring during the algorithm don't get too large. In Subsection 4.3 we analyse of the running time of the algorithm and show how to dispense with the need to know *a priori* the rank profile of the input matrix. In Subsection 4.4 we give a modification of the algorithm which performs integer arithmetic in a residue number system and also show how to recover a single unimodular transforming matrix U from Q and C.

Before proceeding with the formal proof of correctness, we give an explicit example of how the reduction proceeds for a 6×7 input matrix A' with rank 5 and rank profile $[1,3,4,6,7]$. In what follows we denote a possibly nonzero entry of a matrix by $*$ and a necessarily zero entry by a blank. Our input matrix A of algorithm `RowReducedEchelonForm` will have the form

$$A = \left[\begin{array}{c|ccccccc|c} 1 & & & & & & & & \\ \hline & * & * & * & * & * & * & * & \\ & * & * & * & * & * & * & * & \\ & * & * & * & * & * & * & * & \\ & * & * & * & * & * & * & * & \\ & * & * & * & * & * & * & * & \\ & * & * & * & * & * & * & * & \\ \hline & & & & & & & & 1 \end{array}\right]$$

where the center block is our original input matrix A'. Note that the rank profile of A will be $[1,2,4,5,7,8,9]$. In what follows we denote by $\bar{*}$ or $\bar{h}_{*j}$ an entry that is nonnegative and reduced modulo N_j or h_j respectively. Note that $N_1 = h_1 = 1$. The intermediate matrices $Q^{(0)}, C^{(0)}$ and $T^{(0)}$ satisfy

$$\left[\begin{array}{c|ccccccc} 1 & & & & & & & \\ \hline & 1 & & & & & & \\ & & 1 & & & & & \\ & & & 1 & & & & \\ & & & & 1 & & & \\ & & & & & 1 & & \\ & & & & & & 1 & \\ & & & & & & & 1 \end{array}\right] \quad Q^{(0)}$$

$$\left[\begin{array}{c|ccccccc} 1 & & & & & & & \\ \hline & 1 & & & & & & \\ & & 1 & & & & & \\ & & & 1 & & & & \\ & & & & 1 & & & \\ & & & & & 1 & & \\ & & & & & & 1 & \\ & & & & & & & 1 \end{array}\right] \quad C^{(0)}$$

$$A = \left[\begin{array}{c|ccccccc} N_1 & & & & & & & 1 \\ \hline & * & * & * & * & * & * & * & \\ & * & * & * & * & * & * & * & \\ & * & * & * & * & * & * & * & \\ & * & * & * & * & * & * & * & \\ & * & * & * & * & * & * & * & \\ & * & * & * & * & * & * & * & \\ & & & & & & & & 1 \end{array}\right] \quad T^{(0)}$$

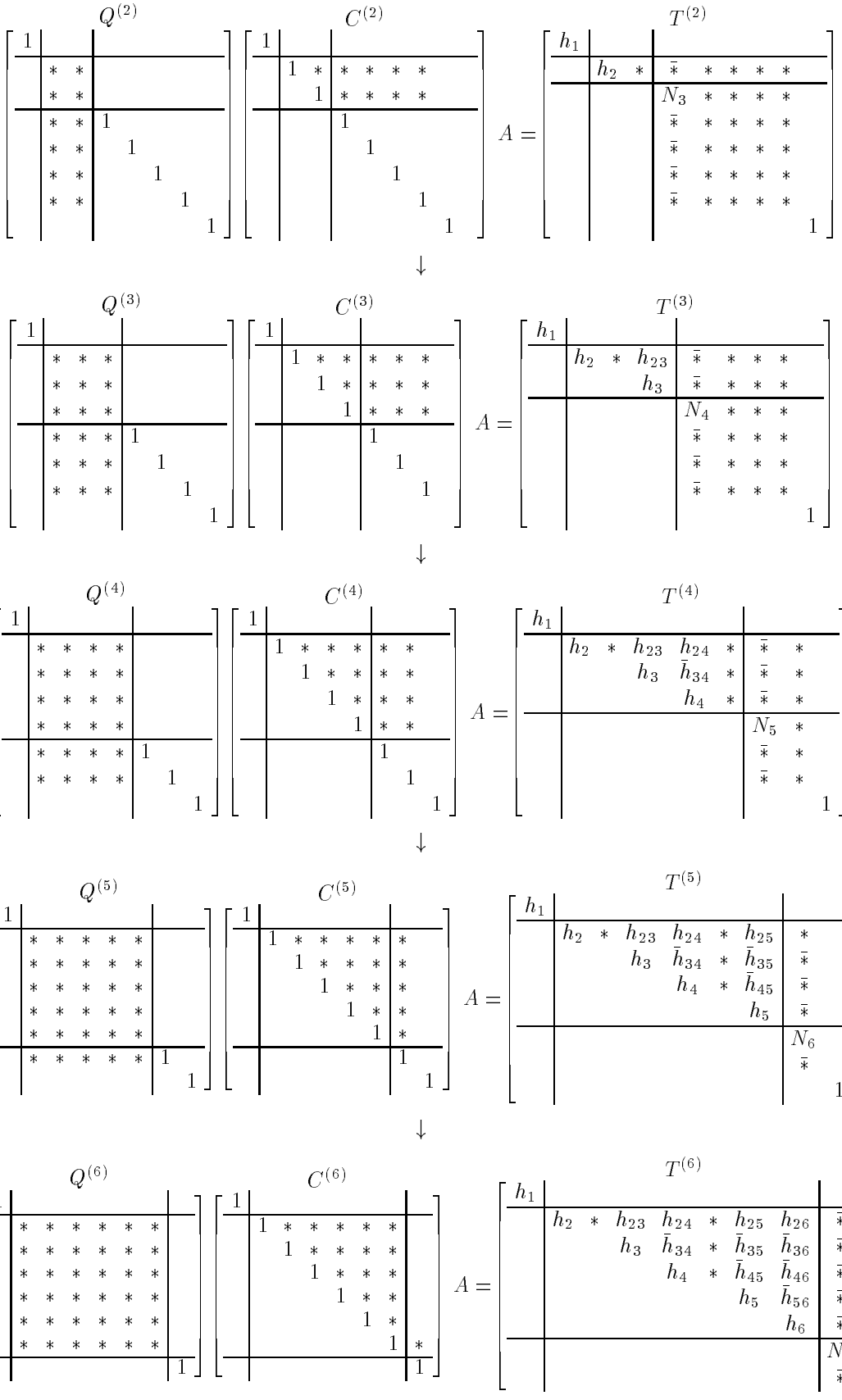
$$\downarrow$$

$$\left[\begin{array}{c|cc|cccccc} N_1 & & & & & & & & \\ \hline & * & & & & & & & \\ \hline & * & 1 & & & & & & \\ & * & & 1 & & & & & \\ & * & & & 1 & & & & \\ & * & & & & 1 & & & \\ & * & & & & & 1 & & \\ & * & & & & & & 1 & \\ & & & & & & & & 1 \end{array}\right] \quad Q^{(1)}$$

$$\left[\begin{array}{c|cc|cccccc} 1 & & & & & & & & \\ \hline & 1 & & * & * & * & * & * & \\ \hline & & 1 & & & & & & \\ & & & 1 & & & & & \\ & & & & 1 & & & & \\ & & & & & 1 & & & \\ & & & & & & 1 & & \\ & & & & & & & 1 & \\ & & & & & & & & 1 \end{array}\right] \quad C^{(1)}$$

$$A = \left[\begin{array}{c|ccccccc} h_1 & & & & & & & & \\ \hline & N_2 & * & * & * & * & * & * & \\ & \bar{*} & * & * & * & * & * & * & \\ & \bar{*} & * & * & * & * & * & * & \\ & \bar{*} & * & * & * & * & * & * & \\ & \bar{*} & * & * & * & * & * & * & \\ & \bar{*} & * & * & * & * & * & * & \\ & & & & & & & & 1 \end{array}\right] \quad T^{(1)}$$

$$\downarrow$$


$Q^{(2)}$
$C^{(2)}$
$A =$
$T^{(2)}$
$Q^{(3)}$
$C^{(3)}$
$A =$
$T^{(3)}$
$Q^{(4)}$
$C^{(4)}$
$A =$
$T^{(4)}$
$Q^{(5)}$
$C^{(5)}$
$A =$
$T^{(5)}$
$Q^{(6)}$
$C^{(6)}$
$A =$
$T^{(6)}$

4.1 Proof of Correctness

In what follows we adopt the convention that a submatrix of row and/or column dimension less than 1 simply comprises no entries of a matrix — the statement that such a submatrix is in row reduced echelon form will be vacuously true. Our first result establishes that that the intermediate matrices $(Q^{(i)}, C^{(i)}, T^{(i)})$ can be written in block form as and satisfy

$$\overset{Q^{(i)}}{\left[\begin{array}{c|c|c} I_1 & & \\ \hline & * & \\ \hline & * & I_{n-i-1} \end{array}\right]} \overset{C^{(i)}}{\left[\begin{array}{c|c|c} I_1 & & \\ \hline & * & * \\ \hline & & I_{n-i-1} \end{array}\right]} A = \overset{T^{(i)}}{\left[\begin{array}{c|c|c} I_1 & & \\ \hline & R_i & * \\ \hline & & * \end{array}\right]} \tag{13}$$

for $i = 0, 1, \ldots, r-1$ where R_i is $(i-1) \times (j_{i+1} - 2)$ and in row reduced echelon form.

Lemma 7 *For $i = 0, 1, \ldots, r-1$ the matrices $(Q^{(i)}, C^{(i)}, T^{(i)})$ can be written as in and satisfy (13) with R_i of size $(i-1) \times (j_{i+1} - 2)$ and in row reduced echelon form. Furthermore*

1. *$Q^{(i)}$ and $C^{(i)}$ will be unimodular,*
2. *the entry $T^{(i)}_{i+1,j_{i+1}}$, which we denote by N_{i+1}, will be nonzero and $0 \leq T_{l,j_{i+1}} < N_{i+1}$ for $l = 1, 2, \ldots, i, i+2, i+3, \ldots, m$.*

Proof. We prove the lemma by induction on i. The initial case $i = 0$ is trivially satisfied. By induction, assume the lemma holds for $i = k - 1 < r$ for some k with $k > 0$. We need to prove the lemma holds for $i = k$. The $n \times 2$ submatrix B_k comprised of columns j_k and j_{k+1} of $T^{(k-1)}$ can be written as

$$B_k = \begin{bmatrix} \bar{*} & * \\ \vdots & \vdots \\ \bar{*} & * \\ N_k & \bar{N}_k \\ a_0 & \bar{a}_0 \\ b_1 & \bar{b}_1 \\ \vdots & \vdots \\ b_{k'} & \bar{b}_{k'} \end{bmatrix} \tag{14}$$

with $k' = n - k - 1$. By the induction hypothesis the lemma is correct for $i = k - 1$. It follows that B_k will satisfy the following two properties: (a) $N_k \neq 0$ since N_k equals $T^{(k-1)}_{k,j_k}$, and (b) the submatrix comprised of the last $k' + 2$ rows of B_k will be 2. Property (b) follows from the fact that B_k is comprised of columns j_k and j_{k+1} of $T^{(k-1)}$ and that all entries to the left of column j_k and in rows $k, k+1, \ldots, m$ of $T^{(k-1)}$ are zero, that is, entries below the block R_i (where $i = k - 1$) in the matrix $T^{(i)}$ of (13) are all zero. The important point here is that properties (a) and (b) ensure that B_k is a valid input matrix for subroutine `ColumnReduction`. The matrices C_k and Q_k produced by `ColumnReduction` will be unimodular, and because of the particular structure for C_k and Q_k guaranteed by Theorem 6, the matrices $Q^{(k)} \leftarrow Q_k C_k Q^{(k-1)} C_k^{-1}$ and $C^{(k)} \leftarrow C_k C^{(k-1)}$ computed in step (2.) can be written as in (13). We need to show that (13) is satisfied for $i = k$ at the end of step (2), that is, that the identity $T^{(k)} = Q^{(k)} C^{(k)} A$ holds. Note that the matrix $T^{(k)}$

is computed as $T^{(k)} \leftarrow Q_k C_k T^{(k-1)}$. Noting that the equality $T^{(k-1)} = Q^{(k-1)} C^{(k-1)} A$ holds by the induction hypothesis, we get

$$\begin{aligned} T^{(k)} &= Q_k C_k T^{(k-1)} \\ &= Q_k C_k (Q^{(k-1)} C^{(k-1)} A) \\ &= Q_k C_k (Q^{(k-1)} (C_k^{-1} C_k) C^{(k-1)} A) \\ &= (Q_k C_k Q^{(k-1)} C_k^{-1})(C_k C^{(k-1)}) A) \\ &= Q^{(k)} C^{(k)} A \end{aligned}$$

which shows that (13) is satisfied for $i = k$. Finally, by the structure for $Q_k C_k B_k$ guaranteed by Theorem 6, and by the fact that the submatrix comprised of rows $k, k+1, \ldots, m$ and columns $j_k, j_k + 1, \ldots, j_{k+1} - 1$ of $T^{(k-1)}$ has rank 1, the matrix $T^{(k)} = Q_k C_k T^{(k-1)}$ can be written as in (13) and will satisfy the requirements of the lemma. ■

Algorithm `RowReducedEchelonForm` requires an arbitrary input matrix A' of size $n' \times m'$ to be embedded into an $(n'+2) \times (m'+2)$ matrix A as in (12). We now show that this is no essential restriction since the output matrix $(Q^{(r-1)}, C^{(r-1)}, T^{(r-1)})$ can be written as and satisfy

$$\begin{array}{ccccccc} Q^{(r-1)} & & C^{(r-1)} & & A & & T^{(r-1)} \\ \left[\begin{array}{c|c|c} I_1 & & \\ \hline & Q' & \\ \hline & * & I_1 \end{array}\right] & \left[\begin{array}{c|c|c} I_1 & & \\ \hline & C' & * \\ \hline & & I_1 \end{array}\right] & \left[\begin{array}{c|c|c} I_1 & & \\ \hline & A' & \\ \hline & & I_1 \end{array}\right] & = & \left[\begin{array}{c|c|c} I_1 & & \\ \hline & R_{r-1} & * \\ \hline & & N_r \end{array}\right] \end{array} \qquad (15)$$

Theorem 8 *Algorithm* `RowReducedEchelonForm` *is correct. The output matrix* $(Q^{(r-1)}, C^{(r-1)}, T^{(r-1)})$ *can be written as in (15). In particular,* Q' *and* C' *are unimodular and satisfy* $Q'C'A' = R_{r-1}$ *where* R_{r-1} *is the row reduced echelon form of* A'.

Proof. The fact that equation (15) holds follows directly from Theorem 7 (i.e. (13) holds for $i = r - 1$). It follows from the structure of $C^{(r-1)}$, $Q^{(r-1)}$ and A' that Q' and C' are unimodular and satisfy $Q'C'A' = R_{r-1}$. ■

4.2 A Bound on the Intermediate Integers

The goal of this section is to prove

Theorem 9 *The intermediate matrices* $C^{(i)}$, $Q^{(i)}$ *and* $T^{(i)}$ *satisfy*

$$\log ||C^{(i)}|| = O(r + \log\log ||A||) \quad \textit{and} \quad \log ||Q^{(i)}||, \log ||T^{(i)}|| = O(r \log r ||A||)$$

for $i = 0, 1, \ldots, r - 1$.

The proof of Theorem 9 will require a number of intermediate results. In what follows, we write $(CA)^{(i)}$ to denote the matrix $C^{(i)}A$ for $0 \leq i \leq r - 1$ and, as in Lemma 7, N_{i+1} will denote the entry $T^{(i)}_{i+1, j_{i+1}}$. An $i \times i$ minor of a matrix is the determinant of an $i \times i$ submatrix of A. By convention, the 0×0 minor is 1.

Lemma 10 *For* $i = 0, 1, \ldots, r - 1$, N_{i+1} *is bounded in magnitude by the absolute value of an* $(i+1) \times (i+1)$ *minor of* $(CA)^{(i)}$ *and entries of* $Q^{(i)}$ *are bounded in magnitude by the absolute values of* $i \times i$ *minors of* $(CA)^{(i)}$.

Proof. By Lemma 7 we can write $Q^{(i)}$ as

$$Q^{(i)} = \left[\begin{array}{c|c|c} I_1 & & \\ \hline & \vec{q}_1 & \\ & \vdots & \\ & \vec{q}_i & \\ \hline & \vec{q}_{i+1} & \\ & \vdots & I_{n-i-1} \\ & \vec{q}_n & \end{array}\right] \tag{16}$$

where the $\vec{q}_j$'s are i dimensional row vectors. Note that the center block of the matrix in (16) must be unimodular since $Q^{(i)}$ is unimodular. Choose some j with $i+1 \leq j \leq n$. Our approach is to prove the lemma for those entries of $Q^{(i)}$ limited to the entries in $\{\vec{q}_1, , \ldots, \vec{q}_i, \vec{q}_j\}$. The result for all entries in $Q^{(i)}$ will follow by letting $j = i+1, i+2, \ldots, n$. Let B be the submatrix of $(CA)^{(i)}$ comprised of columns $j_1, j_2, \ldots, j_{i+1}$. Then $Q^{(i)}B$ is the submatrix of $T^{(i)}$ comprised of columns $j_1, j_2, \ldots, j_{i+1}$ and by the structure for $T^{(i)}$ guaranteed by Lemma 7 we have

$$Q^{(i)}B = \left[\begin{array}{c|ccccc|c} I_1 & & & & & & \\ \hline & h_2 & \bar{h}_{2\,2} & & \bar{h}_{2\,i} & & \bar{*} \\ & & h_3 & \cdots & \bar{h}_{3\,i} & & \bar{*} \\ & & & \ddots & \vdots & & \vdots \\ & & & & h_i & & \bar{*} \\ & & & & & & N_{i+1} \\ \hline & & & & & & \bar{*} \\ & & & & & & \vdots \\ & & & & & & \bar{*} \end{array}\right]$$

where the first i columns are the Hermite normal form of the first i columns of B. Furthermore, N_{i+1} is positive and entries above and below N_{i+1} in column $i+1$ are nonnegative and less than N_{i+1}. In particular, the principal $(i+1) \times (i+1)$ submatrix of $Q^{(i)}B$ will be in Hermite normal form. Let P_j be the $n \times n$ identity matrix except with a 1 in row 1 column j. Premultiplying a matrix by P_j has the effect of adding row j to row 1. Then

$$(P_j Q^{(i)} P_j^{-1})(P_j B) = \left[\begin{array}{c|ccccc|c} I_1 & & & & & & \bar{*} \\ \hline & h_2 & \bar{h}_{2\,2} & & \bar{h}_{2\,i} & & \bar{*} \\ & & h_3 & \cdots & \bar{h}_{3\,i} & & \bar{*} \\ & & & \ddots & \vdots & & \vdots \\ & & & & h_i & & \bar{*} \\ & & & & & & N_{i+1} \\ \hline & & & & & & \bar{*} \\ & & & & & & \vdots \\ & & & & & & \bar{*} \end{array}\right] \tag{17}$$

where the principal $(i+1) \times (i+1)$ submatrix is in Hermite normal form. Because the matrix $P_j Q^{(i)} P_j^{-1}$ has only zero entries to the right of the principal $(i+1) \times (i+1)$ block, equation (17) will still hold if we restrict the matrix on the right hand side and the matrices $(PQ^{(i)}P^{-1}$ and (PB) to their principal $(i+1) \times (i+1)$ submatrices. Let U be the principal

$(i+1) \times (i+1)$ submatrix of $P_j Q^{(i)} P_j^{-1}$, B' be the principal $(i+1) \times (i+1)$ submatrix of $P_j B$, and H be the principal $(i+1) \times (i+1)$ submatrix of the right hand side of (17). Then

$$\left[\begin{array}{c|c} I_1 & \vec{q}_j \\ \hline & \vec{q}_1 \\ & \vdots \\ & \vec{q}_i \end{array}\right]^{U} B' = \left[\begin{array}{c|ccccc} I_1 & & & & & \bar{*} \\ \hline & h_2 & h_{2\,2} & & h_{2\,i} & \bar{*} \\ & & h_3 & \cdots & \bar{h}_{3\,i} & \bar{*} \\ & & & \ddots & \vdots & \vdots \\ & & & & h_i & \bar{*} \\ & & & & & N_{i+1} \end{array}\right]^{H}$$

where U is unimodular and H is in Hermite normal form. Theorem 2 now bounds the size of entries in U by the magnitudes of entries in the adjoint of B'. Recall that entries in the adjoint of the $(i+1) \times (i+1)$ matrix B' will be, up to sign, $i \times i$ minors of B'. The result now follows by noting that B' is a submatrix of $P_j B$, which, because of the special structure of the first row and column of B, will have the same set, up to sign, of $i \times i$ minors as that of B — a submatrix of $(CA)^{(i)}$. ■

The correctness of Theorem 9 will follow directly from the following result.

Lemma 11 *Let A be a valid input matrix for algorithm* `RowReducedEchelonForm` *and let* $\Delta = \max(r^{1/2}||A||, 100)$. *Then*

1. $||C^{(i)}|| < (20r \log_2 \Delta)^{3/2}$ *and each row of* $C^{(i)}$ *contains less than* $20r \log_2 \Delta$ *nonzero entries.*
2. $||Q^{(i)}|| \leq \Delta^{9r}$
3. $||T^{(i)}|| \leq r||A||(20r \log_2 \Delta)^{5/2} \Delta^{9r}$

for $i = 0, 1, \ldots, r-1$.

Proof. We prove the lemma using induction on i. Clearly (1.), (2.) and (3.) hold for $i = 0$ since $||C^{(0)}|| = 1$ and $Q^{(0)}$ is the identity matrix. Assume that (1.), (2.) and (3.) hold for $i = 0, 1, \ldots, k$ for some k with $0 \leq k < r-1$. We need to prove that (1.), (2.) and (3.) hold for $i = k+1$. We first show that (1.) holds for $i = k+1$. By construction, $C^{(k+1)} = C_{k+1} C^{(k)}$ and from the structure of $C^{(k)}$ and C_{k+1} it follows that $C^{(k+1)}$ will be identical to $C^{(k)}$ except with row $k+2$ replaced by row $k+2$ of C_{k+1}. Thus, to show that (1.) holds for $i = k+1$ it will be sufficient to show that the bounds claimed by (1.) hold for the matrix C_{k+1}; we claim this will follow if we can show that

$$|N_{k+1}| \leq \Delta^{9r}. \tag{18}$$

To see this, note that Theorem 6 bounds the magnitudes of entries in C_{k+1} by $\lfloor 2(\log_2 N_{k+1})^{3/2} \rfloor + 1$ which, if (18) holds, is less than $(20r \log_2 \Delta)^{3/2}$. Similarly, Theorem 6 bounds the number of off-diagonal entries in row $k+2$ of C_{k+2} by $\lfloor \log_2 N_{k+1} \rfloor + 1$ which, if (18) holds, leads to the bound $20r \log_2 \Delta$ for the number of nonzero entries in row $k+2$. We now prove that (18) holds. By the induction hypothesis, (1.) holds for $i = k$ and we have

$$\begin{aligned} ||(CA)^{(k)}|| &< (20r \log_2 \Delta) \cdot ||A|| \cdot ||C^{(k)}|| \\ &< ||A|| \cdot (20r \log_2 \Delta)^{5/2} \end{aligned} \tag{19}$$

By Lemma 10, N_{k+1} is a factor of a $(k+1)\times(k+1)$ minor of $(CA)^{(k)}$. Applying Hadamard's bound (Theorem 1) together with (19) and the fact that $k+1 \leq r$ yields

$$\begin{aligned}
|N_{k+1}| &\leq ((k+1)^{1/2}||(CA)^{(k)}||)^{k+1} \\
&\leq ((k+1)^{1/2}||A||(20r\log_2\Delta)^{5/2})^{k+1} \\
&\leq (r^{1/2}||A||(20r\log_2\Delta)^{5/2})^{r} \\
&= (\Delta(20r\log_2\Delta)^{5/2})^{r} \\
&= \Delta^r\cdot((20r\log_2\Delta)^{5/2})^{r} \\
&= \Delta^{r\left(1+\frac{5}{2}\frac{\log(20r\log_2\Delta)}{\log\Delta}\right)} \\
&= \Delta^{r\left(1+\frac{5}{2}\left(\frac{\log(20\log_2\Delta)}{\log\Delta}+\frac{2\log r}{\log r+2\log||A||}\right)\right)} \\
&\leq \Delta^{r\left(1+\frac{5}{2}\left(\frac{\log(20\log_2\Delta)}{\log\Delta}+\frac{2\log r}{\log r}\right)\right)} \\
&< \Delta^{r\left(1+\frac{5}{2}\left(\frac{\log(20\log_2 100)}{\log 100}+2\right)\right)} \\
&< \Delta^{r(1+7.66)} \\
&< \Delta^{9r}
\end{aligned}$$

which proves (18). This shows that (1.) holds for $i = k+1$. We now show that (2.) and (3.) hold for $i = k+1$. By Lemma 10 and Hadamard's bound we have

$$||Q^{(k+1)}|| \leq ((k+1)^{1/2}||(CA)^{(k+1)}||)^{k+1} \tag{20}$$

and noting that $Q^{(k+1)}$ has at most $k+2$ nonzero entries in each row, we get

$$||T^{(k+1)}|| \leq (k+2)\cdot||(CA)^{(k+1)}||\cdot||Q^{(k+1)}|| \tag{21}$$

Substituting (19) into (20) simplifying shows that (2.) holds for $i = k+1$. (Note that the simplification is identical to that used to prove the bound for $|N_{k+1}|$.) Substituting (20) and (19) into (21) shows that (3.) holds for $i = k+1$. ■

4.3 A Bound on the Running Time

First we determining the running time of algorithm `RowReducedEchelonForm` in it's unmodified form.

Lemma 12 *Algorithm* `RowReducedEchelonForm` *requires* $O(nr^3\log^2 r||A|| + r^4\log^3 r||A||)$ *bit operations for calls to subroutine* `ColumnReduction` *plus additional cost* $O(nmr)$ *additions and multiplications with integers bounded in length by* $O(r\log r||A||)$ *bits.*

Proof. Theorems 6 and 9 bound the lengths of integer entries in all intermediate matrices $Q^{(k)}$, $C^{(k)}$, $T^{(k)}$, B_k, Q_k by $O(r\log r||A||)$ bits. Theorem 6 bounds the cost of a single call to `ColumnReduction` by $O(nr^2\log^2 r||A|| + r^3\log^3 r||A||)$ bit operations; the claimed total cost of all calls follows by by noting that `ColumnReduction` is called exactly $r-1$ times. The additional cost is to update the work matrices (Q, C, T) for each iteration of the loop in step (2.) according to the update formulae $(Q, C, T) \leftarrow (Q_kC_kQC_k^{-1}, C_kC, Q_kC_kT)$. It follows from the structure of Q, C, T, Q_k and C_k that a single update requires $O(nm)$ additions and multiplications with integers bounded in length by $O(r\log r||A||)$ bits. In particular, note that to compute $C_kQC_k^{-1}$ we need only compute the first k columns of C_kQ. ■

We now show how to dispense with the need to *a priori* the rank profile of the input matrix.

Corollary 13 *There exists a modification of algorithm* `RowReducedEchelonForm` *that doesn't require as input the rank profile of the input matrix. The running time is the same as that claimed by Lemma 12.*

Proof. Note that j_1 is known to be 1 because of the special structure of the input matrix. At the start of the loop in step (2.) with index $k = 1$, the index j_1 is known and j_2 needs to be determined. In general, at the start the loop in step (2.) the index j_k will be known and j_{k+1} will need to be computed. The work matrix T, which is $T^{(k-1)}$ at this point, can be written as

$$T = \left[\begin{array}{c|c|c} I_1 & & \\ \hline & R_{k-1} & * \\ \hline & & T' \end{array}\right]$$

where the principal entry in the block T' is the positive quantity $N_k = T_{k,j_k}$. The unknown index j_{k+1} will satisfy $j_k < j_{k+1} \leq m$ and is equal to the smallest integer j such that the submatrix of T comprised rows $k, k+1, \ldots, n$ and columns j_k and j has rank 2. The following code fragment finds j in $O(n(j_{k+1} - j_k))$ arithmetic operations.

1. $N \leftarrow T_{k,j_k}$;
2. for $j = j_k + 1$ to m do
3. $\quad \bar{N} \leftarrow T_{k,j}$;
4. $\quad$ for $i = k + 1$ to n do
5 $\quad\quad$ if $NT_{i,j} - \bar{N}T_{i,j_k} \neq 0$ then goto line 8. fi;
6. $\quad$ od;
7. od;
8. $j_{k+1} \leftarrow j$;

The above code fragment will always jump to line 8. for some values of i and j in line (5.) since the input matrix has $j_r = n$ — this is how the rank r is determined. The total cost of computing $j_2, j_3, \ldots, j_r$ is bounded by $O(nm)$ arithmetic operations since $\sum_{k=1..r-1}(j_{k+1} - j_k) = m$. ∎

4.4 Working in a Residue Number System

In this section we show how to do most of the integer arithmetic occurring in algorithm `RowReducedEchelonForm` in a residue number system modulo a basis of word size primes. First we recall the complexity of some standard arithmetic operations. The number of bits in the binary representation of an integer a is given by

$$\lg a = \begin{cases} 1, & \text{if } a = 0; \\ 1 + \lfloor \log_2 |a| \rfloor, & \text{if } a > 0. \end{cases}$$

Using standard arithmetic, a and b can be multiplied in $O((\lg a)(\lg b))$ bit operations, and we can express $a = qb + r$, with $0 \leq |r| < |b|$, in $O((\lg a/b)(\lg b))$ bit operations. We

also use the fact that each direction of the isomorphism implied by the Chinese remainder algorithm can be computed in $O((\lg N)^2)$ bit operations if N is the product of all the moduli.

To work in a residue number system we require knowing beforehand the rank of the input matrix so as to be able to get an *a priori* bound on the size of intermediate integers. We need the following result.

Lemma 14 (Giesbrecht [Gie95]) *Let $x \geq 3$ and $l = 6 + \log\log x$. Then there exists at least $2\lceil\lceil\log_2(2x)\rceil/(l-1)\rceil$ primes p such that $2^{l-1} < p < 2^l$.*

Lemma 15 *There exists a deterministic algorithm that takes as input an $n \times m$ integral input matrix A and returns as output the rank r of A together with the rank profile $[j_1, j_2, \ldots, j_r]$ of A. The cost of the algorithm is $O(nmr\log^2 r||A|| + nmr^2(\log r||A||)(\log r + \log\log||A||))$ bit operations.*

Proof. Let $\beta = (m^{1/2}||A||)^m$ so that by Hadamard's bound β bounds the magnitude of all minors of A. By Lemma 14 we can choose a list of $s = 2\lceil\lceil(\log 2\beta)\rceil/(l-1)\rceil = \Theta((\log\beta)/l)$ primes $[p_1, p_2, \ldots, p_s]$ that are each bounded in length by $l = 6 + \log\log\beta$ bits and such that $p_1 p_2 \cdots p_s > \beta$. The idea is to simply compute the rank profile $L_k = [j_1^{(k)}, j_2^{(k)}, \ldots, j_{r^{(k)}}^{(k)}]$ of A over the finite field $\mathbb{Z}_{p_k}$ of integers modulo p_k for $k = 1, 2, \ldots, s$ in succession. Define the ordering $<_L$ for integer lists as follows: $[a_1, a_2, \ldots, a_{r_a}] <_L [b_1, b_2, \ldots, b_{r_b}]$ if and only if $r_a > r_b$ *or* $r_a = r_b$ and there exists a positive index j such that $a_i = b_i$ for $1 \leq i \leq j$ and $a_j < b_j$. We claim that the actual rank profile of A can be recovered by choosing the smallest (with respect to $<_L$) list from the set $\{L_1, L_2, \ldots, L_s\}$. To see this, note that there must exist at least one nonsingular $r \times r$ minor of the submatrix of A comprised of columns $[j_1, j_2, \ldots, j_r]$; it follows that there must exists at least one prime p_k that doesn't divide this minor. On the other hand, their cannot exists an L_k with $L_k < [j_1, j_2, \ldots, j_r]$ since this would imply that some minor of A was nonsingular over $\mathbb{Z}_p$ but singular over $\mathbb{Z}$, a contradiction. We need one refinement of the algorithm just described. For $i = 1, 2, \ldots, m$, let s_i be the smallest index such that $p_1 p_2 \ldots p_{s_i} > (i^{1/2}||A||)^i$. Instead of computing all the L_k's at once for $k = 1, 2, \ldots, s$, compute the L_k's for $k = 1, 2, \ldots, s_i$ and where $i = 1, 2, \ldots$ in succession. Stop at the *first* i for which the rank of A modulo all of the primes $p_1, p_2, \ldots, p_{s_i}$ is less than i. Then the rank of A must be $i - 1$ since $p_1 p_2 \cdots p_{s_i}$ bounds the magnitudes of all $i \times i$ minors of A. The rank profile of A over $\mathbb{Z}$ can then be recovered by choosing the smallest (with respect to $<_L$) list from the set $\{L_1, L_2, \ldots, L_{s_i}\}$.

We now bound the bit complexity of the method just described. The matrix A_p, obtained by reducing modulo p all entries in A, can be computed in $O(nml\log||A||)$ bit operations. Computing the rank profile of A_p can be done using Guassian elimination over $\mathbb{Z}_p$ at a cost of $O(nmrl^2)$ bit operations. The total cost of computing L_k for $k = 1, 2, \ldots, s_i$ will be $O(s_i \cdot (nmrl^2 + nml\log||A||))$. Substituting $s_i = O((r\log r||A||)/l)$ and $l = O(\log r + \log\log||A||)$ and combining these bounds yields the running time bound claimed by the lemma. ∎

Theorem 16 *There exists a modification of algorithm* `RowReducedEchelonForm` *that requires $O(nmr\log^2 r||A|| + nmr^2(\log r||A||)(\log r + \log\log||A||) + (n+m)r^3\log^2 r||A|| + r^4\log^3 r||A||)$ bit operations using standard integer arithmetic. This bound simplifies to $O(nmr^2\log^2 r||A|| + r^4\log^3 r||A||)$ bit operations.*

Proof. First compute the rank r and rank profile of A using the algorithm of Lemma 15 — this can be accomplished within the running time claimed by the theorem. Let

$\beta = 2(r||A||(20r\log_2(r^{1/2}||A||))^{5/2}(r^{1/2}||A||)^{9r}) + 1$ so that $\log\beta = O(r\log r||A||)$ and by Lemma 11 all intermediate entries in the work matrices Q, C and T of algorithm `RowReducedEchelonForm` will be bounded in magnitude by $(\beta-1)/2$. By Lemma 14 we can choose a list of $s = 2\lceil\lceil(\log 2\beta)\rceil/(l-1)\rceil = \Theta((\log\beta)/l)$ primes $[p_1, p_2, \ldots, p_s]$ that are each bounded in length by $l = 6 + \log\log\beta$ bits and such that $p_1p_2\cdots p_k > \beta$. The idea is to perform all intermediate computations in step (2.) in the residue number system with moduli $[p_1, p_2, \ldots, p_s]$. In step (1.) we need to map all the entries in the input matrix A to the residue number system — this costs $O(nms\log||A||)$ bit operations which is bounded more simply by $O(nmr\log^2 r||A||)$ bit operations. For $k = 1, 2, \ldots, r-1$ we need to convert to standard representation the entries in the $n \times 2$ submatrix B_k of the work matrix T. Similarly, the matrices Q_k and C_k returned by subroutine `ColumnReduction` need to be mapped back to the residue number system. Since the number of nonzero entries in B_k, Q_k and C_k is bounded by $O(n)$, all the transformations in step (2.) to and from the residue number system will cost $O(r\cdot n\cdot r^2\log^2||A||)$ bit operations. In step (3.) we need to map the $(n+m)r$ nonzero entries in the output matrices Q, C and T back to standard representation. This costs $O((n+m)r\cdot r^2\log^2 r||A||)$ bit operations. Finally, note that each arithmetic operation in the residue number system costs $O(sl^2) = O((r\log r||A||)(\log r + \log\log||A||))$ bit operations. The result now follows by combining these complexity bounds and noting that Lemma 12 bounds the number of additional arithmetic operations in step (2.) by $O(nmr)$. ■

Finally, we show that the matrix product $U \leftarrow QC$ can be computed quickly. In the proof of the following Theorem we write $\mathrm{col}(A, j)$ to denote column j of a matrix A.

Theorem 17 *Let Q and C be output matrices produced by* `RowReducedEchelonForm` *with input an $n \times m$ rank r matrix. The product $U \leftarrow QC$ can be computed in $O(nr^3(\log^2 r||A||)(\log r + \log\log||A||))$ bit operations.*

Proof. By Lemma 7 the output matrices Q and T can be written as in (13) with $i = r-1$. In block form we get

$$U = \left[\begin{array}{c|c} Q_1 & \\ \hline Q_2 & I_{n-r} \end{array}\right]\left[\begin{array}{c|c} C_1 & C_2 \\ \hline & I_{n-r} \end{array}\right] = \left[\begin{array}{c} \bar{Q} \\ \hline Q_1 \\ \hline Q_2 \end{array}\right]\left[\begin{array}{c|c} \bar{C} & \\ \hline C_1 & C_2 \end{array}\right] + \left[\begin{array}{c|c} 0_r & \\ \hline & I_{n-r} \end{array}\right]$$

where, by Theorem 9, the entries in $\bar{Q}$ and $\bar{C}$ are bounded in length by $O(r\log r||A||)$ and $O(\log r + \log\log||A||)$ bits respectively. Clearly, the result will follow if we can show that the matrix product $\bar{Q}\bar{C}$ can be computed within the time claimed by the theorem. To get the claimed complexity result we need to take advantage of the possible sparsity of $\bar{C}$. In particular, Lemma 11 guarantees that each row of $\bar{C}$ contains less than $20r\log_2(r^{1/2}||A||)$ nonzero entries so that $\bar{C}$ contains a total of less than $20r^2\log_2(r^{1/2}||A||)$ nonzero entries. We compute $\mathrm{col}(\bar{Q}\bar{C}, j)$ for $j = 1, 2, \ldots, n$ in succession. Let L_j be the set of row indices for which $\bar{C}$ has a nonzero entry in column j. If $|L_j| = 0$ then all entries in column j of $\bar{Q}\bar{C}$ are zero; in this case simply increment j. Otherwise, compute $\mathrm{col}(\bar{Q}\bar{C}, j)$ as the matrix vector product $\bar{Q}\,\mathrm{col}(\bar{C}, j)$ at a cost of $O(n\cdot|L_j|\cdot(\log||\bar{Q}||)(\log||\bar{C}||))$ bit operations. The result now follows by noting that $\sum_{1\le j\le n}|L_j| < 20r^2\log_2(r^{1/2}||A||)$. ■

5 Solving Systems of Linear Diophantine Equations

Let A be an $m \times n$ integer matrix and $\vec{b}$ and n-dimensional integer column vector. Then the system

$$A\vec{x} = \vec{b} \tag{22}$$

is called a system of linear Diophantine equations. A *particular solution* of (22), if one exists, is an integer vector $\vec{x}_p$ that satisfies $A\vec{x}_p = \vec{b}$. If r is the rank of A then the general solution to (22) can be expressed as pair $(\vec{x}_p, Y)$ where $\vec{x}_p$ is a particular solution and Y is an $n \times (n-r)$ rank $n-r$ integer matrix which satisfies $AY = 0$. In other words, the columns of Y comprise a basis for the set of all solutions to the homogeneous system $A\vec{x} = 0$. In case $r = n$ then there exists at most one solution to the system and Y is simply the null matrix (with zero columns). Every solution to (22) can be expressed uniquely as $\vec{x}$ plus some integer linear combination of columns of Y. Thus, the problem of determining the *general solution* to (22) is tantamount to determining either that no solution exists or returning a pair $(\vec{x}_p, Y)$.

Blankinship's [Bla66b] method for computing a general solution to (22) is to compute a row echelon form T of the $(n+1) \times (m+1)$ matrix

$$B = \left[\begin{array}{c|c} -\vec{b}^T & I_1 \\ \hline A^T & \end{array}\right] \tag{23}$$

together with a unimodular transforming matrix U that satisfies $UB = T$. Note that the rank profile of B will be $[j_1, j_2, \ldots, j_r]$ where $j_r = m+1$ and r is one more than the rank of A. Since the first column of U must be the same as the last column of T, we can write U in block form as

$$U = \left[\begin{array}{c|c} \bar{*} & \\ \vdots & \\ \bar{*} & \\ \hline h_r & \vec{x}_p \\ \hline & Y^T \\ & \end{array}\right] \tag{24}$$

where h_r is nonzero, $\vec{x}_p$ is an n-dimensional row vector, and Y is an $n \times (n-r)$ rank r integer matrix.

Lemma 18 (Blankinship [Bla66b]) *For the system $A\vec{x} = \vec{b}$ let B be as in (23) and let U be an $(n+1) \times (n+1)$ unimodular matrix such that UB is in row echelon form. Then U can be written as in (24) and the system (22) admits a particular solution if and only if $|h_r| = 1$. If $|h_r| = 1$ then the general solution to (22) is given by $(h_r\vec{x}_p, Y)$ where h_r, $\vec{x}_p$ and Y are as in (24).*

Theorem 19 *There exists a deterministic algorithm that takes as input an $m \times n$ integer input matrix A together with an n-dimensional integer row vector $\vec{b}$ and returns the general solution to the system of linear Diophantine equations $A\vec{x} = \vec{b}$. If the rank of r is A and $||B|| = \max(||A||, ||\vec{b}||)$ then the cost of the algorithm is $\tilde{O}(nmr^2 \log^2 r||B|| + r^4 \log^4 r||B||)$ bit operations assuming standard integer and matrix multiplication. Moreover, entries in $\vec{x}_p$ and Y will be bounded in length by $O(r \log r||A||)$ bits.*

Proof. Follows directly from Lemma 18 and Theorems 16 and 17. ∎

References

[Bla66a] W. A. Blankinship. Algorithm 287, matrix triangulation with integer arithmetic. *Communications of the ACM*, 9(7):513, July 1966.

[Bla66b] W. A. Blankinship. Algorithm 288, solution of simultaneous linear diophantine equations. *Communications of the ACM*, 9(7):514, July 1966.

[Bod56] E. Bodewig. *Matrix Calculus*. North Holland, Amsterdam, 1956.

[BP74] S. Barnette and I. S. Pace. Efficient algorithms for linear system calculation; part I — Smith form and common divisor of polynomial matrices. *Internat. J. of Systems Sci.*, 5:403–411, 1974.

[Bra71] Gordon H. Bradley. Algorithms for Hermite and Smith normal form matrices and linear diophantine equations. *Mathematics of Computation*, 25(116):897–907, October 1971.

[CC82] Tsu-Wu J. Chou and George E. Collins. Algorithms for the solutions of systems of linear diophantine equations. *SIAM Journal of Computing*, 11:687–708, 1982.

[CW90] D. Coppersmith and S. Winograd. Matrix multiplication via arithmetic progressions. *Journal of Symbolic Computation*, 9:251–280, 1990.

[DKT87] P. D. Domich, R. Kannan, and L. E. Trotter, Jr. Hermite normal form computation using modulo determinant arithmetic. *Mathematics of Operations Research*, 12(1):50–59, February 1987.

[Dom89] P. D. Domich. Residual Hermite normal form computations. *ACM Trans. Math. Software*, 15:275–286, 1989.

[Gan60] F. R. Gantmacher. *Matrix Theory*, volume 1. Chelsea Publishing Company, 1960.

[Gie95] Mark Giesbrecht. Fast computation of the Smith normal form of an integer matrix. In A. H. M. Levelt, editor, *Proc. Int'l. Symp. on Symbolic and Algebraic Computation: ISSAC '95*, pages 110–118, 1995.

[Her51] C. Hermite. Sur l'introduction des variables continues dans la théorie des nombres. *J. Reine Angew. Math.*, 41:191–216, 1851.

[HM91] James L. Hafner and Kevin S. McCurley. Asymptotically fast triangularization of matrices over rings. *SIAM Journal of Computing*, 20(6):1068–1083, December 1991.

[HM94] George Havas and Bohdan S. Majewski. Hermite normal form computation for integer matrices. *Congressus Numerantium*, 105:87–96, 1994.

[Hu69] T. C. Hu. *Integer Programming and Network Flows*. Addison-Wesley, Reading, MA, 1969.

[Ili89a] Costas S. Iliopoulos. Worst-case complexity bounds on algorithms for computing the canonical structure of finite abelian groups and the Hermite and Smith normal forms of an integer matrix. *SIAM Journal of Computing*, 18(4):658–669, August 1989.

[Ili89b] Costas S. Iliopoulos. Worst-case complexity bounds on algorithms for computing the canonical structure of infinite abelian groups and solving systems of linear diophantine equations. *SIAM Journal of Computing*, 18(4):670–678, August 1989.

[KB79] Ravindran Kannan and Achim Bachem. Polynomial algorithms for computing the Smith and Hermite normal forms of and integer matrix. *SIAM Journal of Computing*, 8(4):499–507, November 1979.

[KKS90] Erich Kaltofen, M. S. Krishnamoorthy, and B. David Saunders. Parallel algorithms for matrix normal forms. *Linear Algebra and its Applications*, 136:189–208, 1990.

[Mac56] C. C. MacDuffee. *The Theory of Matrices*. Chelsea, New York, 1956.

[New72] M. Newman. *Integral Matrices*. Academic Press, 1972.

[SL96] Arne Storjohann and George Labahn. Asymptotically fast computation of Hermite normal forms of integer matrices. In Y. N. Lakshman, editor, *Proc. Int'l. Symp. on Symbolic and Algebraic Computation: ISSAC '96*, pages 259–266. ACM Press, 1996.

[SS71] A. Schönhage and V. Strassen. Schnelle Multiplikation grosser Zahlen. *Computing*, 7:281–292, 1971.

[Sto96a] Arne Storjohann. Computing Hermite and Smith normal forms of triangular integer matrices. Technical Report 256, Departement Informatik, ETH Zürich, December 1996.

[Sto96b] Arne 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, 1996.

[Sto96c] Arne Storjohann. Some solutions to the extended gcd problem with applications. Unpublished manuscript., December 1996.