Diss. ETH No. 13922

Algorithms for Matrix Canonical Forms

A dissertation submitted to the
SWISS FEDERAL INSTITUTE OF TECHNOLOGY
ZURICH

for the degree of
Doctor of Technical Sciences

presented by
ARNE STORJOHANN
M. Math., Univ. of Waterloo
born December 20, 1968
citizen of Germany

accepted on the recommendation of
Prof. Dr. Gaston H. Gonnet, examiner
Prof. Dr. Gilles Villard, co-examiner

2013

Acknowledgments

Thanks to

- Gaston Gonnet and Gilles Villard for their support and encouragement
- Thom Mulders for exciting years of collaboration
- other colleagues and mentors in the computer algebra community: Wayne Eberly, Jürgen Gerhard, Mark Giesbrecht, Erich Kaltofen, George Labahn, David Saunders, Joachim von zur Gathen and many others
- Frau Anne Preisig, our institute secretary, for excellent administrative and organizational support
- Leonhard Jaschke for singing, and for assistance in preparing this document
- Ari Kahn for many things, including introducing me to the music of the Grateful Dead
- other friends at ETH: Bettina, Chantal, Christian, Gabi, Gina, Laura, Michela, Mike, Nora, Olli, Preda, Seb, Silvania, Ulrike, Win, Wolf, Xianghong
- Mom and Dad and all my family for their patience and support
- friends back home: Alan, Eric, Laurie, Tania for always being there
- Bill Millar, who once told me that people should write down what they think.

Abstract

Computing canonical forms of matrices over rings is a classical mathematical problem with many applications to computational linear algebra. These forms include the Frobenius form over a field, the Hermite form over a principal ideal domain and the Howell and Smith form over a principal ideal ring. Generic algorithms are presented for computing each of these forms together with associated unimodular transformation matrices. The algorithms are analysed, with respect to the worst case, in terms of number of required operations from the ring. All algorithms are deterministic. For a square input matrix, the algorithms recover each of these forms in about the same number of operations as required for matrix multiplication.

Special emphasis is placed on the efficient computation of transforms for the Hermite and Smith form in the case of rectangular input matrices. Here we analyse the running time of our algorithms in terms of three parameters: the row dimension, the column dimension and the number of nonzero rows in the output matrix.

The generic algorithms are applied to the problem of computing the Hermite and Smith form of an integer matrix. Here the complexity analysis is in terms of number of bit operations. Some additional techniques are developed to avoid intermediate expression swell. New algorithms are demonstrated to construct transformation matrices which have good bounds on the size of entries. These algorithms recover transforms in essentially the same time as required by our algorithms to compute only the form itself.

Kurzfassung

Kanonischen Formen von Matrizen über Ringen zu berechnen, ist ein klassisches mathematisches Problem mit vielen Anwendungen zur konstruktiven linearen Algebra. Diese Formen umfassen die Frobenius Form über einem Körper und die Hermite-, Howell- und Smith-Form über einem Hauptidealring. Wir studieren die Berechnung dieser Formen aus der Sicht von sequentiellen deterministischen Komplexitätsschranken im schlimmsten Fall. Wir präsentieren Algorithmen für das Berechnen aller dieser Formen sowie der dazugehörigen unimodularen Transformationsmatrizen – samt Analyse der Anzahl benötigten Ringoperationen. Die Howell-, Hermite- Smith- und Frobenius-Form einer quadratischen Matrix kann mit ungefähr gleich vielen Operationen wie die Matrixmultiplikation berechnet werden.

Ein Schwerpunkt liegt hier bei der effizienten Berechnung der Hermite- und Smith-Form sowie der dazugehörigen Transformationsmatrizen im Falle einer nichtquadratischen Eingabematrix. In diesem Fall analysieren wir die Laufzeit unserer Algorithmen abhänhig von drei Parametern: die Anzahl der Zeilen, die Anzahl der Spalten und die Anzahl der Zeilen in der berechneten Form, die mindestens ein Element ungleich Null enthalten.

Die generische Algorithmen werden auf das Problem des Aufstellens der Hermite- und Smith-Form einer ganzzahligen Matrix angewendet. Hier wird die Komplizität des Verfahren in der Anzahl der benötigten Bitoperationen ausgedrückt. Einige zusätzliche Techniken wurden entwickelt, um das übermässige Wachsen von Zwischenergebnissen zu vermeiden. Neue Verfahren zur Konstruktion von Transformationsmatrizen für die Hermite- und Smith-Form einer ganzzahligen Matrix wurden entwickelt. Ziel der Bemühungen bei der Entwicklung dieser Verfahren war im Wesentlichen das erreichen der gleichen obere Schranke für die Laufzeit, die unsere Algorithmen benötigen, um nur die Form selbst zu berechnen.

Contents

1 Introduction **1**

1.1 Basic Operations over Rings 11
1.2 Model of Computation 15
1.3 Analysis of algorithms 19
1.4 A Primer on Echelon Forms over Rings 21
1.5 Synopsis and Guide 28

2 Echelon Forms over Fields **33**

2.1 Preliminaries 36
2.2 The GaussJordan Transform 41
2.3 The Gauss Transform 44
2.4 The Modified Gauss Transform 45
2.5 The Triangularizing Adjoint 48

3 Triangularizaton over Rings **53**

3.1 Transformation to Echelon Form 56
3.2 The Index Reduction Transform 63
3.3 Transformation to Hermite Form 66

4 The Howell Form over a PIR **69**

4.1 Preliminaries 72
4.2 The Howell Transform 73

5 Echelon Forms over PIDs **77**

5.1 Modular Computation of an Echelon Form 82
5.2 Fraction Free Computation of an Echelon Form 85
5.3 Solving Systems of Linear Diophantine Equations 88

6 Hermite Form over $\mathbb{Z}$ 91
6.1 Extended Matrix GCD 95
6.2 Computing a Null Space 98

7 Diagonalization over Rings 103
7.1 Reduction of Banded Matrices 105
7.2 From Diagonal to Smith Form 114
7.3 From Upper 2-Banded to Smith Form 117
7.4 Transformation to Smith Form 124

8 Smith Form over $\mathbb{Z}$ 127
8.1 Computing a Smith Conditioner 129
8.2 The Algorithm 135

9 Similarity over a Field 139
9.1 Preliminaries and Subroutines 145
9.2 The Zigzag form 151
9.3 From Block Diagonal to Frobenius Form 155
9.4 From Zigzag to Frobenius Form 157
9.5 Smith Form over a Valuation Ring 160
9.6 Local Smith Forms 163
9.7 The Fast Algorithm 167

10 Conclusions 171
10.1 Algebraic Complexity 171
10.2 Bit Complexity 174

Chapter 1

Introduction

This thesis presents algorithms for computing canonical forms of matrices over rings. For a matrix A over a principal ideal ring R, these include the triangular *Howell form* $H = UA$ and diagonal *Smith form* $S = VAW$ — and for a square matrix A over a field the block diagonal *Frobenius form* $F = PAP^{-1}$. These forms are canonical representatives of the equivalence classes of matrices under unimodular pre-multiplication, unimodular pre- and post-multiplication, and similarity.

Below we describe each of these forms in more detail. To best show the particular structure of a matrix — the shape induced by the nonzero entries — entries which are zero are simply left blank, possibly nonzero entries are labelled with $*$, and entries which satisfy some additional property (depending on the context) are labelled with $\bar{*}$. The $*$ notation is used also to indicate a generic integer index, the range of which will be clear from the context.

The Howell form is an *echelon form* of a matrix A over a principal ideal ring R — nonzero rows proceed zero rows and the first nonzero entry h_* in each nonzero row is to the right of the first nonzero entry in previous rows. The following example is for a 6×9 input matrix.

$$H = UA = \begin{bmatrix} h_1 & * & * & \bar{*} & \bar{*} & * & * & * & * \\ & & & h_2 & \bar{*} & * & * & * & * \\ & & & & h_3 & * & * & * & * \\ & & & & & & & & \\ & & & & & & & & \\ & & & & & & & & \end{bmatrix}.$$

The transforming matrix U is unimodular — this simply means that U is invertible over R. To ensure uniqueness the nonzero rows of H must satisfy some conditions in addition to being in echelon form. When R is a field, the matrix U should be nonsingular and H coincides with the classical Gauss Jordan canonical form — entries h_* are one and entries $\bar{*}$ are zero. When R is a principal ideal domain the Howell form coincides with the better known Hermite canonical form. We wait until Section 1.4 to give more precise definitions of these forms over the different rings. A primary use of the Howell form is to solve systems of linear equations over the domain of entries.

The Smith form

$$S = VAW = \begin{bmatrix} s_1 & & & & & \\ & s_2 & & & & \\ & & \ddots & & & \\ & & & s_r & & \\ & & & & & \\ & & & & & \end{bmatrix}$$

is a canonical form (under unimodular pre- and post-multiplication) for matrices over a principal ideal ring. Each s_i is nonzero and s_i is a divisor of s_{i+1} for $1 \le i \le r-1$. The diagonal entries of S are unique up to multiplication by an invertible element from the ring. The Smith form of an integer matrix is a fundamental tool of abelian group theory. See the text by Cohen (1996) and the monograph and survey paper by Newman (1972, 1997).

Now let A be an $n \times n$ matrix over a field K. The Frobenius form

$$F = PAP^{-1} = \begin{bmatrix} C_{f_1} & & & \\ & C_{f_2} & & \\ & & \ddots & \\ & & & C_{f_l} \end{bmatrix}$$

has each diagonal block C_{f_i} the companion matrix of a monic $f_i \in \mathsf{K}[x]$ and $f_i | f_{i+1}$ for $1 \le i \le l-1$ (see Chapter 9 for more details). This form compactly displays all geometric and algebraic invariants of the input matrix. The minimal polynomial of A is f_l and the characteristic polynomial is the product $f_1 f_2 \cdots f_l$ — of which the constant coefficient is the determinant of A. The rank is the equal to n minus the number

of blocks with zero constant coefficient. The Frobenius form has many uses in addition to recovering these invariants, for example to exponentiate and evaluate polynomials at A and to compute related forms like the rational Jordan form. See Giesbrecht's (1993) thesis for a thorough treatment.

Our programme is to reduce the problem of computing each of the matrix canonical forms described above to performing a number of operations from the ring. The algorithms we present are generic — they are designed for and analysed over an abstract ring R. For the Frobenius form this means a field. For the other forms the most general ring we work over is a principal ideal ring — a commutative ring with identity in which every ideal is principal. Over fields the arithmetic operations $\{+, -, \times,$ divide by a nonzero $\}$ will be sufficient. Over more general rings, we will have to augment this list with additional operations. Chief among these is the "operation" of transforming a 2×1 matrix to echelon form: given $a, b \in \mathsf{R}$, return $s, t, u, v, g \in \mathsf{R}$ such that

$$\begin{bmatrix} s & t \\ u & v \end{bmatrix} \begin{bmatrix} a \\ b \end{bmatrix} = \begin{bmatrix} g \\ \ \end{bmatrix}$$

where $sv - tu$ is a unit from R, and if b is divisible by a then $s = v = 1$ and $t = 0$. We call this operation Gcdex.

Consider for a moment the problems of computing unimodular matrices to transform an input matrix to echelon form (under pre-multiplication) and to diagonal form (under pre- and post-multiplication). Well known constructive proofs of existence reduce these problems to operations of type $\{+, -, \times, \mathrm{Gcdex}\}$. For the echelon form it is well known that $O(n^3)$ such operations are sufficient (see Chapter 3). For the diagonal form over an asbtract ring, it is impossible to derive such an *a priori* bound. As an example, let us attempt to diagonalize the 2×2 input matrix

$$\begin{bmatrix} a & N \\ b & \end{bmatrix}$$

where N is nonzero. First we might apply a transformation of type Gcdex as described above to achieve

$$\begin{bmatrix} s & t \\ u & v \end{bmatrix} \begin{bmatrix} a & N \\ b & \end{bmatrix} \to \begin{bmatrix} g_1 & sN \\ & uN \end{bmatrix}$$

where g_1 is the gcd of a and b. If sN is nonzero, we can apply a transformation of type Gcdex (now via post-multiplication) to compute the gcd

g_2 of g_1 and sN and make the entry in the upper right corner zero. We arrive quickly at the approach of repeatedly triangularizing the input matrix to upper and lower triangular form:

$$\begin{bmatrix} a & N \\ b & \end{bmatrix} \rightarrow \begin{bmatrix} g_1 & * \\ & * \end{bmatrix} \rightarrow \begin{bmatrix} g_2 & \\ * & * \end{bmatrix} \rightarrow \begin{bmatrix} g_3 & * \\ & * \end{bmatrix} \rightarrow \cdots$$

The question is: How many iterations will be required before the matrix is diagonalized? This question is impossible to answer over an abstract ring. All we can say is that, over a principal ideal ring, the procedure is finite. Our solution to this dilemma is to allow the following additional operation: return a ring element c such that the greatest common divisor of the two elements $\{a + cb, N\}$ is equal to that of the three elements $\{a, b, N\}$. We call this operations Stab, and show that for a wide class of principal ideal rings (including all principal ideal domains) it can be reduced constructively to a finite number of operations of type $\{\times, \text{Gcdex}\}$. By first "conditioning" the matrix by adding c times the second row to the first row, a diagonalization can be accomplished in a constant number of operations of type $\{+, -, \times, \text{Gcdex}\}$.

To transform the diagonal and echelon forms to *canonical* form will require some operations in addition to $\{+, -, \times, \text{Gcdex}, \text{Stab}\}$: all such operations that are required — we call them basic operations — are defined and studied in Section 1.1. From now on we will give running time estimates in terms of number of basic operations.

Our algorithms will often allow the use of fast matrix multiplication. Because a lower bound for the cost of this problem is still unknown, we take the approach (following many others) of giving bounds in terms of a parameter θ such that two $n \times n$ matrices over a commutative ring can be multiplied together in $O(n^\theta)$ operations of type $\{+, -, \times\}$ from the ring. Thus, our algorithms allow use of any available algorithm for matrix multiplication. The standard method has $\theta = 3$ whereas the currently asymptotically fastest algorithm allows a θ about 2.376. We assume throughout that θ satisfies $2 < \theta \leq 3$. There are some minor quibbles with this approach, and with the assumption that $\theta > 2$, but see Section 1.2.

In a nutshell, the main theoretical contribution of this thesis is to reduce the problems of computing the Howell, Smith and Frobenius form to matrix multiplication. Given an $n \times n$ matrix A over a principal ideal ring, the Howell form $H = UA$ can be computed in $O(n^\theta)$ and the Smith form $S = VAW$ in $O(n^\theta(\log n))$ basic operations from the ring. Given an $n \times n$ matrix A over a field, the Frobenius form $F = P^{-1}AP$ can be computed in $O(n^\theta(\log n)(\log \log n))$ field operations. The reductions are deterministic and the respective unimodular transforms U, V, W and P are recovered in the same time.

The canonical Howell form was originally described by Howell (1986) for matrices over $\mathbb{Z}/(N)$ but generalizes readily to matrices over an arbitrary principal ideal ring R. Howell's proof of existence is constructive and leads to an $O(n^3)$ basic operations algorithm. When R is a field, the Howell form resolves to the reduced row echelon form and the Smith form to the rank normal form (all $\bar{*}$ entries zero and h_i's and s_i's one). Reduction to matrix multiplication for these problems over fields is known. The rank normal form can be recovered using the LSP-decomposition algorithm of Ibarra *et al.* (1982). Echelon form computation over a field is a key step in Keller-Gehrig's (1985) algorithm for the charactersitic polynomial. Bürgisser *et al.* (1996, Chapter 16) give a survey of fast algorithms for matrices over fields. We show that computing these forms over a principal ideal ring is essentially no more difficult than over a field.

Now consider the problem of computing the Frobenius form of an $n \times n$ matrix over a field. Many algorithms have been proposed for this problem. First consider deterministic algorithms. Lüneburg (1987) and Ozello (1987) give algorithms with running times bounded by $O(n^4)$ field operations in the worst case. We decrease the running time bound to $O(n^3)$ in (Storjohann and Villard, 2000). In Chapter 9 we establish that a transform can be computed in $O(n^\theta(\log n)(\log \log n))$ field operations.

Now consider randomized algorithms. That the Frobenius form can be computed in about the same number of field operations as required for matrix multiplication was first shown by Giesbrecht (1995b). Giesbrecht's algorithm requires an expected number of $O(n^\theta(\log n))$ field operations; this bound assumes that the field has at least n^2 distinct elements. Over a small field, say with only two elements, the expected running time bound for Giesbrecht's asymptotically fast algorithm increases to about $O(n^\theta(\log n)^2)$ and the transform matrix produced might be over a algebraic extension of the ground field. More recently, Eberly (2000) gives an algorithm, applicable over any field, and especially interesting in the small field case, that requires an expected number of $O(n^\theta(\log n))$ field operations to produce a transform.

The problem of computing the Frobenius form has been well studied. Our concern here is sequential deterministic complexity, but much attention has focused also on randomized and fast parallel algorithms. Very recently, Eberly (2000) and Villard (2000) propose and analyse new algorithms for sparse input. We give a more detailed survey in Chapter 9. The algorithms we develop here use ideas from Keller-Gehrig (1985), Ozello (1987), Kaltofen *et al.* (1990), Giesbrecht (1993), Villard (1997) and Lübeck (2002).

A complexity bound given in terms of number of basic operations leaves open the question of how to compute the basic operations themselves. (For example, although greatest common divisors always exists in a principal ideal ring, we might have no effective procedure to compute them.) Fortunately, there are many concrete examples of rings over which we can compute. The definitive example is $\mathbb{Z}$ (a principal ideal domain). The design, analysis and implementation of very fast algorithms to perform basic operations such as multiplication or computation of greatest common divisors over $\mathbb{Z}$ (and many other rings) is the subject of intense study. Bernstein (1998) gives a survey of integer multiplication algorithms.

Complexity bounds given in terms of number of basic operations must be taken *cum grano salis* for another reason: the assumption that a single basic operations has unit cost might be unrealistic. When $\mathsf{R} = \mathbb{Z}$, for example, we must take care to bound the magnitudes of intermediate integers $\to$ intermediate expression swell. An often used technique to avoid expression swell is to compute over a residue class ring $\mathsf{R}/(N)$ (which might be finite compared to R). In many cases, a canonical form over a principal ideal ring R can be recovered by computing over $\mathsf{R}/(N)$ for a well chosen N. Such ideas are well developed in the literature, and part of our contribution here is to explore them further — with emphasis on genericity. To this end, we show in Section 1.1 that all basic operations over a residue class ring of R can be implemented in terms of basic operations from R. Chapter 5 is devoted to the special case R a principal ideal domain and exposes and further develops techniques (many well known) for recovering matrix invariants over R by computing either over the fraction field or over a residue class ring of R.

Tri-parameter Complexity Analysis

While the Frobenius form is defined only for square matrices, the most interesting input matrices for the other forms are often rectangular. For this reason, following the approach of many previous authors, we analyse our algorithms for an $n \times m$ input matrix in terms of the two parameters

n and m. Our algorithm for recovering U when $n > m$ uses ideas from (Hafner and McCurley, 1991), where an $O\tilde{}(nm^{\theta-1})$ basic operations algorithm to recover a non-canonical unimodular triangularization $T = YA$ is given. Figure 1.1 shows the situation when $n > m$. We also

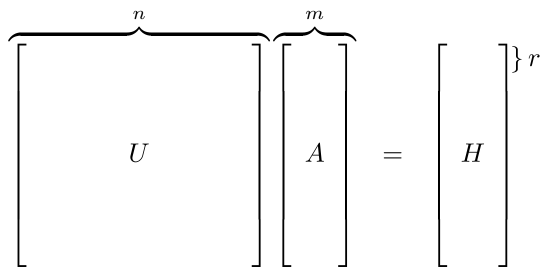


Figure 1.1: Tri-parameter analysis

consider a third parameter r — the number of nonzero rows in the output matrix. The complexity bound for computing the Howell and Smith form becomes $O\tilde{}(nmr^{\theta-2})$ basic operations. Our goal is to provide simple algorithms witnessing this running time bound which handle uniformly all the possible input situations — these are

$$\{n > m, n = m, n < m\} \times \{r = \min(n,m), r < \min(n,m)\}.$$

One of our contributions is that we develop most of our algorithms to work over rings which may have zero divisors. We will have more to say about this in Section 1.4 where we give a primer on computing echelon form over various rings. Here we give some examples which point out the subtleties of doing a tri-parameter analysis.

Over a principal ideal ring with zero divisors we must eschew the use of useful facts which hold over an integral domain — for example the notion of rank as holds over a field. Consider the 4×4 input matrix

$$A = \begin{bmatrix} 8 & 12 & 14 & 7 \\ 8 & 4 & 10 & 13 \\ & & & \\ & & & \end{bmatrix}$$

over $\mathbb{Z}/(16)$. On the one hand, we have

$$\begin{bmatrix} 1 & & & \\ 3 & 1 & & \\ & & 1 & \\ & & & 1 \end{bmatrix} \overset{A}{\begin{bmatrix} 8 & 12 & 14 & 7 \\ 8 & 4 & 10 & 13 \\ & & & \\ & & & \end{bmatrix}} \equiv \begin{bmatrix} 8 & 12 & 14 & 7 \\ & & & \\ & & & \\ & & & \end{bmatrix} \bmod 16$$

On the other hand, we have

$$\begin{bmatrix} 1 & & & \\ 1 & 1 & & \\ & & 1 & \\ & & & 1 \end{bmatrix} \overset{A}{\begin{bmatrix} 8 & 12 & 14 & 7 \\ 8 & 4 & 10 & 13 \\ & & & \\ & & & \end{bmatrix}} \equiv \begin{bmatrix} 8 & 12 & 14 & 7 \\ & & 8 & 4 \\ & & & \\ & & & \end{bmatrix} \bmod 16$$

In both cases we have transformed A to echelon form using a unimodular transformation. Recall that we use the paramater r to mean the number of nonzero rows in the output matrix. The first echelon form has $r = 1$ and the second echelon form has $r = 2$. We call the first echelon form a minimal echelon form of A since r is minimum over all possible echelon forms of A. But the the canonical Howell and Smith form of A over $\mathbb{Z}/(16)$ are

$$H = \begin{bmatrix} 8 & 4 & 2 & 1 \\ & 8 & 4 & 2 \\ & & 8 & 4 \\ & & & 8 \end{bmatrix} \quad \text{and} \quad S = \begin{bmatrix} 1 & & & \\ & & & \\ & & & \\ & & & \end{bmatrix}$$

with $r = 4$ and $r = 1$ respectively. (And when computing the Howell form we must assume that the input matrix has been augmented with zero rows, if necessary, so that $n \geq r$.) We defer until Section 1.4 to define the Howell form more carefully. For now, we just note that what is demonstrated by the above example holds in general:

- The Howell form of A is an echelon form with a maximal number of rows.
- A minimal echelon form will have the same number of nonzero rows as the Smith form.

One of our contributions is to establish that the Smith form together with transform matrices can be recovered in $O\tilde{~}(nmr^{\theta-2})$ basic operations. This result for the Smith form depends on an algorithm for computing a minimal echelon form which also has running time $O\tilde{~}(nmr^{\theta-2})$ basic operations. When the ring is an integral domain r coincides with the unique rank of the input matrix — in this case every echelon and diagonal form will have the same number of nonzero rows. But our point is that, over a ring with zero divisors, the parameter r for the Smith and minimal echelon form can be smaller than that for the Howell form.

This "tri-parameter" model is not a new idea, being inspired by the classical $O(nmr)$ field operations algorithm for computing the Gauss Jordan canonical form over a field.

Canonical Forms of Integer Matrices

We apply our generic algorithms to the problem of computing the Howell, Hermite and Smith form over the concrete rings $\mathbb{Z}$ and $\mathbb{Z}/(N)$. Algorithms for the case $\mathsf{R} = \mathbb{Z}/(N)$ will follow directly from the generic versions by "plugging in" an implementation for the basic operations over $\mathbb{Z}/(N)$. More interesting is the case $\mathsf{R} = \mathbb{Z}$, where some additional techniques are required to keep the size of numbers bounded. We summarize our main results for computing the Hermite and Smith form of an integer matrix by giving running time estimates in terms of bit operations. The complexity model is defined more precisely in Section 1.2. For now, we give results in terms of a function $\mathrm{M}(k)$ such that $O(\mathrm{M}(k))$ bit operations are sufficient to multiply two integers bounded in magnitude by 2^k. The standard method has $\mathrm{M}(k) = k^2$ whereas FFT-based methods allow $\mathrm{M}(k) = k \log k \log\log k$. Note that $O(k)$ bits of storage are sufficient to represent a number bounded in magnitude by $2^{k \times O(1)}$, and we say such a number has length bounded by $O(k)$ bits.

Let $A \in \mathbb{Z}^{n\times m}$ have rank r. Let $||A||$ denote the maximum magnitude of entries in A. We show how to recover the Hermite and Smith form of A in

$$O\tilde{~}(nmr^{\theta-2}(r \log ||A||) + nm\,\mathrm{M}(r \log ||A||))$$

bit operations. This significantly improves on previous bounds (see below). Unimodular transformation matrices are recovered in the same time. Although the Hermite and Smith form are canonical, the transforms to achieve them may be highly non unique[1]. The goal is to produce

[1]Note that $\begin{bmatrix} 1 & u \\ & 1 \end{bmatrix}\begin{bmatrix} 1 & \\ & 1 \end{bmatrix}\begin{bmatrix} 1 & \\ -u & 1 \end{bmatrix} = \begin{bmatrix} 1 & \\ & 1 \end{bmatrix}$ for any u.

transforms with good size bounds on the entries. Our algorithms produce transforms with entries bounded in length by $O(r \log r||A||)$ bits. (Later we derive explicit bounds.) Moreover, when A has maximal rank, one of the transforms for the Smith form will be guaranteed to be very small. For example, in the special case where A has full column rank, the total size (sum of the bit lengths of the entries) of the postmultiplier for the Smith form will be $O(m^2 \log m||A||)$ — note that the total size of the input matrix A might already be more than $m^2 \log_2 ||A||$.

The problems of computing the Hermite and Smith form of an integer matrix have been very well studied. We will give a more thorough survey later. Here we recall the best previously established worst case complexity bounds for these problems under the assumption that $\mathrm{M}(k) = O\tilde{}(k)$. (Most of the previously analysed algorithms make heavy use of large integer arithmetic.) We also assume $n \geq m$. (Many previous algorithms for the Hermite form assume full column rank and the Smith form is invariant under transpose anyway.) Under these simplifying assumptions, the algorithms we present here require $O\tilde{}(nm^\theta \log ||A||)$ bit operations to recover the forms together with transforms which will have entries bounded in length by $O(m \log m||A||)$ bits. The total size of postmultiplier for the Smith form will be $O(m^2 \log m||A||)$.

The transform U for the Hermite form $H = UA$ is unique when A is square nonsingular and can be recovered in $O\tilde{}(n^{\theta+1} \log ||A||)$ bit operations from A and H using standard techniques. The essential problem is to recover a U when n is significantly larger than m, see Figure 1.1. One goal is to get a running time pseudo-linear in n. We first accomplished this in (Storjohann and Labahn, 1996) by adapting the triangularization algorithm of Hafner and McCurley (1991). The algorithm we present here achieves this goal too, but takes a new approach which allows us to more easily derive explicit bounds for the magnitude $||U||$ (and asymptotically better bounds for the bit-length $\log ||U||$).

The derivation of good worst case running time bounds for recovering transforms for the Smith form is a more difficult problem. The algorithm of Iliopoulos (1989a) for this case uses $O\tilde{}(n^{\theta+3}(\log ||A||)^2)$ bit operations. The bound established for the lengths of the entries in the transform matrices is $O\tilde{}(n^2 \log ||A||)$ bits. These bounds are almost certainly pessimistic — note that the bound for a single entry of the postmultipler matches our bound for the total size of the postmultipler.

Now consider previous complexity results for only recover the canonical form itself but not transforming matrices. From Hafner and McCurley (1991) follows an algorithm for Hermite form that requires $O\tilde{}((nm^\theta +$

$m^4) \log ||A||)$ bit operations, see also (Domich *et al.*, 1987) and (Iliopoulos, 1989a). The Smith form algorithm of Iliopoulos (1989a) requires $O\tilde{}(nm^4(\log ||A||)^2)$ bit operations. Bach (1992) proposes a method based on integer factor refinement which seems to require only $O\tilde{}(nm^3(\log ||A||)^2)$ bit operations (under our assumption here of fast integer arithemtic). The running times mentioned so far are all deterministic. Much recent work has focused also on randomized algorithms. A very fast Monte Carlo algorithm for the Smith form of a nonsingular matrix has recently been presented by Eberly *et al.* (2000); we give a more detailed survey in Chapter 8.

Preliminary versions of the results summarized above appear in (Storjohann, 1996c, 1997, 1998b), (Storjohann and Labahn 1996, 1997) and (Storjohann and Mulders 1998). New here is the focus on genericity, the analysis in terms of r, and the algorithms for computing transforms.

1.1 Basic Operations over Rings

Our goal is to reduce the computation of the matrix canonical forms described above to the computation of operations from the ring. Over some rings (such as fields) the operations $\{+, -, \times,$ divide by a nonzero $\}$ will be sufficient. Over more general rings we will need some additional operations such as to compute greatest common divisors. This section lists and defines all the operations — we call them *basic operations* from R — that our algorithms require.

First we define some notation. By PIR (principal ideal ring) we mean a commutative ring with identity in which every ideal is principal. Let R be a PIR. The set of all units of R is denoted by R^*. For $a, b \in \mathsf{R}$, we write (a, b) to mean the ideal generated by a and b. The $(\cdot)$ notation extends naturally to an arbitrary number of arguments. If $(c) = (a, b)$ we call c a gcd of a and b. An element c is said to annihilate a if $ac = 0$. If R has no zero divisors then R is a PID (a principal ideal domain). Be aware that some authors use PIR to mean what we call a PID (for example Newman (1972)).

Two elements $a, b \in \mathsf{R}$ are said to be associates if $a = ub$ for $u \in \mathsf{R}^*$. In a principal ideal ring, two elements are associates precisely when each divides the other. The relation "a is an associate of b" is an equivalence relation on R. A set of elements of R, one from each associate class, is called a prescribed complete set of nonassociates; we denote such a set by $\mathcal{A}(\mathsf{R})$.

Two elements a and c are said to be congruent modulo a nonzero element b if b divides $a - c$. Congruence is also an equivalence relation over R. A set of elements, one from each such equivalence class, is said to be a prescribed complete set of residues with respect to b; we denote such a set by $\mathcal{R}(\mathsf{R}, b)$. By stipulating that $\mathcal{R}(\mathsf{R}, b) = \mathcal{R}(\mathsf{R}, \mathrm{Ass}(b))$, where $\mathrm{Ass}(b)$ is the unique associate of b which is contained in $\mathcal{A}(\mathsf{R})$, it will be sufficient to choose $\mathcal{R}(\mathsf{R}, b)$ for $b \in \mathcal{A}(\mathsf{R})$.

We choose $\mathcal{A}(\mathbb{Z}) = \{0, 1, 2, \ldots\}$ and $\mathcal{R}(\mathbb{Z}, b) = \{0, 1, \ldots, |b| - 1\}$.

List of Basic Operations

Let R be a commutative ring with identity. We will express the cost of algorithms in terms of number of *basic operations* from R. Over an abstract ring, the reader is encouraged to think of these operations as oracles which take as input and return as output a finite number of ring elements.

Let $a, b, N \in \mathsf{R}$. We will always need to be able to perform at least the following: $a+b$, $a-b$, ab, decide if a is zero. For convenience we have grouped these together under the name Arith (abusing notation slightly since the "comparison with zero" operation is unitary).

- $\mathrm{Arith}_{+,-,*,=}(a, b)$: return $a + b$, $a - b$, ab, true if $a = 0$ and false otherwise
- $\mathrm{Gcdex}(a, b)$: return $g, s, t, u, v \in \mathsf{R}$ with $sv - tu \in \mathsf{R}^*$ and

$$\begin{bmatrix} s & t \\ u & v \end{bmatrix} \begin{bmatrix} a \\ b \end{bmatrix} = \begin{bmatrix} g \\ \end{bmatrix}$$

  whereby $s = v = 1$ and $t = 0$ in case b is divisible by a.
- $\mathrm{Ass}(a)$: return the prescribed associate of a
- $\mathrm{Rem}(a, b)$: return the prescribed residue of a with respect to $\mathrm{Ass}(b)$
- $\mathrm{Ann}(a)$: return a principal generator of the ideal $\{b \mid ba = 0\}$

- $\mathrm{Gcd}(a, b)$: return a principal generator of (a, b)
- $\mathrm{Div}(a, b)$: return a $v \in \mathsf{R}$ such that $bv = a$ (if $a = 0$ choose $v = 0$)
- $\mathrm{Unit}(a)$: return a $u \in \mathsf{R}$ such that $ua \in \mathcal{A}(\mathsf{R})$

- $\mathrm{Quo}(a, b)$: return a $q \in \mathsf{R}$ such $a - qb \in \mathcal{R}(\mathsf{R}, b)$
- $\mathrm{Stab}(a, b, N)$: return a $c \in \mathsf{R}$ such that $(a + cb, N) = (a, b, N)$

We will take care to only use operation $\mathrm{Div}(a, b)$ in cases b divides a. If R is a field, and $a \in \mathsf{R}$ is nonzero, then $\mathrm{Div}(1, a)$ is the unique inverse of a. If we are working over a field, the operations Arith and Div are sufficient — we simply say "field operatons" in this case. If R is an integral domain, then we may unambiguously write a/b for $\mathrm{Div}(a, b)$. Note that each of {Gcd, Div, Unit, Quo} can be implemented in terms of the previous operations; the rest of this section is devoted to showing the same for operation Stab when N is nonzero (at least for a wide class of rings including any PID or homomorphic image thereof.)

Lemma 1.1. *Let R be a PIR and $a, b, N \in \mathsf{R}$ with $N \neq 0$. There exists a $c \in \mathsf{R}$ such that $(a + cb, N) = (a, b, N)$.*

Proof. From Krull (1924) (see also (Brown, 1993) or (Kaplansky, 1949)) we know that every PIR is the direct sum of a finite number of integral domains and valuation rings.[2] If R is a valuation ring then either a divides b (choose $c = 0$) or b divides a (choose $c = 1 - \mathrm{Div}(a, b)$).

Now consider the case R is a PID. We may assume that at least one of a or b is nonzero. Let $g = \mathrm{Gcd}(a, b, N)$ and $\bar{g} = \mathrm{Gcd}(a/g, b/g)$. Then $(a/(g\bar{g}) + cb/(g\bar{g}), N/g) = (1)$ if and only if $(a + cb, N) = (g)$. This shows we may assume without loss of generality that $(a, b) = (1)$. Now use the fact that R is a unique factorization domain. Choose c to be a principal generator of the ideal generated by $\{N/\mathrm{Gcd}(a^i, N)) \mid i \in \mathbb{N}\}$. Then $(c, (N/c)) = (1)$ and $(a, c) = 1$. Moreover, every prime divisor of N/c is also a prime divisor of a. It is now easy to show that c satisfies the requirements of the lemma. □

The proof of Lemma 1.1 suggests how we may compute $\mathrm{Stab}(a, b, N)$ when R is a PID. As in the proof assume $(a, b) = 1$. Define $f(a) = \mathrm{Rem}(a^2, N)$. Then set $c = N/\mathrm{Gcd}(f^{\lceil \log_2 k \rceil}(a))$ where k is as in the following corollary. (We could also define $f(a) = a^2$, but the Rem operation will be useful to avoid expression swell over some rings.)

Corollary 1.2. *Let R be a PID and $a, b, N \in \mathsf{R}$ with $N \neq 0$. A $c \in \mathsf{R}$ that satisfies $(a + bc, N) = (a, b, N)$ can be recovered in $O(\log k)$ basic operations of type* {Arith, Rem} *plus $O(1)$ operations of type* {Gcd, Div} *where $k > \max\{l \in \mathbb{N} \mid \exists$ a prime $p \in \mathsf{R} \setminus \mathsf{R}^*$ with $p^l | N\}$.*

[2]Kaplansky (1949) writes that "With this structure theorem on hand, commutative principal ideal rings may be considered to be fully under control."

R is said to be stable if for any $a, b \in \mathsf{R}$ we can find a $c \in \mathsf{R}$ with $(a + cb) = (a, b)$. Note that this corresponds to basic operations Stab when $N = 0$. We get the following as a corollary to Lemma 1.1. We say a residue class ring $\mathsf{R}/(N)$ of R is proper if $N \neq 0$.

Corollary 1.3. *Any proper residue class ring of a PIR is a stable ring.*

Operation Stab needs to be used with care. Either the ring should be stable or we need to guarantee that the third argument N does not vanish.

Notes Howell's 1986 constructive proof of existence of the Howell form uses the fact that $\mathbb{Z}/(N)$ is a stable ring. The construction of the c in the proof of Lemma 1.1 is similar the algorithm for Stab proposed by Bach (1992). Corollary 1.2 is due to Mulders. The operation Stab is a research problem in it's own right, see (Mulders and Storjohann, 1998) and (Storjohann, 1997) for variations.

Basic Operations over a Residue Class Ring

Let $N \neq 0$. Then $\mathsf{R}/(N)$ is a residue class ring of R. If we have an "implementation" of the ring R, that is if we can represent ring elements and perform basic operations, then we can implement basic operations over $\mathsf{R}/(N)$ in terms of basic operations over R. The key is to choose the sets $\mathcal{A}(\cdot)$ and $\mathcal{R}(\cdot,\cdot)$ over $\mathsf{R}/(N)$ *consistently* (defined below) with the choices over R. In other words, the definitions of these sets over $\mathsf{R}/(N)$ should be inherited from the definitions over R. Basic operations over $\mathsf{R}/(N)$ can then be implemented in terms of basic operations over R.

The primary application is when R is a Euclidean domain. Then we can use the Euclidean algorithm to compute gcds in terms of operations {Arith, Rem}. Provided we can also compute Ann over R, the computability of all basic operation over $\mathsf{R}/(N)$ will follow as a corollary.

Let $\phi = \phi_N$ denote the canonical homomorphism $\phi : \mathsf{R} \to \mathsf{R}/(N)$. Abusing notation slightly, define $\phi^{-1} : \mathsf{R}/(N) \to \mathsf{R}$ to satisfy $\phi^{-1}(\bar{a}) \in \mathcal{R}(\mathsf{R}, N)$ for $\bar{a} \in \mathsf{R}/(N)$. Then $\mathsf{R}/(N)$ and $\mathcal{R}(\mathsf{R}, N)$ are isomorphic. Assuming elements from $\mathsf{R}/(N)$ are represented by their unique preimage in $\mathcal{R}(\mathsf{R}, N)$, it is reasonable to make the assumption that the map ϕ costs one basic operation of type Rem, while ϕ^{-1} is free.

Definition 1.4. *For $\bar{a}, \bar{b} \in \mathsf{R}/(N)$, let $a = \phi^{-1}(\bar{a})$ and $b = \phi^{-1}(\bar{b})$. If*

- $\overline{\mathrm{Ass}}(\bar{b}) = \phi(\mathrm{Ass}(\mathrm{Gcd}(b, N)))$.

- $\overline{\mathrm{Rem}}(\bar{a}, \bar{b}) = \phi(\mathrm{Rem}(a, \mathrm{Ass}(\mathrm{Gcd}(b, N)))$

the definitions of $\mathcal{A}(\cdot)$ and $\mathcal{R}(\cdot,\cdot)$ over $\mathsf{R}/(N)$ are said to be consistent with those over R.

Let $\bar{a}, \bar{b}, \bar{d} \in \mathsf{R}/(N)$ and $a = \phi^{-1}(\bar{a})$, $b = \phi^{-1}(\bar{b})$ and $d = \phi^{-1}(\bar{d})$. Below we show how to perform the other basic operations over $\mathsf{R}/(N)$ (indicated using overline) using operations from R.

- $\overline{\mathrm{Arith}}_{+,-,*}(\bar{a}, \bar{b}) := \phi(\mathrm{Arith}_{+,-,*}(a, b))$
- $\overline{\mathrm{Gcdex}}(\bar{a}, \bar{b}) := \phi(\mathrm{Gcdex}(a, b))$
- $\overline{\mathrm{Div}}(\bar{a}, \bar{b}) := \left[\begin{array}{l} (g, s, *, *, *) := \mathrm{Gcdex}(b, N); \\ \textbf{return } \phi(s\mathrm{Div}(a, g)) \end{array}\right.$
- $\overline{\mathrm{Ann}}(\bar{a}) := \left[\begin{array}{l} (*, s, *, u, *) := \mathrm{Gcdex}(a, N); \\ \textbf{return } \phi(\mathrm{Gcd}(\mathrm{Ann}(s), u)) \end{array}\right.$
- $\overline{\mathrm{Gcd}}(\bar{a}, \bar{b}) := \phi(\mathrm{Gcd}(a, b))$
- $\overline{\mathrm{Unit}}(\bar{a}) := \left[\begin{array}{l} (g, s, *, u, *) := \mathrm{Gcdex}(a, N); \\ t := \mathrm{Unit}(g); \\ \textbf{return } \phi(t(s + \mathrm{Stab}(s, u, N)u)) \end{array}\right.$
- $\overline{\mathrm{Quo}}(\bar{a}, \bar{b}) := \mathrm{Div}(\bar{a} - \mathrm{Rem}(\bar{a}, \bar{b}), \bar{b})$
- $\overline{\mathrm{Stab}}(\bar{a}, \bar{b}, \bar{d}) := \phi(\mathrm{Stab}(a, b, \mathrm{Gcd}(d, N))$

1.2 Model of Computation

Most of our algorithms are designed to work over an abstract ring R. We estimate their cost by bounding the number of required *basic operations* from R.

The analyses are performed on an *arithmetic RAM* under the *unit cost model*. By arithmetic RAM we mean the RAM machine as defined in (Aho *et al.*, 1974) but with a second set of *algebraic* memory locations used to store ring elements. By unit cost we mean that each basic operations has unit cost. The usual *binary* memory locations are used to store integers corresponding to loop variables, array indices, pointers, etc. Cost analysis on the arithmetic RAM ignores operations performed with integers in the binary RAM and counts only the number of basic operations performed with elements stored in the algebraic memory.

Computing Basic Operations over $\mathbb{Z}$ or $\mathbb{Z}_N$

When working on an arithmetic RAM where $\mathsf{R} = \mathbb{Z}$ or $\mathsf{R} = \mathbb{Z}/(N)$ we measure the cost of our algorithms in number of bit operations. This is obtained simply by summing the cost in bit operations required by a straight line program in the *bitwise computation model*, as defined in (Aho *et al.*, 1974), to compute each basic operation.

To this end we assign a function $\mathrm{M}(k) : \mathbb{N} \mapsto \mathbb{N}$ to be the cost of the basic operations of type Arith and Quo: given $a, b \in \mathbb{Z}$ with $|a|, |b| \leq 2^k$, each of $\mathrm{Arith}_*(a, b)$ and $\mathrm{Quo}(a, b)$ can be computed in $O_B(\mathrm{M}(n))$ bit operations. The standard methods have $\mathrm{M}(k) = k^2$. The currently fastest algorithms allows $\mathrm{M}(k) = k \log k \log\log k$. For a discussion and comparison of various integer multiplication algorithms, as well as a more detailed exposition of many the ideas to follow below, see von zur Gathen and Gerhard (2003).

Theorem 1.5. *Let integers $a, b, N \in \mathbb{Z}$ all have magnitude bounded by 2^k. Then each of*

- $\mathrm{Arith}_{+,-,=}(a, b)$, $\mathrm{Unit}(a)$, $\mathrm{Ass}(a)$, *determine if* $a \leq b$

can be performed in $O_B(k)$ bit operations. Each of

- Arith_*, $\mathrm{Div}(a, b)$, $\mathrm{Rem}(a, b)$, $\mathrm{Quo}(a, b)$,

can be performed in $O_B(\mathrm{M}(k))$ bit operations. Each of

- $\mathrm{Gcd}(a, b)$, $\mathrm{Gcdex}(a, b)$, $\mathrm{Stab}(a, b, N)$

can be performed in $O_B(\mathrm{M}(k) \log k)$ bit operations.

Proof. An exposition and analysis of algorithms witnessing these bounds can be found in (Aho *et al.*, 1974). The result for Arith_* is due to Schönhage and Strassen (1971) and the Gcdex operation is accomplished using the half–gcd approach of Schönhage (1971). The result for Stab follow from Mulders → Corollary 1.2. □

In the sequel we will give complexity results in terms of the function

$$\mathrm{B}(k) = \mathrm{M}(k) \log k = O(k(\log k)^2(\log\log k)).$$

Every complexity result for algorithms over $\mathbb{Z}$ or $\mathbb{Z}_N$ will be given in terms of a parameter β, a bound on the magnitudes of integers occurring during the algorithm. (This is not quite correct — the bit-length of integers will be bounded by $O(\log \beta)$.)

It is a feature of the problems we study that the integers can become large — both intermediate integers as well as those appearing in the final output. Typically, the bit-length increases about linearly with the dimension of the matrix. For many problems we have $\beta = (\sqrt{r}||A||)^r$ where $||A||$ bounds the magnitudes of entries in the input matrix A of rank r. For example, a 1000×1000 input matrix with entries between -99 and 99 might lead to integers with 3500 decimal digits.

To considerably speed up computation with these large integers in practice, we perform the lion's share of computation modulo a basis of small primes, also called a RNS (Residue Number System). A collection of s distinct odd primes p_* gives us a RNS which can represent signed integers bounded in magnitude by $p_1 p_2 \cdots p_s/2$. The RNS representation of such an integer a is the list $(\mathrm{Rem}(a, p_1), \mathrm{Rem}(a, p_2), \ldots, \mathrm{Rem}(a, p_s))$.

Giesbrecht (1993) shows, using bounds from Rosser and Schoenfeld (1962), that we can choose $l \geq 6 + \log\log \beta$. In other words, for such an l, there exist at least $s = 2\lceil(\log_2 2\beta)/(l-1)\rceil$ primes p_* with $2^{l-1} < p_* < 2^l$, and the product of s such primes will be greater than 2β. (Recall that we use the paramater β as a bound on magnitudes of integers that arise during a given computation.) A typical scheme in practice is to choose l to be the number of bits in the machine word of a given binary computer. For example, there are more than $2 \cdot 10^{17}$ 64-bit primes, and more than 98 million 32-bit primes.

From Aho *et al.* (1974), Theorem 8.12, we know that the mapping between standard and RNS representation (the isomorphism implied by the Chinese remainder theorem) can be performed in either direction in time $O_B(\mathrm{B}(\log \beta))$. Two integers in the RNS can be multiplied in time $O_B(s \cdot \mathrm{M}(l))$. We are going to make the assumption that the multiplication table for integers in the range $[0, 2^l - 1]$ has been precomputed. This table can be built in time $O_B((\log \beta)^2 \,\mathrm{M}(l))$. Using the multiplication table, two integers in the RNS can be multiplied in time $O_B(\log \beta)$. Cost estimates using this table will be given in terms of *word operations*.

Complexity estimates in terms of word operations may be transformed to obtain the true asymptotic bit complexity (i.e. without assuming linear multiplication time for l-bit words) by replacing terms $\log \beta$ not occuring as arguments to $\mathrm{B}(\cdot)$ as follows

$$(\log \beta) \to (\log \beta)\, \mathrm{M}(\log\log \beta)/(\log\log \beta)$$

Matrix Computations

Let R be a commutative ring with identity (the most general ring that we work with). Let $\mathrm{MM}(a,b,c)$ be the number of basic operation of type Arith required to multiply an $a \times b$ matrix together with a $b \times c$ matrix over R. For brevity we write $\mathrm{MM}(n)$ to mean $\mathrm{MM}(n,n,n)$. Standard matrix multiplication has $\mathrm{MM}(n) \leq 2n^3$. Better asymptotic bounds are available, see the notes below. Using an obvious block decomposition we get:

Fact 1.6. *We have*

$$\mathrm{MM}(a,b,c) \leq \lceil a/r \rceil \cdot \lceil b/r \rceil \cdot \lceil c/r \rceil \cdot (\mathrm{MM}(r) + r^2)$$

where $r = \min(a,b,c)$.

Our algorithms will often reduce a given problem to that of multiplying a number of matrices of smaller dimension. To give complexity results in terms of the function $\mathrm{MM}(\cdot)$ would be most cumbersome. Instead, we use a parameter θ such that $\mathrm{MM}(n) = O(n^\theta)$ and make the assumption in our analysis that $2 < \theta \leq 3$. As an example of how we use this assumption, let $n = 2^k$. Then the bound

$$S = \sum_{i=0}^{k} 4^i \,\mathrm{MM}(n/2^i) = O(n^\theta)$$

is easily derived. But note, for example, that if $\mathrm{MM}(n) = \Theta(n^2 (\log n)^c)$ for some integer constant c, then $S = O(\mathrm{MM}(n)(\log n))$. That is, we get an extra log factor. On the one hand, we will be very concerned with logarithmic factors appearing in the complexity bounds of our generic algorithms (and try to expel them whenever possible). On the other hand, we choose not to quibble about such factors that might arise under the assumption that the cost of matrix multiplication is softly quadratic; if this is shown to be the case the analysis of our algorithms can be redone.

Now consider the case $\mathsf{R} = \mathbb{Z}$. Let $A \in \mathbb{Z}^{a \times b}$ and $B \in \mathbb{Z}^{b \times c}$. We will write $||A||$ to denote the maximum magnitude of all entries in A. Then $||AB|| \leq b \cdot ||A|| \cdot ||B||$. By passing over the residue number system we get the following:

Lemma 1.7. *The product AB can be computed in*

$$O(\mathrm{MM}(a,b,c)(\log \beta) + (ab + bc + ac)\,\mathrm{B}(\log \beta))$$

word operations where $\beta = b \cdot ||A|| \cdot ||B||$.

Notes

The currently best known upper bound for θ is about 2.376, due to Coppersmith and Winograd (1990). The derivation of upper and lower bounds for $\mathrm{MM}(\cdot)$ is an important topic in algebraic complexity theory, see the text by Bürgisser *et al.* (1996). Note that Fact 1.6 implies $\mathrm{MM}(n,n,n^r) = O(n^{2+r(\theta-2)})$ for $0 < r \leq 1$. This bound for rectangular matrix multiplication can be substantially improved. For example, (Coppersmith96) shows that $\mathrm{MM}(n,n,n^r) = O(n^{2+\epsilon})$ for any $\epsilon > 0$ if $r \leq 0.294$, $n \to \infty$. For recent work and a survey of result on rectangular matrix multiplication, see Huang and Pan (1997).

1.3 Analysis of algorithms

Throughout this section, the variables n and m will be positive integers (corresponding to a row and column dimensions respectively) and r will be a nonnegative integer (corresponding, for example, to the number of nonzero rows in the output matrix). We assume that θ satisfies $2 < \theta \leq 3$.

Many of the algorithms we develop are recursive and the analysis will involve bounding a function that is defined via a recurrence relation. For example, if

$$f_\gamma(m) = \begin{cases} \gamma & \text{if } m = 1 \\ 2f_\gamma(\lceil m/2 \rceil) + \gamma m^{\theta-1} & \text{if } m > 1 \end{cases}$$

then $f_\gamma(m) = O(\gamma m^{\theta-1})$. Note that the big O estimate also applies to the parameter γ.

On the one hand, techniques for solving such recurrences, especially also in the presence of "floor" and "ceiling" functions, is an interesting topic in it's own right, see the text by Cormen *et al.* (1989). On the other hand, it will not be edifying to burden our proofs with this topic. In subsequent chapters, we will content ourselves with establishing the recurrence together with the base cases. The claimed bounds for recurrences that arise will either follow as special cases of the Lemmas 1.8 and 1.9 below, or from Cormen *et al.* (1989), Theorem 4.1 (or can be derived using the techniques described there).

Lemma 1.8. *Let c be an absolute constant. The nondeterministic func-*

tion $f : \mathbb{Z}^2_{\geq 0} \to \mathbb{R}_{\geq 0}$ *defined by*

$f_\gamma(m, r) =$

```
if $m = 1$ or $r = 0$ then return $\gamma cm$
else
    Choose nonngative $r_1$ and $r_2$ which satisfy $r_1 + r_2 = r$;
    return $f_\gamma(\lfloor m/2 \rfloor, r_1) + f_\gamma(\lceil m/2 \rceil, r_2) + \gamma cm$
fi
```

satisfies $f_\gamma(m, r) = O(\gamma m \log r)$.

Proof. It will be sufficient to prove the result for the case $\gamma = 1$. Assume for now that m is a power of two (we will see later that we may make this assumption).

Consider any particular execution tree of the function. The root is labelled (m, r) and, if $m > 1$ and $r > 0$, the root has two children labelled $(m/2, r_1)$ and $(m/2, r_2)$. In general, level i $(0 \leq i \leq \log_2 m)$ has at most 2^i nodes labelled $(m/2^i, *)$. All nodes at level i have associated cost $cm/2^i$ and if either $i = \log_2 m$ or the second argument of the label is zero the node is a leaf (one of the base cases). The return value of $f(m, r)$ with this execution tree is obtained by adding all the costs.

The cost of all the leaves is at most cm. It remains to bound the "merging cost" associated with the internal nodes. The key observation is that there can be at most r internal nodes at each level of the tree. The result follows by summing separately the costs of all internal nodes up to and including level $\lceil \log 2r \rceil$ (yielding $O(m \log r)$), and after level $\lceil \log_2 r \rceil$ (yielding $O(m)$).

Now consider the general case, when m may not a power of two. Let $\bar{m}$ be the smallest power of two greater than or equal m. Then $\lceil m/2 \rceil \leq \bar{m}/2$ implies $\lceil \lceil m/2 \rceil /2 \rceil < \bar{m}/4$ and so on. Thus any tree with root (m, r) can be embedded in some execution tree with root $(\bar{m}, r)$ such that the corresponding nodes in $(\bar{m}, r)$ have cost greater or equal to the associated node in (m, r). □

Lemma 1.9. *Let* $r, r_1, r_2 \geq 0$ *satisfy* $r_1 + r_2 = r$. *Then* $r_1^{\theta-2} + r_2^{\theta-2} \leq 2^{3-\theta} r^{\theta-2}$.

Lemma 1.10. *Let c be a an absolute constant. The nondeterministic*

function $f_\gamma : \mathbb{Z}^2_{\geq 0} \to \mathbb{R}_{\geq 0}$ *defined by*

$f_\gamma(m, r) =$

```
if $m = 1$ or $r = 0$ then return $\gamma cm$
else
    Choose nonngative $r_1$ and $r_2$ which satisfy $r_1 + r_2 = r$;
    return $f_\gamma(\lfloor m/2 \rfloor, r_1) + f_\gamma(\lceil m/2 \rceil, r_2) + \gamma cmr^{\theta-2}$
fi
```

satisfies $f_\gamma(m, r) = O(\gamma m r^{\theta-2})$.

Proof. It will be sufficient to consider the case when m is a power of two, say $m = 2^k$. (The same "tree embedding" argument used in the proof of Lemma 1.8 works here as well.) Induction on k, together with Lemma 1.9, shows that $f_\gamma(m, r) \leq 3c/(1 - 2^{2-\theta})\gamma m r^{\theta-2}$. □

1.4 A Primer on Echelon Forms over Rings

Of the remaining eight chapters of this thesis, five are concerned with the problem of transforming an input matrix A over a ring to echelon form under unimodular pre-multiplication. This section gives a primer on this topic. The most familiar situation is matrices over a field. Our purpose here is to point out the key differences when computing echelon forms over more general rings and thus to motivate the work done in subsequent chapters.

$$UA = \left[\begin{array}{ccccccccc} & & & & H & & & & \\ h_1 & * & * & \bar{*} & \bar{*} & * & * & * & * \\ & & & h_2 & \bar{*} & * & * & * & * \\ & & & & h_3 & * & * & * & * \\ & & & & & & & & \\ & & & & & & & & \end{array}\right] \qquad VAW = \left[\begin{array}{ccccc} & & S & & \\ s_1 & & & & \\ & s_2 & & & \\ & & \ddots & & \\ & & & s_r & \\ & & & & \\ & & & & \end{array}\right]$$

Figure 1.2: Transformation to echelon and diagonal form

We begin with some definitions. Let R be a commutative ring with identity. A square matrix U over R is *unimodular* if there exists a matrix V over R such that $UV = I$. Such a V, if it exists, is unique and also satisfies $VU = I$. Thus, unimodularity is precisely the notion of invertibility over a field extended to a ring. Two matrices $A, H \in \mathsf{R}^{n \times m}$

are *left equivalent* to each other if there exists a unimodular matrix U such that $UA = H$. Two matrices $A, S \in \mathsf{R}^{n\times m}$ are *equivalent* to each other if there exists unimodular matrices V and W such that $VAW = S$. Equivalence and left equivalence are equivalence relations over $\mathsf{R}^{n\times m}$.

Following the historical line, we first consider the case of matrices over field, then a PID and finally a PIR. Note that a field is a PID and a PID is a PIR. We will focus our discussion on the echelon form.

Echelon forms over fields Consider the matrix

$$A = \begin{bmatrix} -10 & 35 & -10 & 2 \\ -16 & 56 & -17 & 3 \\ 54 & -189 & 58 & -10 \end{bmatrix} \tag{1.1}$$

to be over the field $\mathbb{Q}$ of rational numbers. Applying Gaussian elimination to A yields

$$\left[\begin{array}{ccc} 1 & 0 & 0 \\ -8/5 & 1 & 0 \\ \hline -1 & 4 & 1 \end{array}\right] A = \left[\begin{array}{cccc} -10 & 35 & -10 & 2 \\ & & -1 & -1/5 \\ \hline & & & \\ \end{array}\right] \begin{array}{l} \left.\vphantom{\begin{array}{c}1\\1\end{array}}\right\} r \\ \\ \end{array}$$

where the transforming matrix on the left is nonsingular and the matrix on the right is an echelon form of A. The number r of nonzero rows in the echelon form is the rank of A. The last $n-r$ rows of the transforming matrix comprise a basis for the null space of A over $\mathbb{Q}$. Continuing with some elementary row operations (which are invertible over $\mathbb{Q}$) we get

$$\overset{U}{\left[\begin{array}{ccc} 0 & -\frac{29}{5} & -\frac{17}{10} \\ 0 & \frac{27}{5} & 8/5 \\ \hline 1 & -4 & -1 \end{array}\right]} A = \overset{H}{\left[\begin{array}{cccc} 1 & -7/2 & 0 & -2/5 \\ & & 1 & 1/5 \\ \hline & & & \\ \end{array}\right]}$$

with H the Gauss Jordan canonical form of A and $\det U = -121/50$ (i.e. U is nonsingular). Given another matrix $B \in \mathbb{Q}^{*\times m}$, we can assay if the vector space generated by the rows of A is equal to that generated by the rows of B by comparing the Gauss Jordan forms of A and B. Note that if A is square nonsingular, the the Gauss Jordan form of H of A is the identity matrix, and the transform U such that $UA = H$ is the unique inverse of A.

Echelon forms over PIDs Now let R be a PID. A canonical form for left equivalence over R is the Hermite form — a natural generalization of the Gauss Jordan form over a field. The Hermite form of an $A \in \mathsf{R}^{n\times m}$ is the $H \in \mathsf{R}^{n\times m}$ which is left equivalent to A and which satisfies:

(r1) H is in echelon form, see Figure 1.4.

(r2) Each $h_* \in \mathcal{A}(\mathsf{R})$ and entries $\bar{*}$ above h_* satisfy $\bar{*} \in \mathcal{R}(\mathsf{R}, h_*)$.

As a concrete example of the form, consider the matrix

$$A = \begin{bmatrix} -10 & 35 & -10 & 2 \\ -16 & 56 & -17 & 3 \\ 54 & -189 & 58 & -10 \end{bmatrix} \tag{1.2}$$

to be over the ring of integers. We will transform A to echelon form via unimodular row transformations in two stages. Note that over $\mathbb{Z}$ the unimodular matrices are precisely those with determinant ± 1. Gaussian elimination (as over $\mathbb{Q}$) might begin by zeroing out the second entry in the first column by adding an appropriate multiple of the first row to the second row. Over the PID $\mathbb{Z}$ this is not possible since -16 is not divisible by -10. First multiplying the second row by 5 would solve this problem but this row operation is not invertible over $\mathbb{Z}$. The solution is to replace the division operation with Gcdex. Note that $(-2, -3, 2, 8, 5)$ is a valid return tuple for the basic operation Gcdex$(-10, -16)$. This gives

$$\begin{bmatrix} -3 & 2 & \\ 8 & -5 & \\ & & 1 \end{bmatrix} \overset{A}{\begin{bmatrix} -10 & 35 & \\ -16 & 56 & \cdots \\ 54 & -189 & \end{bmatrix}} = \begin{bmatrix} -2 & 7 & \\ & 0 & \cdots \\ 54 & -189 & \end{bmatrix}$$

The next step is to apply a similar transformation to zero out entry 54. A valid return tuple for the basic operation Gcdex$(-2, 54)$ is $(-2, 1, 0, 27, 1)$. This gives

$$\begin{bmatrix} 1 & & \\ & 1 & \\ 27 & & 1 \end{bmatrix} \begin{bmatrix} -2 & 7 & \\ & 0 & \cdots \\ 54 & -189 & \end{bmatrix} = \begin{bmatrix} -2 & 7 & \\ & 0 & \cdots \\ & 0 & \end{bmatrix}.$$

Note that the first two columns (as opposed to only one column) of the matrix on the right hand side are now in echelon form. This is because, for this input matrix, the first two columns happen to have rank one. Continuing in this fashion, from left to right, using transformation of type Gcdex, and multiplying all the transforms together, we get the echelon form

$$\left[\begin{array}{ccc} -3 & 2 & 0 \\ 8 & -5 & 0 \\ \hline -1 & 4 & 1 \end{array}\right] A = \left[\begin{array}{cccc} -2 & 7 & -4 & 0 \\ & & 5 & 1 \\ \hline & & & \end{array}\right].$$

The transforming matrix has determinant -1 (and so is unimodular over $\mathbb{Z}$). This completes the first stage. The second stage applies some elementary row operations (which are invertible over $\mathbb{Z}$) to ensure that each h_* is positive and entries $\bar{*}$ are reduced modulo the diagonal entry in the same column (thus satisfying condition (r2)). For this example we only need to multiply the first row by -1. We get

$$\overset{U}{\left[\begin{array}{ccc} 3 & -2 & 0 \\ 8 & -5 & 0 \\ \hline -1 & 4 & 1 \end{array}\right]} A = \overset{H}{\left[\begin{array}{cccc} 2 & -7 & 4 & 0 \\ & & 5 & 1 \\ \hline & & & \end{array}\right]} \tag{1.3}$$

where H is now in Hermite form. This two stage algorithm for transformation to Hermite form is given explicitly in the introduction of Chapter 3.

Let us remark on some similarities and differences between echelon forms over a field and over a PID. For an input matrix A over a PID R, let $\bar{A}$ denote the embedding of A into the fraction field $\bar{\mathsf{R}}$ of R (eg. $\mathsf{R} = \mathbb{Z}$ and $\bar{\mathsf{R}} = \mathbb{Q}$). The similarity is that every echelon form of A (over R) or $\bar{A}$ (over $\bar{\mathsf{R}}$) will have the same rank profile, that is, the same number r of nonzero rows and the entries h_* will be located in the same columns. The essential difference is that any r linearly independent rows in the row space of $\bar{A}$ constitute a basis for the row space of $\bar{A}$. For A this is not the case[3]. The construction of a basis for the set of all R-linear combinations of rows of A depends essentially on the basic operation Gcdex.

[3]For example, neither row of $\left[\begin{smallmatrix} 2 \\ 3 \end{smallmatrix}\right] \in \mathbb{Z}^{2\times 1}$ generates the rowspace, which is $\mathbb{Z}^1$.

A primary use of the Hermite form is to solve a system of linear diophantine equations over a PID R. Continuing with the same example over $\mathbb{Z}$, let us determine if the vector $b = \left[\begin{array}{cccc} 4 & -14 & 23 & 3 \end{array}\right]$ can be expressed as a $\mathbb{Z}$-linear combination of the rows of the matrix A in (1.2). In other words, does there exists an integer vector x such that $xA = b$? We can answer this question as follows. First augment an echelon form of A (we choose the Hermite form H) with the vector b as follows

$$\left[\begin{array}{c|cccc} 1 & 4 & -14 & 23 & 3 \\ \hline & 2 & -7 & 4 & 0 \\ & & & 5 & 1 \\ & & & & \end{array}\right].$$

Because H is left equivalent to A, the answer to our question will be affirmative if and only if we can express b as an R-linear combination of the rows of H. (The last claim is true over any commutative ring R.) Now transform the augmented matrix so that the off-diagonal entries in the first row satisfy condition (r2). We get

$$\left[\begin{array}{c|ccc} 1 & -2 & -3 & 0 \\ \hline & 1 & & \\ & & 1 & \\ & & & 1 \end{array}\right] \left[\begin{array}{c|cccc} 1 & 4 & -14 & 23 & 3 \\ \hline & 2 & -7 & 4 & 0 \\ & & & 5 & 1 \\ & & & & \end{array}\right] = \left[\begin{array}{c|cccc} 1 & 0 & 0 & 0 & 0 \\ \hline & 2 & -7 & 4 & 0 \\ & & & 5 & 1 \\ & & & & \end{array}\right]$$

We have considered here a particular example, but in general there are two conclusions we may now make:

- If all off-diagonal entries in the first row of the transformed matrix are zero, then the answer to our question is affirmative.
- Otherwise, the the answer to our question is negative.

On our example, we may conclude, comparing with (1.3), that the vector

$$x = \left[\begin{array}{cc} 2 & 3 \end{array}\right] \left[\begin{array}{ccc} 3 & -2 & 0 \\ 8 & -5 & 0 \end{array}\right] = \left[\begin{array}{ccc} 30 & -16 & 0 \end{array}\right]$$

satisfies $xA = b$. This method for solving a linear diophantine system is applicable over any PID.

Echelon forms over PIRs Now consider the input matrix A of (1.2) as being over the PIR $\mathbb{Z}/(4)$. Note that

$$A \equiv \begin{bmatrix} 2 & 3 & 2 & 2 \\ 0 & 0 & 3 & 3 \\ 2 & 3 & 2 & 2 \end{bmatrix} \bmod 4$$

so the the transformation of A to echelon form is easy:

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 3 & 0 & 1 \end{bmatrix} \begin{bmatrix} 2 & 3 & 2 & 2 \\ 0 & 0 & 3 & 3 \\ 2 & 3 & 2 & 2 \end{bmatrix} \equiv \begin{bmatrix} 2 & 3 & 2 & 2 \\ & & 3 & 3 \\ & & & \end{bmatrix} \bmod 4.$$

In general, the same procedure as sketched above to compute an echelon form over a PID will work here as well. But there are some subtleties since $\mathbb{Z}/(4)$ is a ring with zero divisors. We have already seen, with the example on page 8, that different echelon forms of A can have different numbers of nonzero rows. For our example here we could also obtain

$$\begin{bmatrix} 1 & 2 & 0 \\ 0 & 3 & 0 \\ 3 & 0 & 1 \end{bmatrix} \begin{bmatrix} 2 & 3 & 2 & 2 \\ 0 & 0 & 3 & 3 \\ 2 & 3 & 2 & 2 \end{bmatrix} \equiv \begin{bmatrix} 2 & 3 & 0 & 0 \\ & & 1 & 1 \\ & & & \end{bmatrix} \bmod 4 \qquad (1.4)$$

and

$$\overset{U}{\begin{bmatrix} 0 & 2 & 3 \\ 1 & 0 & 1 \\ 0 & 3 & 0 \end{bmatrix}} \overset{A}{\begin{bmatrix} 2 & 3 & 2 & 2 \\ 0 & 0 & 3 & 3 \\ 2 & 3 & 2 & 2 \end{bmatrix}} \equiv \overset{H}{\begin{bmatrix} 2 & 1 & 0 & 0 \\ & 2 & 0 & 0 \\ & & 1 & 1 \end{bmatrix}} \bmod 4 \qquad (1.5)$$

Both of these echelon forms satisfy condition (r2) and so are in Hermite form. Obviously, we may conclude that the Hermite form is not a canonical form for left equivalence of matrices over a PIR. We need another condition, which we develop now. By $\mathrm{S}(A)$ we mean the set of all R-linear combinations of rows of A, and by $\mathrm{S}_j(A)$ the subset of $\mathrm{S}(A)$ comprised of all rows which have first j entries zero. The echelon form H in (1.5) satisfies the *Howell property*: $\mathrm{S}_1(A)$ is generated by the last two rows and $\mathrm{S}_2(A)$ is generated by the last row. The Howell property is defined more precisely in Chapter 4. For now, we just point out that, for an

echelon form that satisfies the Howell property, the procedure sketched above for solving a linear system is applicable. (In fact, a Hermite form of A is in Howell form precisely when this procedure works correctly for every $b \in \mathsf{R}^m$.) The echelon form in (1.4) is not suitable for this task. Note that

$$\left[\begin{array}{c|cccc} 1 & 0 & 2 & 0 & 0 \\ \hline & 2 & 3 & 0 & 0 \\ & & & 1 & 1 \\ & & & & \end{array}\right]$$

is in Hermite form, but $\begin{bmatrix} 0 & 2 & 0 & 0 \end{bmatrix}$ is equal, modulo 4, to 2 times $\begin{bmatrix} 2 & 3 & 0 & 0 \end{bmatrix}$.

An echelon form which satisifes the Howell property has the maximum number of nonzero rows that an echelon form can have. For complexity theoretic purposes, it will be useful also to recover an echelon form as in (1.4) which has a minimum number of nonzero rows.

We have just established and motivated four conditions which we might want an echelon form H of an input matrix A over a PIR to possess. Using these conditions we distinguish in Table 1.4 a number of intermediate echelon forms which arise in the subsequent chapters.

	Conditions			
Form	r1	r2	r3	r4
echelon	●			
minimal echelon	●		●	
Hermite	●	●		
minimal Hermite	●	●	●	
weak Howell	●			●
Howell	●	●		●

(r1) H is in echelon form, see Figure 1.4.

(r2) Each $h_* \in \mathcal{A}(\mathsf{R})$ and $\bar{*} \in \mathcal{R}(\mathsf{R}, h_*)$ for $\bar{*}$ above h_*.

(r3) H has a minimum number of nonzero rows.

(r4) H satisfies the Howell property.

Table 1.1: Echelon forms over PIRs

1.5 Synopsis and Guide

The remaining eight chapters of this thesis can be divided intro three parts:

Left equivalence: Chapters 2, 3, 4, 5, 6.
Equivalence: Chapters 7 and 8.
Similarity: Chapter 9.

Chapters 2 through 6 are concerned primarily with computing various echelon forms of matrices over a field, a PID or a PIR. Figure 1.3 recalls the relationship between these and some other rings. At the start of

$$\begin{tikzcd}
\text{field} \arrow[d] \arrow[dr] & \\
\text{PID} \arrow[r] \arrow[d] & \text{ID} \arrow[d] \\
\text{PIR} \arrow[r] & \text{commutative with identity}
\end{tikzcd}$$

Figure 1.3: Relationship between rings

each chapter we give a high level synopsis which summarizes the main results of the chapter and exposes the links and differences to previous chapters. For convenience, we collect these eight synopsis together here.

Chapter 2: Echelon Forms over a Field Our starting point, appropriately, is the classical Gauss and Gauss Jordan echelon forms for a matrix over a field. We present simple to state and implement algorithms for these forms, essentially recursive versions of fraction free Gaussian elimination, and also develop some variations which will be useful to recover some additional matrix invariants. These variations include a modification of fraction free Gaussian elimination which conditions the pivot entries in a user defined way, and the triangularizing adjoint, which can be used to recover all leading minors of an input matrix. All the algorithms are fraction free and hence applicable over an integral domain.

Complexity results in the bit complexity model for matrices over $\mathbb{Z}$ are also stated. The remaining chapters will call upon the algorithms here many times to recover invariants of an input matrix over a PID such as the rank, rank profile, adjoint and determinant.

Chapter 3: Triangularization over Rings The previous chapter considered the fraction free computation of echelon forms over a field. The algorithms there exploit a feature special to fields — every nonzero element is a divisor of one. In this chapter we turn our attention to computing various echelon forms over a PIR, including the Hermite form which is canonical over a PID. Here we need to replace the division operation with Gcdex. This makes the computation of a single unimodular transform for achieving the form more challenging. An additional issue, especially from a complexity theoretic point of view, is that over a PIR an echelon form might not have a unique number of nonzero rows — this is handled by recovering echelon and Hermite forms with minimal numbers of nonzero rows. The primary purpose of this chapter is to establish sundry complexity results in a general setting — the algorithms return a single unimodular transform and are applicable for any input matrix over any PIR (of course, provided we can compute the basic operations).

Chapter 4: Howell Form over a PIR This chapter, like the previous chapter, is about computing echelon forms over a PIR. The main battle fought in the previous chapter was to return a single unimodular transform matrix to achieve a minimal echelon form. This chapter takes a more practical approach and presents a simple to state and implement algorithm — along the lines of those presented in Chapter 2 for echelon forms over fields — for producing the canonical Howell form over a PIR. The algorithm is developed especially for the case of a stable PIR (such as any residue class ring of a PID). Over a general PIR we might have to augment the input matrix to have some additional zero rows. Also, instead of producing a single unimodular transform matrix, we express the transform as a product of structured matrices. The usefulness of this approach is exposed by demonstating solutions to various linear algebra problems over a PIR.

Chapter 5: Echelon Forms over PIDs The last three chapters gave algorithms for computing echelon forms of matrices over rings. The focus of Chapter 2 was matrices over fields while in Chapter 3 all the algorithms are applicable over a PIR. This chapter focuses on the case of matrices over a PID. We explore the relationship — with respect to computation of echelon forms — between the fraction field of a PID and the residue class ring of a PID for a well chosen residue. The primary motivation for this exercise is to develop techniques for avoiding the potential problem of intermediate expression swell when working over a PID such as $\mathbb{Z}$ or $\mathbb{Q}[x]$. Sundry useful facts are recalled and their usefulness to the design of effective algorithms is exposed. The main result is to show how to recover an echelon form over a PID by computing, in a fraction free way, an echelon form over the fraction field thereof. This leads to an efficient method for solving a system of linear diophantine equations over $\mathbb{Q}[x]$, a ring with potentially nasty expression swell.

Chapter 6: Hermite Form over $\mathbb{Z}$ An asymptotically fast algorithm is described and analysed under the bit complexity model for recovering a transformation matrix to the Hermite form of an integer matrix. The transform is constructed in two parts: the first r rows (what we call a solution to the extended matrix gcd problem) and last r rows (a basis for the row null space) where r is the rank of the input matrix. The algorithms here are based on the fraction free echelon form algorithms of Chapter 2 and the algorithm for modular computation of a Hermite form of a square nonsingular integer matrix developed in Chapter 5.

Chapter 7: Diagonalization over Rings An asymptotically fast algorithm is described for recovering the canonical Smith form of a matrix over PIR. The reduction proceeds in several phases. The result is first given for a square input matrix and then extended to rectangular. There is an important link between this chapter and chapter 3. On the one hand, the extension of the Smith form algorithm to rectangular matrices depends essentially on the algorithm for minimal echelon form presented in Chapter 3. On the other hand, the algorithm for minimal echelon form depends essentially on the square matrix Smith form algorithm presented here.

Chapter 8: Smith Form over $\mathbb{Z}$ An asymptotically fast algorithm is presented and analysed under the bit complexity model for recovering pre- and post-multipliers for the Smith form of an integer matrix. The theory of algebraic preconditioning — already well exposed in the literature — is adpated to get an asymptotically fast method of constructing a small post-multipler for an input matrix with full column

rank. The algorithms here make use of the fraction free echelon form algorithms of Chapter 2, the integer Hermite form algorithm of Chapter 6 and the algorithm for modular computation of a Smith form of a square nonsingular integer matrix of Chapter 7.

Chapter 9: Similarity over a Field Fast algorithms for recovering a transform matrix for the Frobenius form are described. This chapter is essentially self contained. Some of the techniques are analogous to the diagonalization algorithm of Chapter 7.

to

Susanna Balfegó Vergés

Chapter 3

Triangularizaton over Rings

The previous chapter considered the fraction free computation of echelon forms over a field. The algorithms there exploit a feature special to fields — every nonzero element is a divisor of one. In this chapter we turn our attention to computing various echelon forms over a PIR, including the Hermite form which is canonical over a PID. Here we need to replace the division operation with Gcdex. This makes the computation of a single unimodular transform for achieving the form more challenging. An additional issue, especially from a complexity theoretic point of view, is that over a PIR an echelon form might not have a unique number of nonzero rows — this is handled by recovering echelon and Hermite forms with minimal numbers of nonzero rows. The primary purpose of this chapter is to establish sundry complexity results in a general setting — the algorithms return a single unimodular transform and are applicable for any input matrix over any PIR (of course, provided we can compute the basic operations).

Let R be a PIR. Every $A \in \mathsf{R}^{n\times m}$ is left equivalent to an $H \in \mathsf{R}^{n\times m}$ that satisfies:

(r1) Let r be the number of nonzero rows of H. Then the first r rows of H are nonzero. For $0 \leq i \leq r$ let $H[i, j_i]$ be the first nonzero entry in row i. Then $0 = j_0 < j_1 < j_2 < \ldots < j_r$.

(r2) $H[i, j_i] \in \mathcal{A}(\mathsf{R})$ and $H[k, j_i] \in \mathcal{R}(\mathsf{R}, H[i, j_i])$ for $1 \leq k < i \leq r$.

(r3) H has a minimal number of nonzero rows.

Using these conditions we can distinguish between four forms as in Table 3. This chapter gives algorithms to compute each of these forms. The cost estimates are given in terms of number of basic operations and

Form	r1	r2	r3	Cost
echelon	•			$nmr^{\theta-2} + nr^{\theta-2}(\log 2n/r)$
minimal echelon	•		•	$nmr^{\theta-2} + nr^{\theta-2}(\log n)$
Hermite	•	•		$nmr^{\theta-2} + nr^{\theta-2}(\log 2n/r)$
minimal Hermite	•	•	•	$nmr^{\theta-2} + nr^{\theta-2}(\log n)$

Table 3.1: Non canonical echelon forms over a PIR

include the time to recover a unimodular transform matrix $U \in \mathsf{R}^{n\times m}$ such that $UA = H$. Some of our effort is devoted to expelling some or all of the logorithmic factors in the cost estimates in case a complete transform matrix is not required.

Section 3.1 shows how to transform A to echelon or minimal echelon form. Section 3.2 shows how to transform an echelon form to satisfy also (r2). Section 3.3 simply combines the results of the previous two sections and gives algorithms for the Hermite and minimal Hermite form.

The Hermite form — the classical canonical form for left equivalence of matrices over a PID — is a natural generalization of the Gauss Jordon form over a field. If R is a field, and we choose $\mathcal{A}(\mathsf{R}) = \{1\}$ and $\mathcal{R}(\mathsf{R}, *) = \{0\}$, then H is the GaussJordan form of A. Over a PIR the Hermite form is not a canonical, and condition (r3) is motivated because different echelon forms can have different number of nonzero rows.

Notes

Existence and uniqueness of the Hermite form over a PID is a classical result, see (Newman, 1972). The following code, applicable over a PIR, uses $O(nmr)$ basic operations of type {Arith, Gcdex} to transform an $n \times m$ matrix A to echelon form in place.

```
$r := 0$;
for $k$ to $m$ do
    for $i$ from $r+2$ to $n$ do
        $(g, s, t, u, v) :=$ Gcdex$(A[r+1, k], A[i, k])$;
        $\begin{bmatrix} A[r+1,*] \\ A[i,*] \end{bmatrix} := \begin{bmatrix} s & t \\ u & v \end{bmatrix} \begin{bmatrix} A[r+1,*] \\ A[i,*] \end{bmatrix}$;
    od;
    if $A[r+1, k] \neq 0$ then $r := r+1$ fi
od;
```

Condition (r2) can be satisified using an additional $O(mr^2)$ operation of type {Unit, Quo} by continuing with the following code fragment.

```
$r := 1$;
for $k$ to $m$ do
    if $A[r, k] \neq 0$ then
        $A[r, *] :=$ Unit$(A[r, k])A[r, *]$;
        for $i$ to $r-1$ do
            $q :=$ Quo$(A[i, k], A[r, k])$;
            $A[i, *] := A[i, *] - qA[r, *]$
        od;
        $r := r+1$;
    fi
od;
```

A transform U can be recovered by working with the augmented matrix $\left[\begin{array}{c|c} A & I_n \end{array}\right]$. Unfortunately, the cost increases (when $n > m$) to $O(nmr + n^2 r)$ basic operations.

Asymptotically fast algorithms for transforming A to upper triangular form (but not necesarily echelon form) are given by Hafner and McCurley (1991). An important feature of the algorithms there is that a unimodular transformation matrix U is also recovered. When $n > m$ the cost estimate is $O(nm^{\theta-2}(\log 2n/m))$ which is almost linear in n (compare with the bound $O(nmr + n^2 r)$ derived above). On the one hand, the details of obtaining an echelon form (as opposed to only upper triangular) are not dealt with in (Hafner and McCurley, 1991). On the other hand, with modest effort the echelon form algorithm of Keller-Gehrig (1985), see also (Bürgisser *et al.*, 1996, Section 16.5), for matries over fields can be modified to work over more general rings. Essentially, section 3.1 synthesises all these results and incorporates some new ideas to get an algorithm for echelon form over a principal ideal ring that admits a good complexity bound in terms of also r.

A different solution for the problem in Section 3.2 is given in (Storjohann and Labahn, 1996).

Left Transforms

Let $A \in \mathsf{R}^{n \times m}$. A unimodular $U \in \mathsf{R}^{n \times n}$ is called a *left transform* for A. Assume now that A has trailing n_1 rows zero, n_1 chosen maximal. If U can be written as

$$U = \left[\begin{array}{c|c} * & \\ \hline & I_{n_1} \end{array}\right],$$

then we call U a *principal left transform* for A. Note that if U is a principal left transform for A, then the principal $n_1 \times n_1$ submatrix of U is a principal transform for the principal $n_1 \times m$ submatrix of A.

3.1 Transformation to Echelon Form

Let R be a PIR and $A \in \mathsf{R}^{n \times m}$. Our goal is to compute an echelon form T of A. We begin by outlining the approach. The first result we need is a subroutine transforming to echelon form an input matrix which has a special shape and at most twice as many rows as columns. This is shown in Figure 3.1. A complexity of $O(m^\theta)$ basic operations

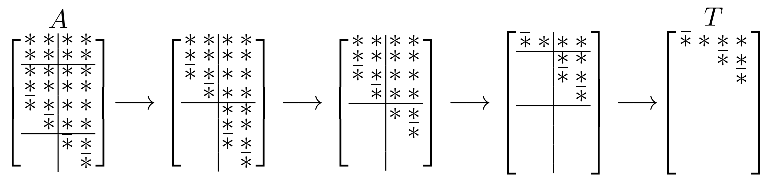


Figure 3.1: Subroutine for transformation to echelon form

is achieved by using a divide and conquer paradigm — the problem is reduced to four subproblems of half the size. Next we give an algorithm that sweeps across an input matrix from left to right using the subroutine sketched in Figure 3.1. This is shown in Figure 3.2. The complexity of the left-to-right method is $O(nm^{\theta-1})$ basic operations. Finally, we give an algorithm the sweeps across the input matrix from bottom to top, applying the left-to-right method on a contiguous slice of rows. This is shown in Figure 3.3. By carefully specifying the number of rows to

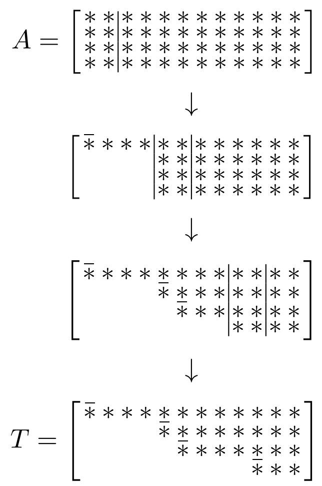


Figure 3.2: Left-to-right transformation to echelon form

consider for each slice, we can derive a complexity $O(nmr^{\theta-2})$ basic

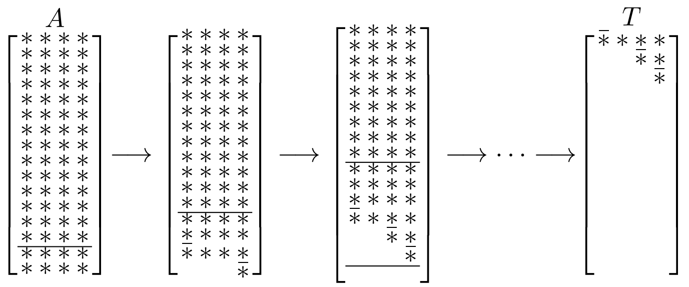


Figure 3.3: Bottom-to-top transformation to echelon form

operations for the bottom-to-top method. A subtlety we face is that r may not unique with respect to A but only with respect to the exact method used to compute T. (This subtlety disappears when R is an integral domain since then r is the unique rank of A.) We deal with this problem by ensuring that, when applying the bottom-to-top method, the number of nonzero rows the succesive echelon forms we compute is nondecreasing. Furthermore, if we want a minimal echelon form then we ensure that each echelon form computed is minimal.

Now we present the algorithms and fill in all the details, including the recovery of transformation matrices. Lemma 3.1 gives the algorithm sketched in Figure 3.1. The proof is similar to (Bürgisser *et al.*, 1996, Proposition 16.8) for a matrix over a field, attributed to Schönhage (1973).

Lemma 3.1. *Let $A \in \mathsf{R}^{(t+k)\times m}$ have last k rows nonzero and in echelon form, $0 \le t \le m$, $0 \le k \le m$. An echelon form T of A with at least k nonzero rows together with a principal left transform U such that $UA = T$ can be recovered in $O(m^\theta)$ basic operation of type* {Arith, Gcdex}.

Corollary 3.2. *Let $\mathsf{R} = \mathbb{Z}/(N)$. The complexity bound of Lemma 3.1 becomes $O(m^\theta(\log\beta) + m^2(\log m)\,\mathsf{B}(\log\beta))$ word operations where $\beta = mN$.*

Proof. Let $f(m)$ be the number of basic operations required to compute a U and T which satisfy the requirements of the lemma. By augmenting the input matrix with at most $m-1$ zero columns we may assume without loss of generality that m is a power of two. This shows it will be sufficient to bound $f(m)$ for m a power of two. If $m = 1$ then $t + k \le 2$ and the problem reduces to at most a single basic operation of type Gcdex. This shows $f(1) = 1$.

Now assume $m > 1$, m a power of two. Partition the input matrix as

$$A = \left[\begin{array}{c|c} * & * \\ \hline * & * \\ \hline \bar{*}_1 & * \\ & \bar{*} \end{array}\right]$$

where each block has column dimension $m/2$, the principal horizontal slice comprises the first $\lfloor t/2 \rfloor$ rows of A, the middle horizontal slice the next $\lceil t/2 \rceil$ rows and $\bar{*}$ denotes a block which is in echelon form with no zero rows. (One of the $\bar{*}$ blocks is subscripted to allow refering to it later.) Recursively compute a principal left transform U_1 such that

$$\overset{U_1}{\left[\begin{array}{c|c|c} I & & \\ \hline & * & \\ \hline & & I \end{array}\right]} \overset{A}{\left[\begin{array}{c|c} * & * \\ \hline * & * \\ \bar{*}_1 & * \\ \hline & \bar{*} \end{array}\right]} = \overset{A_1}{\left[\begin{array}{c|c} * & * \\ \hline \bar{*}_2 & * \\ & *_1 \\ \hline & \bar{*} \end{array}\right]}$$

with $\bar{*}_2$ having at least as many rows as $\bar{*}_1$. Recover the last $m/2$ columns of A_1 in $O(m^\theta)$ arithmetic operations by premultiplying by U_1. Partition

A_1 anew as

$$A_1 = \left[\begin{array}{c|c} * & * \\ \bar{*}_2 & * \\ \hline & *_1 \\ & \bar{*} \end{array}\right].$$

Since the row dimension of $\bar{*}_2$ in A_2 is at least that of $\bar{*}_1$ in A_1, the block $*_1$ has at most $\lceil t/2 \rceil$ rows. In particular, since $t \le m$ and m is even also $\lceil t/2 \rceil \le m/2$.

Recursively compute a principal left transform U_2 such that

$$\overset{U_2}{\left[\begin{array}{cc|c} I & & \\ & I & \\ \hline & & * \end{array}\right]} \overset{A_1}{\left[\begin{array}{c|c} * & * \\ \bar{*} & * \\ \hline & * \\ & \bar{*} \end{array}\right]} = \overset{A_2}{\left[\begin{array}{c|c} * & * \\ \bar{*} & * \\ \hline & \bar{*} \end{array}\right]}.$$

At this point the sum of the row dimensions of the blocks labelled $\bar{*}$ in A_2 is at least k. The next two transformations are analogous. The complete transformation sequence is:

$$\overset{A}{\left[\begin{array}{cc} * & * \\ * & * \\ \bar{*} & * \\ & \bar{*} \end{array}\right]} \xrightarrow{U_1} \overset{A_1}{\left[\begin{array}{cc} * & * \\ \bar{*} & * \\ & * \\ & \bar{*} \end{array}\right]} \xrightarrow{U_2} \overset{A_2}{\left[\begin{array}{cc} * & * \\ \bar{*} & * \\ & \bar{*} \end{array}\right]} \xrightarrow{U_3} \overset{A_3}{\left[\begin{array}{cc} \bar{*} & * \\ & * \\ & \bar{*} \end{array}\right]} \xrightarrow{U_4} \overset{T}{\left[\begin{array}{cc} \bar{*} & * \\ & \bar{*} \end{array}\right]}.$$

T is in echelon form with at least k nonzero rows. Recover U as $U_4U_3U_2U_1$. This shows that $f(m) \le 4f(m/2) + O(m^\theta)$. The result follows. $\square$

Now we extend the previous result to handle efficiently the case when $m > n$. This situation is sketched in Figure 3.2. The next lemma and subsequent proposition actually present two algorithms which recover an echelon form satisfying one of two conditions (a) or (b). The "(b)" algorithms will depend on the "(a)" algorithms but not vice versa. Lemma 3.3 is similar to (Bürgisser *et al.*, 1996, Proposition 16.11) for a matrix over a field, attributed to Keller-Gehrig (1985).

Lemma 3.3. *Let $A \in \mathsf{R}^{n\times m}$. An echelon form T of A together with a principal left transform U such that $UA = T$ can be recovered in $O(mn^{\theta-1})$ basic operations of type* {Arith, Gcdex}. *The user may choose to have either (but not necessarily both) of the following conditions satisfied:*

- *(a) T will have at least k nonzero rows where k is maximal such that the last k rows of A are nonzero and in echelon form.*
- *(b) T will be a minimal echelon form of A.*

Achieving condition (b) incurs an additional cost of $O(n^\theta(\log n))$ basic operations of type {Arith, Gcdex, Stab}.

Corollary 3.4. *Let $\mathsf{R} = \mathbb{Z}/(N)$. The complexity bounds of Lemma 3.3 become $O(mn^{\theta-1}(\log\beta) + mn(\log n)\,\mathrm{B}(\log\beta))$ and $O(n^\theta(\log n)(\log\beta))$ word operations where $\beta = nN$.*

Proof. We first demonstrate an algorithm to achieve condition (a). By augmenting the input matrix with at most $m-1$ columns of zeroes we may assume that m is a multiple of n, say $m = ln$. Partition the input matrix as $A = \begin{bmatrix} A_1 & A_2 & \cdots & A_l \end{bmatrix}$ where each block is $n \times n$. Perform the following.

```
$z := n$;    $U := I_n$;
for $i$ from 1 to $l$ do
    $B :=$ the last $z$ rows of $UA_i$;
    $V :=$ a principal left transform such that $VB$ is in echelon form;
    $U := \begin{bmatrix} I_{n-z} & \\ & V \end{bmatrix} U$;
    $z :=$ the number of trailing zero rows in $VB$;
    # Now the first $in$ columns of $UA$ are in echelon form.
od;
```

Upon completion, U is a principal left transform such that UA is in echelon form. By Lemma 3.1, each iteration of the loop costs $O(n^\theta)$ basic operations. The cost bound follows. The claim about the number of nonzero rows can be shown using a similar argument as in Lemma 3.1.

We now demonstrate an algorithm to achieve condition (b). Transform A^T to echelon form R by repeatedly applying Lemma 3.1 to the maximal number of trailing nonzero rows (maximal so that the submatrix is a valid input for the Lemma). This costs at most $O(mn^{\theta-1})$ basic operations. Use Lemma 7.14 to recover a principal right transform V such that A^TV is left equivalent to Smith form[1]. This costs $O(n^\theta(\log n))$ basic operations. Then A^TV has a maximal number of trailing zero columns. Compute U using the algorithm described above for achieving condition (a) with input V^TA and return UV^T. □

[1]To expel this forward reference would require heroic efforts.

Now we put everything together to get the algorithm sketched in Figure 3.3.

Proposition 3.5. *Let $A \in \mathsf{R}^{n\times m}$. An echelon form T of A can be recovered in $O(nmr^{\theta-2})$ basic operations of type* {Arith, Gcdex} *where r is the number of nonzero rows in T. An $E \in \mathsf{R}^{r\times n}$ such that EA equals the first r rows of T can be recovered in the same time. The user may choose to have either (but not necessarily both) of the following conditions satisfied:*

- *(a) $r \geq k$ where k is maximal such that the last k rows of A are nonzero and in echelon form.*
- *(b) T will be a minimal echelon form of A.*

Achieving condition (b) incurs an additional cost of $O(nr^{\theta-1}(\log r))$ basic operations of type {Arith, Gcdex, Stab}.

Corollary 3.6. *Let $\mathsf{R} = \mathbb{Z}/(N)$. The complexity bounds of Proposition 3.5 become $O(nmr^{\theta-2}(\log\beta) + nm(\log r)\,\mathrm{B}(\log\beta))$ and $O(nr^{\theta-1}(\log r)(\log\beta))$ word operations where $\beta = rN$.*

Proof. Perform the following:

```
$T :=$ a copy of $A$;   $z := 1$;   $r := 1$;
for $i$ from 1 do
    $d := \min(\max(r, 1), n - z)$;
    # Let $(T_i, r_i, z_i, d_i)$ be a copy of $(T, r, z, d)$ at this point.
    $z := z + d$;
    $B :=$ the last $z$ rows of $T$;
    $V :=$ a principal left transform such that $VB$ is in echelon form;
    $U_i := \begin{bmatrix} I_{n-z} & \\ & V \end{bmatrix}$;
    $T := UT$;
    $r :=$ the number of nonzero rows in $VB$;
    if $z = n$ then break fi;
od;
```

Induction on i shows that

$$\overset{U_i}{\left[\begin{array}{c|c|c} I & & \\ \hline & * & \\ \hline & & I \end{array}\right]} \overset{T_i}{\left[\begin{array}{c} * \\ \hline * \\ \bar{*} \\ \hline \\ \end{array}\right]} = \overset{T_{i+1}}{\left[\begin{array}{c} * \\ \hline \bar{*} \\ \\ \\ \end{array}\right]}$$

where the label $\bar{*}$ denotes a block which is in echelon form with no zero rows and

- the principal slice of T_i and T_{i+1} has $\max(0, n-z-d_i)$ rows,
- the middle slice of T_i has row dimension $d_i + r_i$
- $\bar{*}$ in T_i has r_i rows,
- $\bar{*}$ in T_{i+1} has r_{i+1} rows.

Note that $d_i + r_i \le 2d_i$. By Lemma 3.3, there exists an absolute constant c such that U_i and T_{i+1} can be recovered in fewer than $cmd_i^{\theta-1}$ basic operations. Both the algorithm supporting Lemma 3.3a and Lemma 3.3b ensure that $r_{i+1} \ge r_i$.

On termination T is in echelon form with r nonzero rows. The amount of progress made during each loop iteration is d_i. The average cost per unit of progress (per row zeroed out) is thus $cmd_i^{\theta-2}$. Since $d_i \le r$, and the loop terminates when $\sum_i d_i = n-1$, the overall running time is as stated.

Finally, E can be recovered in the allotted time by computing the product $\left[\begin{array}{c|c} I_r & 0 \end{array}\right] U_i \cdots U_2 U_1$ from left to right. □

The proof of Propsosition 3.7 is similar to (Hafner and McCurley, 1991, Theroem 3.1). Here we return an echelon form (instead of just triangular as they do) and analyse under the tri-paramater model (instead of just n and m).

Proposition 3.7. *Let $A \in \mathsf{R}^{n\times m}$. An echelon form $T \in \mathsf{R}^{n\times m}$ together with a principal left transform $U \in \mathsf{R}^{n\times n}$ such that $UA = T$ can be recovered in $O(nr^{\theta-1}(\log 2n/r) + nmr^{\theta-2})$ basic operations of type* {Arith, Gcdex} *where r is the number of nonzero rows in T. The condition that T should be a minimal echelon form can be achieved at an additional cost of $O(nr^{\theta-1}(\log r))$ basic operations of type* {Arith, Gcdex, Stab}.

Corollary 3.8. *Let $\mathsf{R} = \mathbb{Z}/(N)$. The complexity bounds of Proposition 3.7 become $O(nmr^{\theta-2}(\log 2n/r)(\log\beta) + nm(\log n)\,\mathrm{B}(\log\beta))$ and $O(nr^{\theta-1}(\log r)(\log\beta))$ word operations where $\beta = rN$.*

Proof. Below we describe an algorithm for recovering a U and T which satisfy the requirements of the proposition. For $A \in \mathsf{R}^{n\times m}$, let $r(A)$ denote the number of nonzero rows in the echelon form produced by the aformentioned algorithm on input $A \in \mathsf{R}^{n\times m}$. The function $r(A)$ is well defined because the algorithm is deterministic.

For $r \ge 0$, let $f_{m,r}(n)$ be a bound on the number of basic operations required by the algorithm with input from $\{A \in \mathsf{R}^{n\times m} \mid r(A) \le r\}$. By augmenting A with at most $n-1$ rows of zeroes, we may assume that n is a power of two. This shows it will be sufficient to bound $f_{m,*}(n)$ for n a power of two. If $n = 1$ there is nothing to do; choose $U = I_1$. This shows $f_{m,*}(1) = 0$.

Now assume $n > 1$, n a power of two. Let A_1 be the first and A_2 the last $n/2$ rows of A. Recursively (two recursive calls) compute a principal left transform U_1 such that

$$\overset{U_1}{\left[\begin{array}{c|c} * & \\ \hline & * \end{array}\right]} \left[\begin{array}{c} A_1 \\ \hline A_2 \end{array}\right] = \left[\begin{array}{c} T_1 \\ 0 \\ \hline T_2 \\ 0 \end{array}\right]$$

where T_i is in echelon form with $r(A_i)$ rows, $i = 1, 2$. By permuting the blocks T_1 and T_2 (if necessary) we may assume that T_2 has at least as many rows as T_1. Use Lemma 3.3 to compute a principal left transform U_2 such that

$$\overset{U_2}{\left[\begin{array}{cc|cc} * & & * & \\ & I & & \\ \hline * & & * & \\ & & & I \end{array}\right]} \left[\begin{array}{c} T_1 \\ 0 \\ \hline T_2 \\ 0 \end{array}\right] = T$$

where T is in echelon form. If T should be a minimal echelon form, then use the algorithm supporting Lemma 3.3b. Otherwise, use the algorithm supporting Lemma 3.3a. In either case, T will have at least $\max(r(A_1), r(A_2))$ rows.

This shows that $f_{m,r}(n) \le 2f_{m,r}(n/2) + O((n+m)r^{\theta-1})$. This resolves to $f_{m,r}(n) \le n/\bar{r} f_{m,r}(\bar{r}) + O(nr^{\theta-1}(\log 2n/r) + nmr^{\theta-2})$ where $\bar{r}$ is the smallest power of two such that $\bar{r} \ge r$. From Lemma 3.3a we may deduce that $f_{m,r}(\bar{r}) \le O(mr^{\theta-1})$ and from Lemma 3.3b that $f_{m,r}(\bar{r}) \le O(mr^{\theta-1} + r^{\theta}(\log r))$. The result follows. □

3.2 The Index Reduction Transform

Let $A \in \mathsf{R}^{n\times n}$ be upper triangular with diagonal entries nonzero and in $\mathcal{A}(\mathsf{R})$. The motivating application of the algorithm developed in this section to compute a unit upper triangular $U \in \mathsf{R}^{n\times n}$ such that UA is in Hermite form.

For $1 \le i < j \le n$, let $\phi_{i,j}(a)$ be a prescribed function on R which satisfies $\phi_{i,j}(a + \phi_{i,j}(a)A[j,j]) = 0$ for all $a \in \mathsf{R}$. Different applications of the index reduction transform will call for different choices of the $\phi_{*,*}$.

Definition 3.9. *An* index k reduction transform for A with respect to a function family $\phi_{*,*}$ *is a unit upper triangular matrix U which satisfies and can be written as*

$$\begin{array}{ccc} U & A & H \\ \left[\begin{array}{c|c} I_{n-k} & * \\ \hline & * \end{array}\right] & \left[\begin{array}{c|c} * & * \\ \hline & * \end{array}\right] = & \left[\begin{array}{c|c} * & * \\ \hline & * \end{array}\right] \end{array}$$

where the block decomposition is conformal and $\phi_{i,j}(H[i,j]) = 0$ for $1 \le i < j$, $n-k \le j \le n$.

We say "reduction transform" to mean "index 0 reduction transform".

Proposition 3.10. *A reduction transform for $A \in \mathbb{Z}^{n\times n}$ with respect to $\phi_{*,*}$ can be computed in $O(n^\theta)$ basic operations of type* Arith *plus fewer than nm calls to $\phi_{*,*}$.*

Corollary 3.11. *Let $\mathsf{R} = \mathbb{Z}/(N)$. If the cost of one call to $\phi_{*,*}$ is bounded by $O(\mathrm{B}(\log N))$ bit operations, the complexity bound of Propostion 3.10 becomes $O(n^\theta(\log\beta) + n^2(\log n)\,\mathrm{B}(\log\beta))$ word operations where $\beta = nN$.*

Proof. Computing an index k reduction transform for an $n \times 1$ matrix requires $n - k < n$ applications of $\phi_{*,*}$. Let $f_n(k)$ be the number of basic operations (not counting applications of $\phi_{*,*}$) required to compute an index k transform for an $m \times m$ matrix A where $m \le n$. Then $f_n(1) = 0$. The result will follow if we show that for any indices k_1, k_2 with $k_1 + k_2 = k > 1$, we have $f_n(k) \le f_n(k_1) + f_n(k_2) + O(nk^{\theta-2})$.

Compute an index k_1 transform U_1 for the principal $(n-k_2)\times(n-k_2)$ submatrix of A. Let A_2 be the last m_2 columns of $\mathrm{diag}(U_1, I_{k_2})A$. Compute an index k_2 transform U_2 for A_2. Then

$$\begin{array}{cccc} U_2 & \mathrm{diag}(U_1, I_{k_2}) & A & H \\ \left[\begin{array}{c|c|c} I & & * \\ \hline & I & * \\ \hline & & * \end{array}\right] & \left[\begin{array}{c|c|c} I & * & \\ \hline & * & \\ \hline & & I_{k_2} \end{array}\right] & \left[\begin{array}{c|c|c} * & * & * \\ \hline & * & * \\ \hline & & * \end{array}\right] = & \left[\begin{array}{c|c|c} * & * & * \\ \hline & * & * \\ \hline & & * \end{array}\right]. \end{array}$$

Note that computing the product U_2U_1 requires no basic operations. It is clear that A_2 can be recovered in the allotted time (cf. Lemma 2.5). $\square$

Our first example is the Hermite reduction.

Example 3.12. *Let $T \in \mathsf{R}^{n\times n}$ be upper triangular with diagonal entries in $\mathcal{A}(\mathsf{R})$. For $1 \le i < j \le n$, define $\phi_{i,j}$ by $\phi_{i,j}(a) \mapsto -\mathrm{Quo}(a, T[j,j])$. If U is a reduction transform for T, then UA is in Hermite form. The following example is over $\mathbb{Z}$.*

$$\begin{array}{ccc} U & A & H \\ \begin{bmatrix} 1 & -1 & 14 & -138 \\ & 1 & -26 & 259 \\ & & 1 & -10 \\ & & & 1 \end{bmatrix} & \begin{bmatrix} 1 & 5 & 38 & 31 \\ & 5 & 79 & 85 \\ & & 3 & 63 \\ & & & 6 \end{bmatrix} = & \begin{bmatrix} 1 & 0 & 1 & 0 \\ & 5 & 1 & 1 \\ & & 3 & 3 \\ & & & 6 \end{bmatrix} \end{array}$$

The reductions of the next two examples are used in Section 8.1. In the Hermite reduction off-diagonal entries were reduced modulo the diagonal entry in the same column. The next examples perform more general reductions: off-diagonal entry $A_{i,j}$ will be reduced modulo $E_{i,j}$ where E is a specified matrix over R. The next example shows how to effect a certain reduction of a lower triangular matrix via column operations by passing over the transpose.

Example 3.13. *Let $L \in \mathsf{R}^{n\times n}$ be unit lower triangular and $E \in \mathsf{R}^{n\times n}$ have each entry from $\mathcal{A}(\mathsf{R})$. For $1 \le i < j \le n$, define $\phi_{i,j}$ by $\phi_{i,j}(a) \mapsto -E^T_{i,j}\mathrm{Quo}(a, E^T_{i,j})$. If V^T is a reduction transform for L^T, then*

- *V is unit lower triangular with $V_{i,j}$ a multiple of $E_{i,j}$, and*
- *LV is unit lower triangular with $(LV)_{i,j} \in \mathcal{R}(\mathsf{R}, E_{i,j})$.*

The next example shows how to effect a certain reduction of an upper triangular matrix via column. Allowing ourselves the introduction of one more Greek letter, we write $\Psi(A)$ to mean the matrix obtained from A be reversing the order of rows and columns and transposing. For example, $\Psi(\Psi(A)) = A$, and if A is upper triangular, then $\Psi(A)$ will also be upper triangular with $\Psi(A)_{i,j} = A_{n-i,n-j}$.

Example 3.14. *Let $T \in \mathsf{R}^{n\times n}$ be unit upper triangular and $E \in \mathsf{R}^{n\times n}$ have each entry from $\mathcal{A}(\mathsf{R})$. For $1 \le i < j \le n$, define $\phi_{i,j}$ by $\phi_{i,j}(a) \mapsto -\phi(E)_{i,j}\mathrm{Quo}(a, \Psi(E)_{i,j})$. If $\Psi(V)$ is a reduction transform for $\Psi(T)$, then*

- *V is unit upper triangular with $V_{i,j}$ a multiple of $E_{i,j}$, and*
- *TV is unit upper triangular with $(TV)_{i,j} \in \mathcal{R}(\mathsf{R}, E_{i,j})$.*

3.3 Transformation to Hermite Form

Proposition 3.15. *Let $A \in \mathsf{R}^{n\times m}$. There exists an algorithm that recovers a Hermite form H of A together with an $E \in \mathsf{R}^{r\times m}$ such that EA equals the nonzero rows of A. The algorithm uses basic operations of type* {Arith, Gcdex, Unit, Quo}.

1. *The cost of producing H and E is $O(nmr^{\theta-2})$ basic operations.*
2. *A unimodular $U \in \mathsf{R}^{n\times n}$ that satisfies $UA = H$ can be recovered in $O(nr^{\theta-1}(\log 2n/r) + nmr^{\theta-2})$ basic operations.*

The condition that r should be minimal can met at an additional cost of $O(nr^{\theta-1}(\log r))$ basic operations of type $\{Arith, \text{Gcdex}, \text{Stab}\}$.

Corollary 3.16. *Let $\mathsf{R} = \mathbb{Z}/(N)$. The complexity bounds of Proposition 3.15 become $O(nmr^{\theta-2}(\log\beta) + nm(\log r)\,\mathrm{B}(\log\beta))$, $O(nmr^{\theta-2}(\log 2n/r)(\log\beta) + nm(\log n)\,\mathrm{B}(\log\beta))$ and $O(nmr^{\theta-1}(\log r))$ word operations where $\beta = rN$.*

Proof. The algorithm works by computing a number of intermediate matrices U_1, U_2, U_3 from which U and H will be recovered. These satsify

$$\left[\begin{array}{c|c} U_3 & \\ \hline & I_{n-r}\end{array}\right] \left[\begin{array}{c|c} U_2 & \\ \hline & I_{n-r}\end{array}\right] \overset{U_1}{\left[\begin{array}{c|c} * & * \\ \hline * & *\end{array}\right]} \overset{A}{\left[\begin{array}{c} * \\ \hline * \end{array}\right]} = \overset{H}{\left[\begin{array}{c} * \\ \hline \end{array}\right]} \tag{3.1}$$

where the block decompostion is conformal. For part 1. of the proposition only the first r rows of U_1 are recovered. The algorithm has five steps:

1. Recover an echelon form $T \in \mathsf{R}^{n\times m}$ together with a unimodular $U_1 \in \mathsf{R}^{n\times n}$ such that $U_1A = T$.
2. Let $(j_1, j_2, \ldots, j_r)$ be such that $T[i, j_i]$ is the first nonzero entry in row i of T, $1 \le i \le r$. Let $\bar{T}$ be the submatrix of T comprised of first r rows and columns $j_1, j_2, \ldots, j_r$.
3. Set $U_2 \in \mathsf{R}^{r\times r}$ to be the diagonal matrix which has $U_2[i,i] = \mathrm{Unit}(\bar{T}[i,i])$ for $1 \le i \le r$. Then each diagonal entry in $U_2\bar{T}$ belongs to assoc(R).

4. Recover a unit upper triangular $U_3 \in \mathsf{R}^{r\times r}$ such that $U_3U_2\bar{T}$ in Hermite form.
5. Set $U = \mathrm{diag}(U_3U_2, I_{n-r})U_1$. Then $UA = H$ with H in Hermite form.

Correctness is obvious. The cost bound for steps 2, 3 and 5 is clear; that for step 1 follows from Propositions 3.5 and 3.7 and for step 4 from Prospostion 3.10 and Example 3.12. $\square$

Chapter 7

Diagonalization over Rings

An asymptotically fast algorithm is described for recovering the canonical Smith form of a matrix over PIR. The reduction proceeds in several phases. The result is first given for a square input matrix and then extended to rectangular. There is an important link between this chapter and chapter 3. On the one hand, the extension of the Smith form algorithm to rectangular matrices depends essentially on the algorithm for minimal echelon form presented in Chapter 3. On the other hand, the algorithm for minimal echelon form depends essentially on the square matrix Smith form algorithm presented here.

Let R be a PIR. Corresponding to any $A \in \mathsf{R}^{n\times m}$ there exist unimodular matrices U and V over R such that

$$S = UAV = \begin{bmatrix} s_1 & & & & \\ & s_2 & & & \\ & & \ddots & & \\ & & & s_r & \\ & & & & \\ & & & & \end{bmatrix}$$

with each s_i nonzero and s_i a divisor of s_{i+1} for $1 \leq i \leq r-1$. The matrix S is the *Smith canonical form* of A. The diagonal entries of S are unique up to associates. We show how to recover S together with unimodular $U \in \mathsf{R}^{n\times n}$ and $V \in \mathsf{R}^{m\times m}$ such that $UAV = S$ in $O\tilde{~}(nmr^{\theta-2})$ basic operations of type {Arith, Gcdex, Stab}.

The algorithm proceeds in two stages. The first stage, shown in Figure 7.1, is transform an upper triangular input matrix to be upper bi-diagonal. The algorithms for band reduction are presented in Section 7.1. These are iterated $O(\log n)$ times until the input matrix is upper bi-diagonal. The second stage is to transform the bi-diagonal matrix to

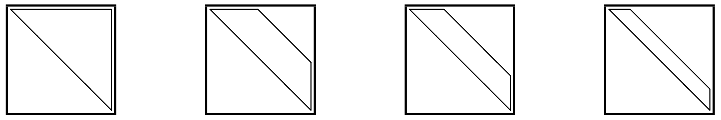


Figure 7.1: Banded Reduction

Smith form. This is presented in Section 7.3. The algorithm there depends on a subroutine, presented in Section 7.2, for transforming an already diagonal matrix to Smith form. Finally, Section 7.4 combines the results of the previous 3 section, together the algorithms of the Chapter 3 for triangularizing matrices, to get the complete Smith form algorithm.

Notes

Existence of the form was first proven by Smith (1861) for integer matrices. Existence and uniqueness over a PID is a classical result. Newman (1972) gives a lucid treatmeant. Existence and uniqueness over a PIR follows from Kaplansky (1949), see also (Brown, 1993).

Most work on computing Smith forms has focused on concrete rings such as $\mathsf{R} = \mathbb{Z}$. We postpone the discussion of this until Chapter 8.

Transforms

Let $A \in \mathsf{R}^{n\times m}$. Let $U \in \mathsf{R}^{n\times n}$ and $V \in \mathsf{R}^{m\times m}$ be unimodular. We call U a *left transform*, V a *right transform* and (U, V) a *transform* for A. The algorithms of the next section work by applying a sequence of transforms to the work matrix A as follows: $A \leftarrow UAV$.

Assume now that A has all entries in the last n_1 rows and m_1 columns zero, n_1 and m_1 chosen maximal. A *principal transform* for A is a

transform (U, V) which can be written as

$$\begin{array}{cccc} & U & A & V \\ A & \left[\begin{array}{c|c} * & \\ \hline & I_{n_1}\end{array}\right] & \left[\begin{array}{c|c} * & \\ \hline & \end{array}\right] & \left[\begin{array}{c|c} * & \\ \hline & I_{m_1}\end{array}\right]\end{array}.$$

Note that the principal $n - n_1$ submatrix of U is a left transform for the principal $n - n_1$ submatrix of A. A similar comment applies to V.

7.1 Reduction of Banded Matrices

A square matrix A is *upper b-banded* if $A_{ij} = 0$ for $j < i$ and $j \geq i + b$, that is, if A can be written as

$$A = \begin{bmatrix} * & \cdots & * & & & & \\ & \ddots & & \text{i>j} \;\iddots & & & \\ & & \ddots & & \ddots & & \\ & \text{j<i} & & \ddots & & \ddots & \\ & & & & \ddots & & * \\ & & & & & \ddots & \vdots \\ & & & & & & * \end{bmatrix}. \tag{7.1}$$

In this section we develop an algorithm which transforms A to an equivalent matrix, also upper banded, but with band about half the width of the band of the input matrix.

Our result is the following.

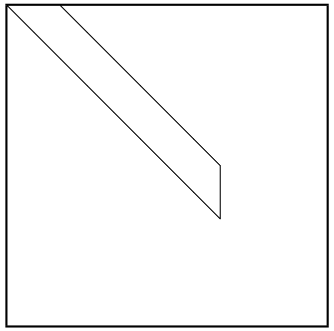
Proposition 7.1. *For $b > 2$, there exists an algorithm that takes as input an upper b-banded $A \in \mathsf{R}^{n\times n}$, and produces as output an upper $(\lfloor b/2 \rfloor + 1)$-banded matrix A' that is equivalent to A.*

1. *The cost of producing A' is bounded by $O(n^2 b^{\theta-2})$ basic operations.*
2. *A principal transform (U, V) satisfying $UAV = A'$ can be recovered in $O(n^\theta)$ basic operations.*

The algorithm uses basic operations of type {Arith, Gcdex}.

Corollary 7.2. *Let $\mathsf{R} = \mathbb{Z}/(N)$. The complexity bounds of Proposition 7.1 become $O(n^2 b^{\theta-2}(\log\beta) + n^2(\log b)\,\mathrm{B}(\log\beta))$ and $O(n^\theta(\log\beta) + n^2(\log n)\,\mathrm{B}(\log\beta))$ word operations where $\beta = nN$.*

Proof. By augmenting A with at most $2b$ rows and columns we may assume that A at least $2b$ trailing columns of zeroes. In what follows, we write $\mathrm{sub}[i,k] = \mathrm{sub}_A[i,k]$ to denote the the symmetric $k \times k$ submatrix of A comprised of rows and columns $i+1, \ldots, i+k$. Our work matrix, initially the input matrix A, has the form



Our approach is to transforms A to A' by applying (in place) a sequence of principal transforms to $\mathrm{sub}[is_1, n_1]$ and $\mathrm{sub}[(i+1)s_1 + js_2, n_2]$, where i and j are nonnegative integer parameters and

$$\begin{aligned} s_1 &= \lfloor b/2 \rfloor, \\ n_1 &= \lfloor b/2 \rfloor + b - 1, \\ s_2 &= b - 1, \\ n_2 &= 2(b-1). \end{aligned}$$

The first step is to convert the work matrix to an equivalent matrix but with first s_1 rows in correct form. This transformation is accomplished using subroutine `Triang`, defined below by Lemma 7.3.

Lemma 7.3. *For $b > 2$, there exists an algorithm* `Triang` *that takes as*

input an $n_1 \times n_1$ upper b-banded matrix

$$B = \left[\begin{array}{ccc|cccccc} * & \cdots & * & * & \cdots & * & * & & \\ & \ddots & \vdots & \vdots & & \vdots & \vdots & \ddots & \\ & & * & * & \cdots & * & * & \cdots & * \\ \hline & & & * & \cdots & * & * & \cdots & * \\ & & & & \ddots & & & & \vdots \\ & & & & & * & & & * \\ & & & & & & * & & * \\ & & & & & & & \ddots & \vdots \\ & & & & & & & & * \end{array}\right]$$

where the principal block is $s_1 \times s_1$, and produces as output an equivalent matrix

$$B' = \left[\begin{array}{ccc|cccccc} * & \cdots & * & * & & & & & \\ & \ddots & \vdots & \vdots & \ddots & & & & \\ & & * & * & \cdots & * & & & \\ \hline & & & * & \cdots & * & * & \cdots & * \\ & & & \vdots & & & & & \vdots \\ & & & * & & & & & * \\ & & & * & & & & & * \\ & & & \vdots & & & & & \vdots \\ & & & * & \cdots & * & * & \cdots & * \end{array}\right]$$

The cost of the algorithm is $O(b^\theta)$ basic operations.

Proof. Write the input matrix as

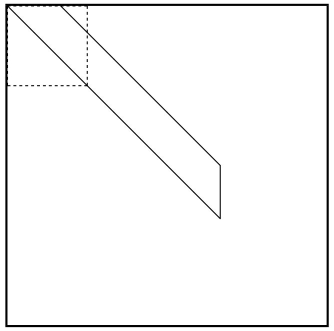
$$B = \left[\begin{array}{c|c} B_1 & B_2 \\ \hline & B_3 \end{array}\right]$$

where B_2 is $s_1 \times s_2$. Using the algorithm of Lemma 3.1, compute a principal left transform W^T which triangularizes B_2^T. Set

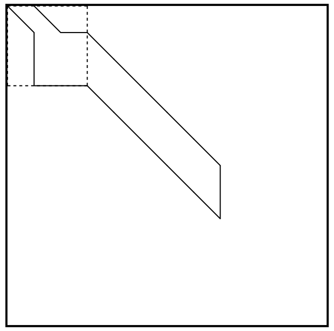
$$B' = \left[\begin{array}{c|c} B_1 & B_2 \\ \hline & B_3 \end{array}\right] \left[\begin{array}{c|c} I_{s_1} & \\ \hline & W \end{array}\right]. \tag{7.2}$$

Since $n_1 < 2b$, the cost is as stated. □

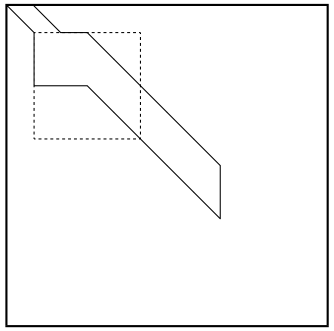
Apply subroutine `Triang` to sub$[0, n_1]$ of our initial work matrix to effect the following transformation:



$\downarrow$



At this stage we can write the work matrix as



where the focus of attention is now sub$[s_1, n_2]$. Subsequent transformations will be limited to rows $s_1+1, s_1+2, \ldots, n-t$ and columns

$s_1+s_2+1, s_1+s_2+2, \ldots, n-t$. The next step is to transform the work matrix back to an upper b-banded matrix. This is accomplished using subroutine `Shift`, defined below by Lemma 7.4.

Lemma 7.4. *For $b > 2$, there exits an algorithm* `Shift` *that takes as input an $n_2 \times n_2$ matrix*

$$C = \left[\begin{array}{ccc|ccc} * & \cdots & * & * & & \\ \vdots & & \vdots & \vdots & \ddots & \\ * & \cdots & * & * & \cdots & * \\ \hline & & & * & \cdots & * \\ & & & & \ddots & \vdots \\ & & & & & * \end{array}\right]$$

over R, where each block is $s_2 \times s_2$, and produces as output an equivalent matrix

$$C' = \left[\begin{array}{ccc|ccc} * & \cdots & * & * & & \\ & \ddots & \vdots & \vdots & \ddots & \\ & & * & * & \cdots & * \\ \hline & & & * & \cdots & * \\ & & & \vdots & & \vdots \\ & & & * & \cdots & * \end{array}\right].$$

The cost of the algorithm is $O(b^\theta)$ basic operations.

Proof. Write the input matrix as

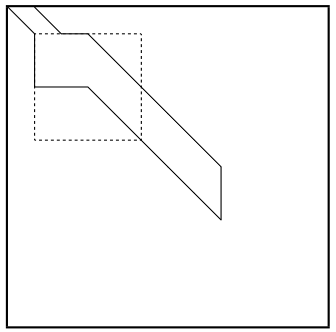
$$C = \left[\begin{array}{c|c} C_1 & C_2 \\ \hline & C_3 \end{array}\right]$$

where each block is $s_2 \times s_2$. Use the algorithm of Lemma 3.1 to compute, in succession, a principal transform U^T such that $C_1^T U^T$ is lower triangular, and then a principal transform V such that $(UC_2)V$ is lower triangular. Set

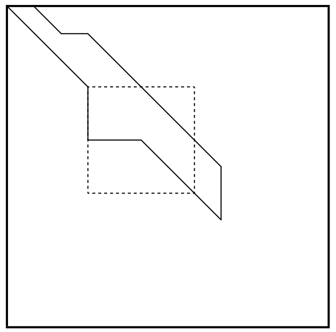
$$C' = \left[\begin{array}{c|c} U & \\ \hline & I_{s_2} \end{array}\right] \left[\begin{array}{c|c} C_1 & C_2 \\ \hline & C_3 \end{array}\right] \left[\begin{array}{c|c} I_{s_2} & \\ \hline & V \end{array}\right]. \tag{7.3}$$

Since $n_2 < 2b$, the cost is as stated. □

Apply subroutine `Shift` to sub$[s_1 + js_2, n_2]$ for $j = 0, 1, 2, \ldots, \lfloor (n - s_1)/n_2 \rfloor$ to get the following sequence of transformations.



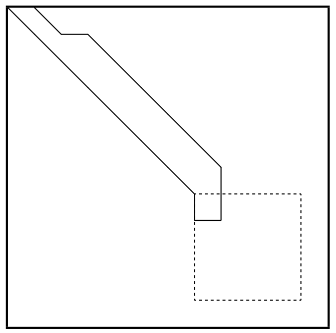
$\downarrow$



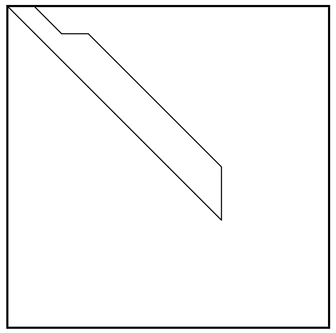
$\downarrow$

$\vdots$

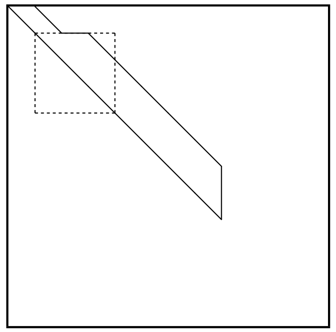
$\downarrow$



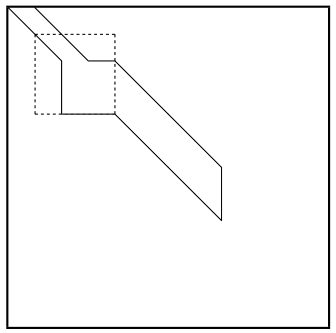
$\downarrow$



The procedure just described is now recursively applied to the trailing $(n - s_1) \times (n - s_1)$ submatrix of the work matrix, itself an upper b-banded matrix. For example, the next step is to apply subroutine `Triang` to sub$[s_1, n_1]$ to get the following transformation.



$\downarrow$



We have just shown correctness of the following algorithm supporting Proposition 7.1.

```
**Algorithm** BandReduction$(A, b)$
**Input:** An upper $b$-banded $A \in \mathsf{R}^{n\times n}$ with $b > 2$ and last $t$ columns zero.
**Output:** An upper $(\lfloor b/2\rfloor + 1)$-banded matrix that is equivalent to $A$ and also has last $t$ columns zero.
$s_1 := \lfloor b/2\rfloor$;
$n_1 := \lfloor b/2\rfloor + b - 1$;
$s_2 := b - 1$;
$n_2 := 2(b - 1)$;
$B :=$ a copy of $A$ augmented with $2b - t$ rows and columns of zeroes;
for $i = 0$ to $\lceil (n - t)/s_1\rceil - 1$ do
    Apply Triang to $\mathrm{sub}_B[is_1, n_1]$;
    for $j = 0$ to $\lceil (n - t - (i + 1)s_1)/s_2\rceil - 1$ do
        Apply Shift to $\mathrm{sub}_B[(i + 1)s_1 + js_2, n_2]$;
    od;
od;
return $\mathrm{sub}_B[0, n]$;
```

We now prove part 1 of Proposition 7.1. The number of iterations of the outer loop is

$$L_i = \lceil (n-t)/s_1\rceil < \frac{2n}{b-1}$$

while the number of iterations, for any fixed value of i, of the inner loop is

$$L_j = \lceil (n-t-(i+1)s_1)/s_2\rceil < \frac{n}{b-1}.$$

The number of applications of either subroutine `Triang` or `Shift` occurring during algorithm `BandReduction` is seen to be bounded by $L_i(1 + L_j) = O(n^2/b^2)$. By Lemmas 7.3 and 7.4 the cost of one application of either of these subroutines is bounded by $O(b^\theta)$ basic operations. The result follows.

We now prove part 2 of Proposition 7.1. Fix i and consider a single pass of the outer loop. For the single call to subroutine `Triang`, let W be as in (7.2). For each call $j = 0, \ldots, L_j - 1$ to subroutine `Shift` in the inner loop, let (U_j, V_j) be the (U, V) as in (7.3). Then the principal transform applied to $\mathrm{sub}_B(0, n)$ during this pass of the outer loop is

given by $(U^{(i)}, V^{(i)}) =$

$$\left(\begin{bmatrix} I_{(i+1)s_1} & & & & \\ & U_1 & & & \\ & & \ddots & & \\ & & & U_{L_j-1} & \\ & & & & I_{s_1}\end{bmatrix}, \begin{bmatrix} I_{s_1} & & & & \\ & W & & & \\ & & V_1 & & \\ & & & \ddots & \\ & & & & V_{L_j-1}\end{bmatrix}\right)$$

A principal transform which transforms A to an upper $(\lfloor b/2\rfloor+1)$-banded matrix is then given by

$$(U^{(L_i-1)}\cdots U^{(1)}U^{(0)},\ V^{(0)}V^{(1)}\cdots V^{(L_i-1)}). \tag{7.4}$$

Note that each $U^{(i)}$ and $V^{(i)}$ is $b - 1$ banded matrix. The product of two $b - 1$ banded matrices is $2b - 1$ banded; using an obvious block decomposition two such matrices can be multiplied in $O(nb^{\theta-1})$ ring operations. It is easy to show that the multiplications in (7.4) can be achieved in the allotted time if a binary tree paradigm is used. We don't belabor the details here. □

Corollary 7.5. *There exists an algorithm that takes as input an upper triangular $A \in \mathsf{R}^{n\times n}$, and produces as output an upper 2-banded matrix A' that is equivalent to A.*

1. *The cost of producing A' is bounded by $O(n^\theta)$ basic operations.*
2. *A principal transform (U, V) satisfying $UAV = A'$ can be recovered in $O(n^\theta(\log n))$ basic operations.*

The algorithm uses basic operations of type {Arith, Gcdex}.

Corollary 7.6. *Let $\mathsf{R} = \mathbb{Z}/(N)$. The complexity bounds of Corollary 7.5 become $O(n^\theta(\log\beta) + n^2(\log n)\,\mathrm{B}(\log\beta))$ and $O(n^\theta(\log n)(\log\beta) + n^2(\log n)\,\mathrm{B}(\log\beta))$ word operations where $\beta = nN$.*

Proof. By augmenting the input matrix with at most n rows and columns of zeroes, we may assume that $n = 2^k + 1$ for some $k \in \mathbb{N}$.

We first show part 1. Let $f_n(b)$ be a bound on the number of basic operations required to compute an upper 2-banded matrix equivalent to an $(n+1)\times(n+1)$ upper $(b+1)$-banded matrix. Obviously, $f_n(1) = 0$. From Proposition 7.1 we have that $f_n(b) \le f_n(b/2) + O(n^2b^{\theta-2})$ for b a power of two.

Part 2 follows by noting that algorithm `BandReduction` needs to be applied k times. Combining the $n \times n$ principal transforms produced by each invocation requires $O(kn^\theta)$ basic operations. □

7.2 From Diagonal to Smith Form

Proposition 7.7. *Let $D \in \mathsf{R}^{n\times n}$ be diagonal. A principal transform (U, V) such that UDV is in Smith form can be computed in $O(n^\theta)$ basic operations of type* {Arith, Gcdex}.

Corollary 7.8. *Let $\mathsf{R} = \mathbb{Z}/(N)$. The complexity bound of Proposition 7.7 becomes $O(n^\theta(\log\beta) + n^2(\log n)\,\mathsf{B}(\log\beta))$ word operations where $\beta = nN$.*

Proof. Let $f(n)$ be a bound on the number of basic operations required to compute a principal transform which satisfies the requirements of the theorem. Obviously, $f(1) = 0$. By augmenting an input matrix with at most $n-1$ rows and columns of zeroes, we may assume that n is a power of two. The result will follow if we show that $f(n) \le 2f(n/2) + O(n^\theta)$.

Let $D \in \mathsf{R}^{n\times n}$ be diagonal, $n > 1$ a power of two. Partition D as $\mathrm{diag}(D_1, D_2)$ where each of D_1 and D_2 has dimension $n/2$. Recursively compute principal transforms (U_1, V_1) and (U_2, V_2) such that

$$\left[\begin{array}{c|c} U_1 & \\ \hline & U_2 \end{array}\right]\left[\begin{array}{c|c} D_1 & \\ \hline & D_2 \end{array}\right]\left[\begin{array}{c|c} V_1 & \\ \hline & V_2 \end{array}\right] = \left[\begin{array}{c|c} A & \\ \hline & B \end{array}\right]$$

with A and B in Smith form. If either of A or B is zero, then $\mathrm{diag}(A, B)$ can be transformed to Smith form by applying a primary permutation transform.

Otherwise, it remains to compute a principal transform (U_3, V_3) such that $U_3\mathrm{diag}(A, B)V_3$ is in Smith form. The rest of this section is devoted to showing that this merge step can be performed in the allotted time. $\square$

We begin with some definitions. If A is in Smith form, we write $\mathrm{first}(A)$ and $\mathrm{last}(A)$ to denote the first and last nonzero entry in A respectively. If A is the zero matrix then $\mathrm{first}(A) = 0$ and $\mathrm{last}(A) = 1$. If B is in Smith form, we write $A < B$ to mean that $\mathrm{last}(A)$ divides $\mathrm{first}(B)$.

Definition 7.9. *Let $A, B \in \mathsf{R}^{n\times n}$ be nonzero and in Smith form. A* merge transform *for (A, B) is a principal transform (U, V) which satisfies*

$$\overset{U}{\left[\begin{array}{c|c} * & * \\ \hline * & * \end{array}\right]}\left[\begin{array}{c|c} A & \\ \hline & B \end{array}\right]\overset{V}{\left[\begin{array}{c|c} * & * \\ \hline * & * \end{array}\right]} = \overset{S}{\left[\begin{array}{c|c} A' & \\ \hline & B' \end{array}\right]}$$

where A' and B' are in Smith form and:

1. *$A' < B'$;*
2. $\mathrm{last}(A')$ *divides* $\mathrm{last}(A)$;
3. *If B has no zero diagonal entries, then* $\mathrm{last}(A')$ *divides* $\mathrm{last}(B)$;
4. *If A has no zero diagonal entries, then A' has no zero diagonal entries.*

Let $f(n)$ be the number of basic operations required to compute a principal merge transform for (A, B).

Lemma 7.10. *$f(1) = O(1)$.*

Proof. Let $a, b \in \mathsf{R}$ both be nonzero. Compute $(g, s, t, u, v) = \mathrm{Gcdex}(a, b)$ and $q = -\mathrm{Div}(tb, g)$. Then (U, V) is a merge transform for $([a], [b])$ where

$$\overset{U}{\begin{bmatrix} s & t \\ u & v \end{bmatrix}}\begin{bmatrix} a & \\ & b \end{bmatrix}\overset{V}{\begin{bmatrix} 1 & q \\ 1 & 1+q \end{bmatrix}} = \overset{S}{\begin{bmatrix} g & \\ & vb \end{bmatrix}}.$$

$\square$

Theorem 7.11. *For n a power of two, $f(n) = O(n^\theta)$.*

Proof. The result will follow from Lemma 7.10 if we show that $f(n) \le 4f(n/2) + O(n^\theta)$. Let $A, B \in \mathsf{R}^{n\times n}$ be nonzero and in Smith form, $n > 1$ a power of two. Let $t = n/2$ and partition A and B into t-dimension blocks as $A = \mathrm{diag}(A_1, A_2)$ and $B = \mathrm{diag}(B_1, B_2)$. The work matrix can be written as

$$\left[\begin{array}{c|c|c|c} A_1 & & & \\ \hline & A_2 & & \\ \hline & & B_1 & \\ \hline & & & B_2 \end{array}\right]. \tag{7.5}$$

Note that A_1 has no zero diagonal entries in case that A_2 is nonzero. Similarly, B_1 has no zero diagonal entries in case that B_2 is nonzero. We will modify the work matrix inplace by applying a finite sequence of principal transforms. To begin, the work matrix satisfies $A_1 < A_2$ and $B_1 < B_2$. The algorithm has five steps:

1. Compute a merge transform (U, V) for (A_1, B_1). By inserting t rows/columns after the tth and $2t$th row/column, we can extend

U and V to matrices in $\mathsf{R}^{4t\times 4t}$ as follows:

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

Apply the resulting principal transform to the work matrix as follows:

$$\left[\begin{array}{c|c|c|c} * & & * & \\ \hline & I & & \\ \hline * & & * & \\ \hline & & & I_t \end{array}\right] \left[\begin{array}{c|c|c|c} A_1 & & & \\ \hline & A_2 & & \\ \hline & & B_1 & \\ \hline & & & B_2 \end{array}\right] \left[\begin{array}{c|c|c|c} * & & * & \\ \hline & I & & \\ \hline * & & * & \\ \hline & & & I_t \end{array}\right]$$

The blocks labelled A_2 and B_2 stay unchanged, but the work matrix now satisfies $A_1 < B_1$. By condition 2 of Definition 7.9, we still have $A_1 < A_2$. Since B was in Smith form to start with, by condition 3 we also have $A_1 < B_2$. Then last(A_1) divides all entries in diag(A_2, B_1, B_2)). This condition on last(A_1) will remain satisfied since all further unimodular transformations will be limited to the trailing three blocks of the work matrix. Note that $B_1 < B_2$ may no longer be satisfied. Also, B_1 may now have trailing zero diagonal entries even if B_2 is nonzero.

2. If either of A_2 or B_2 is zero, skip this step. Compute a merge transform for (A_2, B_2). Similar to above, this merge transform can be expanded to obtain a principal transform which affects only blocks A_2 and B_2 of the work matrix. Apply the merge transform to diag(A_2, B_2) so that $A_2 < B_2$.

3. If either of A_2 or B_1 is zero, skip this step. Compute and apply a merge transform for (A_2, B_1) so that $A_2 < B_1$. At this point last(A_2) divides all entries in diag(B_1, B_2); this condition on last(A_2) will remain satisfied.

4. If either of B_1 or B_2 is zero, skip this step. Compute and apply a merge transform for (B_1, B_2). The work matrix now satisfies $A_1 < A_2 < B_1 < B_2$.

5. If B_2 is zero, skip this step. If B_1 has some trailing diagonal entries zero, apply a principal permutation transformation to the trailing $2t \times 2t$ submatrix of the work matrix so that this submatrix is in Smith form.

Combining the at most five principal transforms constructed above can be accompished in $O(n^\theta)$ basic operations.

It remains to show that the work matrix satisfies conditions 2, 3 and 4 of Definition 7.9. It is easy to see that condition 2 and 4 will be satisfied. Now consider condition 3. Assume that B has no nonzero diagonal entries. If A_2 was zero to start with, then condition 3 is achieved after step 1. Otherwise, condition 3 is achieved after step 2. □

7.3 From Upper 2-Banded to Smith Form

Proposition 7.12. *Let $A \in \mathsf{R}^{n\times n}$ be upper 2-banded. A transform (U, V) such that UAV is in Smith form can be computed in $O(n^\theta)$ basic operations of type* {Gcdex*,* Stab*,* Div}.

Corollary 7.13. *Let $\mathsf{R} = \mathbb{Z}/(N)$. The complexity bound of Proposition 7.12 becomes $O(n^\theta(\log\beta) + n^2(\log n)\,\mathrm{B}(\log\beta))$ word operations.*

Proof. Let $f(n)$ be a bound on the number of basic operations required to compute a transform which satisfies the requirements of the theorem. Obviously, $f(0) = f(1) = 0$. The result will follow if we show that

$$f(n) = f(\lfloor (n-1)/2 \rfloor) + f(\lceil (n-1)/2 \rceil) + O(n^\theta).$$

Let $A \in \mathsf{R}^{n\times n}$ be upper 2-banded, $n > 1$.

$$A = \begin{bmatrix} * & * & & & & \\ & * & * & & & \\ & & * & * & & \\ & & & * & & \\ & & & & \ddots & * \\ & & & & & * \end{bmatrix}$$

We will transform A to Smith form by applying (in place) a sequence of transforms to A. The algorithm has nine steps.

1. Apply a left transform as follows:

```
$n_1 := \lfloor (n-1)/2 \rfloor$;
$n_2 := \lceil (n-1)/2 \rceil$;
**for** $i$ **from** $n$ **by** $-1$ **to** $n_1 + 2$ **do**
    $(g, s, t, u, v) :=$ Gcdex$(A[i-1, i], A[i, i])$;
    $\begin{bmatrix} A[i-1,*] \\ A[i,*] \end{bmatrix} := \begin{bmatrix} s & t \\ u & v \end{bmatrix} \begin{bmatrix} A[i-1,*] \\ A[i,*] \end{bmatrix}$
```

```
od;
for $i$ from $n_1+2$ to $n-1$ do
    $(g,s,t,u,v) :=$ Gcdex$(A[i,i], A[i+1,i])$;
    $\left[\begin{array}{c} A[i,*] \\ A[i+1,*] \end{array}\right] := \left[\begin{array}{cc} s & t \\ u & v \end{array}\right]\left[\begin{array}{c} A[i,*] \\ A[i+1,*] \end{array}\right]$
od;
```

The following diagrams show A at the start, after the first loop and after completion. In each diagram, the principal block is $n_1 \times n_1$ and the trailing block is $n_2 \times n_2$. Note that $n_1 + 1 + n_2 = n$.

$$\left[\begin{array}{cccc|c|ccccc}
\ast & \ast & & & & & & & & \\
& \ast & \ast & & & & & & & \\
& & \ast & & & & & & & \\
& & \ddots & \ast & & & & & & \\
& & & \ast & \ast & & & & & \\
\hline
& & & & \ast & \ast & & & & \\
\hline
& & & & & \ast & \ast & & & \\
& & & & & & \ast & \ast & & \\
& & & & & & & \ast & & \\
& & & & & & & & \ddots & \ast \\
& & & & & & & & & \ast
\end{array}\right]$$

$$\downarrow$$

$$\left[\begin{array}{cccc|c|ccccc}
\ast & \ast & & & & & & & & \\
& \ast & \ast & & & & & & & \\
& & \ast & & & & & & & \\
& & \ddots & \ast & & & & & & \\
& & & \ast & \ast & & & & & \\
\hline
& & & & \ast & & & & & \\
\hline
& & & & \ast & \ast & & & & \\
& & & & & \ast & \ast & & & \\
& & & & & & \ast & \ast & & \\
& & & & & & & & \ddots & \\
& & & & & & & & \ast & \ast
\end{array}\right]$$

$$\downarrow$$

$$\left[\begin{array}{cccc|c|ccccc}
\ast & \ast & & & & & & & & \\
& \ast & \ast & & & & & & & \\
& & \ast & & & & & & & \\
& & \ddots & \ast & & & & & & \\
& & & \ast & \ast & & & & & \\
\hline
& & & & \ast & & & & & \\
\hline
& & & & \ast & \ast & \ast & & & \\
& & & & \ast & & \ast & \ast & & \\
& & & & \ast & & & \ast & & \\
& & & & \ast & & & & \ddots & \ast \\
& & & & \ast & & & & & \ast
\end{array}\right]$$

2. Recursively compute transforms (U_1, V_1) and (U_2, V_2) which transform the principal $n_1 \times n_1$ and trailing $n_2 \times n_2$ block of A respectively to Smith form. This costs $f(n_1) + f(n_2)$ basic operations. Apply $(\text{diag}(U_1, I_1, U_2), \text{diag}(V_1, I_1, V_2))$ to get

$$\left[\begin{array}{cccc|c|ccccc}
\ast & & & & \ast & & & & & \\
& \ast & & & \ast & & & & & \\
& & \ast & & \ast & & & & & \\
& & & \ddots & \vdots & & & & & \\
& & & \ast & \ast & & & & & \\
\hline
& & & & \ast & & & & & \\
\hline
& & & & \ast & \ast & & & & \\
& & & & \ast & & \ast & & & \\
& & & & \ast & & & \ast & & \\
& & & & \vdots & & & & \ddots & \\
& & & & \ast & & & & & \ast
\end{array}\right].$$

3. Let P be a permutation matrix which maps rows $(n_1+1, n_1+2, \ldots, n)$ to rows $(n, n_1+1, n_1+2, \ldots, n-1)$. Apply $(P, P^{(-1)})$ to

get

$$\left[\begin{array}{ccccc|ccccc|c}
* & & & & & & & & & & * \\
& * & & & & & & & & & * \\
& & * & & & & & & & & * \\
& & & \ddots & & & & & & & \vdots \\
& & & & * & & & & & & * \\ \hline
& & & & & * & & & & & * \\
& & & & & & * & & & & * \\
& & & & & & & * & & & * \\
& & & & & & & & \ddots & & \vdots \\
& & & & & & & & & * & * \\ \hline
& & & & & & & & & & *
\end{array}\right].$$

4. Compute a transform (U, V) which transforms the principal $(n-1) \times (n-1)$ submatrix of A to Smith form; by Proposition 7.7 this costs $O(n^\theta)$ basic operations. Apply $(\text{diag}(U, I_1), \text{diag}(V, I_1))$ to get

$$\left[\begin{array}{cccccc}
a_1 & & & & & * \\
& a_2 & & & & * \\
& & a_3 & & & * \\
& & & \ddots & & \vdots \\
& & & & a_{k-1} & * \\
& & & & & * \\
& & & & & * \\
& & & & & \vdots \\
& & & & & *
\end{array}\right]$$

for some $k \in \mathbb{N}$, a_{k-1} nonzero.

5. Apply a left transform as follows:

```
for $i$ from $k$ to $n-1$ do
    $(g, s, t, u, v) := \text{Gcdex}(A[k, n], A[k+1, n])$;
    $\begin{bmatrix} A[t,*] \\ A[t+1,*] \end{bmatrix} := \begin{bmatrix} s & t \\ u & v \end{bmatrix} \begin{bmatrix} A[k,*] \\ A[k+1,*] \end{bmatrix}$
od;
```

After completion we have

$$\left[\begin{array}{cccccc}
a_1 & & & & & * \\
& a_2 & & & & * \\
& & a_3 & & & * \\
& & & \ddots & & \vdots \\
& & & & a_{k-1} & * \\
& & & & & * \\
& & & & & \\
& & & & & \\
& & & & &
\end{array}\right].$$

6. Let P be the $n \times n$ permutation which switches columns $k+1$ and n. Apply P to get

$$\left[\begin{array}{cccccc|c}
a_1 & & & & & A[1,k] & \\
& a_2 & & & & A[2,k] & \\
& & a_3 & & & A[3,k] & \\
& & & \ddots & & \vdots & \\
& & & & a_{k-1} & A[k-1,k] & \\
& & & & & A[k,k] & \\ \hline
& & & & & & \\
& & & & & &
\end{array}\right]. \tag{7.6}$$

The focus of attention from now on is the principal $k \times k$ submatrix of A.

7. Apply a transform as follows:

```
for $i$ from $k-1$ by $-1$ to $1$ do
    $c := \text{Stab}(A[i,k], A[i+1,k], A[i,i])$;
    $q := -\text{Div}(cA[i+1,i+1], A[i,i])$;
    Add $c$ times row $i+1$ of $A$ to row $i$ of $A$;
    Add $q$ times column $i$ of $A$ to column $i+1$ of $A$;
od;
```

It is easy to verify that the matrix can still be written as in (7.6) after each iteration of the loop. From the definition of Stab, it follows that

$$(a_j, A[j,k]) = (a_j, A[j,k], \ldots, A[k,k]) \quad \text{for} \quad l < j < k \tag{7.7}$$

holds for $l = i - 1$ after the loop completes for a given value of i.

8. Apply a right transform as follows:

```
for $i$ to $k-1$ do
    $(s_i, s, t, u, v) := \mathrm{Gcdex}(A[i,i], A[i,k])$;
    $\begin{bmatrix} A[*,i] & A[*,k] \end{bmatrix} := \begin{bmatrix} A[*,i] & A[*,k] \end{bmatrix} \begin{bmatrix} s & u \\ t & v \end{bmatrix}$
od;
```

After completion of the loop for a given value of i, we have

$$\left[\begin{array}{cccccccc|c} s_1 &&&&&&&& \\ * & s_2 &&&&&&& \\ \vdots & \vdots & \ddots &&&&&& \\ * & * && s_i &&&&& \\ * & * & \cdots & * & a_{i+1} &&& b_{i+1} & \\ * & * && * && a_{i+2} && b_{i+2} & \\ \vdots & \vdots && \vdots &&& \ddots & \vdots & \\ * & * & \cdots & * &&&& b_k & \\ \hline &&&&&&&& \end{array}\right].$$

For convenience, set $s_0 = 1$. We now prove that the following items hold for $l = 0, 1, \ldots, k-1$.

(a) s_l divides all entries in the trailing $(n-l+1)$th submatrix of A.

(b) (7.7) holds.

Note that (a) holds trivially for $l = 0$, while (b) for $l = 0$ follows from the preconditioning performed in the previous step. By induction, assume (a) and (b) hold for $l = i$. After the loop completes with $i+1$, we have

$$\left[\begin{array}{cccccccc|c} s_1 &&&&&&&& \\ * & s_2 &&&&&&& \\ \vdots & \vdots & \ddots &&&&&& \\ * & * && s_i &&&&& \\ * & * & \cdots & * & s_{i+1} &&&& \\ * & * && * & tb_{i+2} & a_{i+2} && vb_{i+2} & \\ \vdots & \vdots && \vdots & \vdots && \ddots & \vdots & \\ * & * & \cdots & * & tb_k &&& vb_k & \\ \hline &&&&&&&& \end{array}\right].$$

where s_{i+1} is a gcd of a_{i+1} and b_{i+1}. That (a) holds for $l = i+1$ follows from the assumption that (b) holds for $l = i$. Using (b) for $l = i$ we also get

$$\begin{aligned} (a_{i+2}, vb_{i+2}) &= (a_{i+2}, va_{i+2}, vb_{i+2}) \\ &= (a_{i+2}, v(a_{i+2}, b_{i+2}, \ldots, b_k)) \\ &= (a_{i+2}, vb_{i+2}, \ldots, vb_k) \end{aligned}$$

which shows (b) for $l = i+1$.

We have just shown (by induction) that after completion we have

$$\left[\begin{array}{cccccc|c} s_1 &&&&&& \\ * & s_2 &&&&& \\ * & * & s_3 &&&& \\ & \vdots && \ddots &&& \\ * & * & * && s_{k-1} && \\ * & * & * & \cdots & * & s_k & \\ \hline &&&&&& \end{array}\right]$$

with $\mathrm{diag}(s_1, \ldots, s_k)$ in Smith form and with all off-diagonal entries in row j divisible by s_j for $1 \le j \le k-1$.

9. Let P be the $n \times n$ permutation which reverses the order of first k rows. Apply (P, P) to get

$$\left[\begin{array}{cccccc|c} s_k & * & \cdots & * & * & * & \\ & s_{k-1} && * & * & * & \\ && \ddots && \vdots && \\ &&& s_3 & * & * & \\ &&&& s_2 & * & \\ &&&&& s_1 & \\ \hline &&&&&& \end{array}\right].$$

Compute an index 1 reduction transform U for the submatrix of A comprised of rows $1, \ldots, n$ and column $2, \ldots, k$. (See Definition 3.9). By Proposition 3.10 this costs $O(n^\theta)$ basic operations. Apply

(PU, P) to get

$$\begin{bmatrix} s_1 & & & & & & \\ & s_2 & & & & & \\ & & s_3 & & & & \\ & & & \ddots & & & \\ & & & & s_{k-1} & & \\ & & & & & s_k & \\ & & & & & & \\ & & & & & & \end{bmatrix}.$$

We are finished.

By augmenting A with identity matrices as shown below, we can automatically record all transforms applied to A and recover a final transform (U, V) such that $UAV = S$.

$$\left[\begin{array}{c|c} A & I_n \\ \hline I_n & \end{array}\right] \rightarrow \left[\begin{array}{c|c} S & U \\ \hline V & \end{array}\right]$$

The cost of each step is bounded by $O(n^\theta)$ basic operations plus the two recursive calls in step 2. The result follows. □

7.4 Transformation to Smith Form

First the result for square matrices:

Lemma 7.14. *Let $A \in \mathsf{R}^{n\times n}$. The Smith form S of A can be computed in $O(n^\theta)$ basic operations. A principal transform (U, V) such that $UAV = S$ can be computed in $O(n^\theta(\log n))$ basic operations. The algorithms use basic operations of type* {Arith, Gcdex, Stab}.

Corollary 7.15. *Let $\mathsf{R} = \mathbb{Z}/(N)$. The complexity bounds of Lemma 7.14 become $O(n^\theta(\log\beta) + n^2(\log n)\,\mathrm{B}(\log\beta)))$ and*
$O(n^\theta(\log n)(\log\beta) + n^2(\log n)\,\mathrm{B}(\log\beta))$ bit operations respectively where $\beta = mN$.

Proof. Compute an echelon form T of A using Lemma 3.1. Now apply in succesion the algorithms of Propositions 7.1 and 7.12 to transform the principal $m\times m$ block of T to Smith form. A principal transform can be obtained by multiplying together the transforms produced by Lemma 3.1 and Propositions 7.1 and 7.12. □

For rectangular matrices:

Proposition 7.16. *Let $A \in \mathsf{R}^{n\times m}$. The Smith form S of A can be computed in $O(nmr^{\theta-2}(\log r))$ basic operations. A principal transform (U, V) such that $UAV = S$ can be computed in $O(nmr^{\theta-1}(\log n + m))$ basic operations. The algorithms use basic operations of type* {Arith, Gcdex, Stab}.

Corollary 7.17. *Let $\mathsf{R} = \mathbb{Z}/(N)$. The complexity bounds of Proposition 7.16 become $O(nmr^{\theta-2}(\log r)(\log\beta) + nm(\log r)\,\mathrm{B}(\log\beta)))$ and $O(nmr^{\theta-2}(\log n)(\log\beta) + nm(\log n)(\log r)\,\mathrm{B}(\log\beta))$ word operations where $\beta = mN$.*

Proof. If no transform is desired, compute a minimal echelon form B of A^T using Proposition 3.5b. Similarly, compute a minimal echelon form T of B^T. Then T has all entries outside the principal $r\times r$ submatrix zero. Now use Lemma 7.14. If a transform is desired, use instead the algorithm of Proposition 3.7 to produce B. A transform can be produced by multiplying together the transforms produced by Propositions 3.7 and Lemma 7.14. □

Modular Computation of the Smith Form

Let $N \in \mathcal{A}(\mathsf{R})$. Recall that we use ϕ to denotes the canonical homomorphism from from R to $\mathsf{R}/(N)$.

The following lemma follows from the canonicity of the Smith form over R and $\mathsf{R}/(N)$. Note that if (U, V) is a transform for A, then $(\phi(U), \phi(V))$ is a transform for $\phi(A)$.

Lemma 7.18. *Let $A \in \mathsf{R}^{n\times m}$ have Smith form S with nonzero diagonal entries $s_1, s_2, \ldots, s_r \in \mathcal{A}(\mathsf{R})$. Let $N \in \mathcal{A}(\mathsf{R})$ be such that s_r divides N but N does not divide s_r. If the definition of $\mathcal{A}$ over $\mathsf{R}/(N)$ is consistent, then $\phi(S)$ is the Smith form of $\phi(A)$.*

Corollary 7.19. *$\phi^{-1}(\phi(S))$ is the Smith form of A over R.*

Now consider the case $\mathsf{R} = \mathbb{Z}$. A suitable N in the sense of Lemma 7.18 can be reovered by computing a fraction free Gauss transform A.

Proposition 7.20. *The Smith form of an $A \in \mathbb{Z}^{n\times m}$ can be recovered in $O(nmr^{\theta-2}(\log\beta) + nm(\log r)\,\mathrm{B}(\log\beta))$ bit operations where r is the rank of A and $\beta = (\sqrt{r}||A||)^r$.*