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

2000

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

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.

$$H = UA = \left[\begin{array}{ccccccccc} h_1 & * & * & \bar{*} & \bar{*} & * & * & * & * \\ & & & h_2 & \bar{*} & * & * & * & * \\ & & & & h_3 & * & * & * & * \\ & & & & & & & & \\ & & & & & & & & \\ & & & & & & & & \end{array}\right].$$

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

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

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:

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

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 (1998).

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 $\rightarrow$ 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

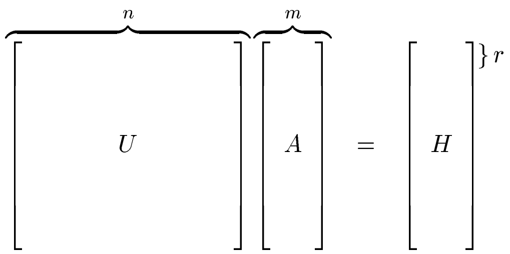


Figure 1.1: Tri-parameter analysis

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. $\square$

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 pre-image 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 $|a|, |b| \le 2^k$, 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 (1999).

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* $a \le b$

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 $l \ge 6 + \log\log\beta$. 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 $\mathrm{MM}(n) \le 2n^3$. Better asymptotic bounds are available, see the notes below. Using an obvious block decomposition we get:

Fact 1.6. *We have*

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

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 $2 < \theta \le 3$. 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 $||AB|| \le b \cdot ||A|| \cdot ||B||$. 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 $0 < r \le 1$. This bound for rectangular matrix multiplication can be substantially improved. For example, Coppersmith (1996) shows that $\mathrm{MM}(n,n,n^r) = O(n^{2+\epsilon})$ for any $\epsilon > 0$ if $r \le 0.294$, $n \to \infty$. 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 $2 < \theta \le 3$.

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*

$$f_\gamma(m,r) = \left[\begin{array}{l} \textbf{if } m = 1 \textbf{ or } r = 0 \textbf{ then return } \gamma cm \\ \textbf{else} \\ \quad \text{Choose nonngative } r_1 \text{ and } r_2 \text{ which satisfy } r_1 + r_2 = r; \\ \quad \textbf{return } f_\gamma(\lfloor m/2 \rfloor, r_1) + f_\gamma(\lceil m/2 \rceil, r_2) + \gamma cm \\ \textbf{fi} \end{array}\right.$$

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). $\square$

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*

$$f_\gamma(m,r) = \left[\begin{array}{l} \textbf{if } m = 1 \textbf{ or } r = 0 \textbf{ then return } \gamma cm \\ \textbf{else} \\ \quad \text{Choose nonngative } r_1 \text{ and } r_2 \text{ which satisfy } r_1 + r_2 = r; \\ \quad \textbf{return } f_\gamma(\lfloor m/2 \rfloor, r_1) + f_\gamma(\lceil m/2 \rceil, r_2) + \gamma cmr^{\theta-2} \\ \textbf{fi} \end{array}\right.$$

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 $f_\gamma(m,r) \leq 3c/(1 - 2^{2-\theta})\gamma m r^{\theta-2}$. $\square$

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.

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

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

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

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

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

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

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

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

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

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

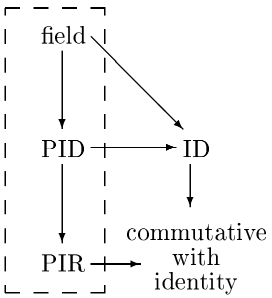


Figure 1.3: Relationship between rings

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 2

Echelon Forms over Fields

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.

Let K be a field. Every $A \in \mathsf{K}^{n\times m}$ can be transformed using elementary row operations to an $R \in \mathsf{K}^{n\times m}$ which satisfies the following:

(r1) Let r be the number of nonzero rows of R. Then the first r rows of R are nonzero. For $1 \leq i \leq r$ let $R[i, j_i]$ be the first nonzero entry in row i. Then $j_1 < j_2 < \cdots < j_r$.

(r2) $R[i, j_i] = 1$ and $R[k, j_i] = 0$ for $1 \leq k < i \leq r$.

R is the unique reduced row echelon form of A, also called the GaussJordan canonical form of A. The sequence $(j_1, \ldots, j_r)$ is the *rank profile* of A — the lexicographically smallest subsequence of $(1, \ldots, m)$ such that columns $j_1, \ldots, j_r$ of A have rank r. The following example has rank profile $(1, 4, 5)$.

$$R = \left[\begin{array}{ccccccccc} 1 & * & * & & & * & * & * & * \\ & & & 1 & & * & * & * & * \\ & & & & 1 & * & * & * & * \\ \hline & & & & & & & & \\ & & & & & & & & \\ & & & & & & & & \end{array}\right]$$

It is a classical result that entries in R can be expressed as quotients of minors of the input matrix A. Let $P \in \mathsf{K}^{n\times n}$ be a permutation matrix such that the first r rows of PA are linearly independent. Let $A_1 \in \mathsf{K}^{r\times r}$ and $A_2 \in \mathsf{K}^{(n-r)\times r}$ be the submatrix of PA comprised of columns $j_1, \ldots, j_r$ and first r and last $n-r$ rows respectively. Now let $U_1 = A_1^{\mathsf{adj}}$ and $U_2 = -A_2 A_1^{\mathsf{adj}}$. Then

$$\overset{U}{\left[\begin{array}{c|c} U_1 & \\ \hline U_2 & dI_{n-r} \end{array}\right]} \overset{PA}{\left[\begin{array}{ccccccccc} * & * & * & * & * & * & * & * & * \\ * & * & * & * & * & * & * & * & * \\ * & * & * & * & * & * & * & * & * \\ \hline * & * & * & * & * & * & * & * & * \\ * & * & * & * & * & * & * & * & * \\ * & * & * & * & * & * & * & * & * \end{array}\right]} = \overset{dR}{\left[\begin{array}{ccccccccc} d & * & * & & & * & * & * & * \\ & & & d & & * & * & * & * \\ & & & & d & * & * & * & * \\ \hline & & & & & & & & \\ & & & & & & & & \\ & & & & & & & & \end{array}\right]} \qquad (2.1)$$

where d is the determinant of A_1. The key point is that all entries in U and dR are minors of A bounded in dimension by r. We call (U, P, r, d) a fraction free GaussJordan transform of A. The transform exists over an integral domain R. In Section 2.2 we give an algorithm for fraction free GaussJordan transform that requires $O(nmr^{\theta-2})$ multiplications and subtractions plus $O(nm \log r)$ exact divisions from R. When $\mathsf{R} = \mathbb{Z}$ the cost of the algorithm is $O(nmr^{\theta-2}(\log \beta) + nm(\log r)\,\mathrm{B}(\log \beta))$ word operations where $\beta = (\sqrt{r}||A||)^r$.

In Section 2.3 we modify the algorithm to compute a fraction free Gauss transform of A. This is a tuple (U, P, r, d) where P, r and d are as before but the principal $r \times r$ submatrix of U is lower triangular. For example

$$\overset{U}{\left[\begin{array}{ccc|c} 1 & & & \\ * & d_1 & & \\ * & * & d_2 & \\ \hline & U_2 & & dI_{n-r} \end{array}\right]} \overset{PA}{\left[\begin{array}{ccccccccc} * & * & * & * & * & * & * & * & * \\ * & * & * & * & * & * & * & * & * \\ * & * & * & * & * & * & * & * & * \\ \hline * & * & * & * & * & * & * & * & * \\ * & * & * & * & * & * & * & * & * \\ * & * & * & * & * & * & * & * & * \end{array}\right]} = \left[\begin{array}{ccccccccc} d_1 & * & * & * & * & * & * & * & * \\ & & & d_2 & * & * & * & * & * \\ & & & & d_3 & * & * & * & * \\ \hline & & & & & & & & \\ & & & & & & & & \\ & & & & & & & & \end{array}\right] \qquad (2.2)$$

where d_i is the determinant of the ith principal submatrix ot A_1 (whereby $d = d_3$). This is the form obtained using fraction free Gaussian elimination. The last row of the ith principal submatrix of U is the last row in the adjoint of the ith principal submatrix of A_1. The principal $r \times r$ submatrix of U is called the triangularizing adjoint of A_1 (see Section 2.5).

Section 2.5 an algorithm is given for computing a modified fraction free Gauss transform of A. The only difference from the Gauss transform is that the role of P is played by a unit upper triangular "conditioning" matrix C. This form gives a simple and efficient algorithm for computing a solution to a system of linear diophantine equations over $\mathbb{Q}[x]$ (see Section 5.3).

Finally, Section 2.5 gives an algorithm for computing the triangularizing adjoint of a square matrix in the general case, when the leading minors might not all be nonzero.

Notes

The classical algorithm — GaussJordan elimination — requires $O(nmr)$ field operations. By recording row operations during the reduction, transformation matrices U and P can be recovered in the same time. Fraction free versions of Gaussian elimination are given by Bareiss (1968, 1972) and Edmonds (1967) (see also (Geddes *et al.*, 1992)). The algorithms we present here are nothing more than recursive versions of fraction free Gaussian elimination. The essential idea is straightforward; our purpose here is to provide simple-to-state (and implement) algorithms which handle uniformly an input matrix of arbitrary shape and rank profile. The algorithms in this chapter will be called upon many times in the rest of this thesis.

A $O(nm^{\theta-2})$ field operations algorithm (where $m \geq n$) for the echelon form has been given already by Keller-Gehrig (1985). Better known is the LUP-decomposition algorithm of Bunch and Hopcroft (1974) (see also the texts by Aho *et al.* (1974) and Cormen *et al.* (1989)) and the more general LSP-decomposition of Ibarra *et al.* (1982). The LSP decomposition is well suited to solving many liner algebra problems (system solving, determinant and inverse) but has the drawback that the rank profile of A is not recovered, nor is S a canonical form. Many asymptotically fast algorithms for linear algebra problems over fields have been given previously. For a survey we refer to the text by Bürgisser *et al.* (1996), chapter 16.

2.1 Preliminaries

In this section we recall fraction free Gaussian elimination and give an example motivating the main idea of the asymptotically fast algorithms to follow.

The following simplified version of fraction free Gauss Jordan elimination assumes that all leading minors of A are nonsingular. In this case no permutations of rows will be necessary to ensure that pivots are nonzero. The reader is encouraged to examine the algorithm and subsequent lemma together with the worked example below.

$B :=$ a copy of A;
$e_0 := 1$;
for k **to** m **do**
  $e_k := B[k,k]$
  $E_k := e_k I_n$; $E_k[*,k] := -B[*,k]$; $E_k[k,k] := e_{k-1}$;
  $B := \frac{1}{e_{k-1}} E_k B$;
od;

Lemma 2.1. *Let $A \in \mathsf{R}^{n\times m}$ have all leading minors nonsingular. Let e_k and E_k be as produced by the code given above with input A, $1 \le k \le m$, $e_0 = 1$. Then e_k is the kth principal minor of A and*

$$\frac{1}{e_{k-1}}E_k \cdots \frac{1}{e_1}E_2\frac{1}{e_0}E_1 \tag{2.3}$$

is the first component of a fraction free GaussJordan transform of the submatrix comprised of the first k columns of A. Moreover, for any $1 \le i \le k$, the matrix

$$\frac{1}{e_{k-1}}E_k \cdots \frac{1}{e_i}E_{i+1}E_i$$

has columns $1,2,\ldots,i-1,k+1,k+2,\ldots,m$ equal to the same columns in e_kI_n and columns $i,i+1,\ldots,k$ equal to the same columns of the matrix in (2.3).

Consider the input matrix

$$A = \begin{bmatrix} 4 & 5 & 0 & 3 \\ 1 & 3 & 3 & 2 \\ 2 & 4 & 3 & 4 \\ 0 & 2 & 3 & 3 \\ 5 & 4 & 0 & 0 \end{bmatrix}.$$

The leading minors of A are $(e_0,e_1,e_2,e_3,e_4) = (1,4,7,3,9)$. We get

$$\overset{E_1}{\begin{bmatrix} 1 & & & & \\ -1 & 4 & & & \\ -2 & & 4 & & \\ 0 & & & 4 & \\ -5 & & & & 4 \end{bmatrix}} \begin{bmatrix} 4 & 5 & 0 & 3 \\ 1 & 3 & 3 & 2 \\ 2 & 4 & 3 & 4 \\ 0 & 2 & 3 & 3 \\ 5 & 4 & 0 & 0 \end{bmatrix} = \begin{bmatrix} 4 & 5 & 0 & 3 \\ & 7 & 12 & 5 \\ & 6 & 12 & 10 \\ & 8 & 12 & 12 \\ & -9 & 0 & -15 \end{bmatrix}$$

$$\frac{1}{4}\overset{E_2}{\begin{bmatrix} 7 & -5 & & & \\ & 4 & & & \\ & -6 & 7 & & \\ & -8 & & 7 & \\ & 9 & & & 7 \end{bmatrix}} \begin{bmatrix} 4 & 5 & 0 & 3 \\ & 7 & 12 & 5 \\ & 6 & 12 & 10 \\ & 8 & 12 & 12 \\ & -9 & 0 & -15 \end{bmatrix} = \begin{bmatrix} 7 & & -15 & -1 \\ & 7 & 12 & 5 \\ & & 3 & 10 \\ & & -3 & 11 \\ & & 27 & -15 \end{bmatrix}$$

$$\frac{1}{7}\overset{E_3}{\begin{bmatrix} 3 & & 15 & & \\ & 3 & -12 & & \\ & & 7 & & \\ & & 3 & 3 & \\ & & -27 & & 3 \end{bmatrix}} \begin{bmatrix} 7 & & -15 & -1 \\ & 7 & 12 & 5 \\ & & 3 & 10 \\ & & -3 & 11 \\ & & 27 & -15 \end{bmatrix} = \begin{bmatrix} 3 & & & 21 \\ & 3 & & -15 \\ & & 3 & 10 \\ & & & 9 \\ & & & -45 \end{bmatrix}$$

$$\frac{1}{3}\overbrace{\begin{bmatrix} 9 & & & -21 & \\ & 9 & & 15 & \\ & & 9 & -10 & \\ & & & 3 & \\ & & & 45 & 9 \end{bmatrix}}^{E_4}\begin{bmatrix} 3 & & & 21 & \\ & 3 & & -15 & \\ & & 3 & 10 & \\ & & & 9 & \\ & & & -45 & \end{bmatrix} = \begin{bmatrix} 9 & & & & \\ & 9 & & & \\ & & 9 & & \\ & & & 9 & \\ & & & & \end{bmatrix}$$

Combining the E_is gives the fraction free Gauss transform for A.

$$\frac{1}{e_3}E_4(\frac{1}{e_2}E_3(\frac{1}{e_1}E_2(\frac{1}{e_0}E_1))) = \begin{bmatrix} -9 & -3 & 24 & -21 & \\ 9 & 6 & -21 & 15 & \\ -6 & 2 & 11 & -10 & \\ 0 & -6 & 3 & 3 & \\ 9 & -9 & -36 & 45 & 9 \end{bmatrix}.$$

The recursive algorithm of the next section works by dividing the input matrix into slices of contiguous columns. These slices are recursively divided until the base case (one column) is reached. Each base case is precisely the computation of one of the E_i's and e_i's as in the algorithm above. As an example we give the top level of the recursion. The input matrix A is divided into two slices: the first two columns A_1 and second two columns B.

First the fraction free GuassJordan transform $\frac{1}{e_1}E_2E_1$ is computed for A_1. Then we premultiply B by this transform to get our second subproblem A_2.

$$\overbrace{\begin{bmatrix} 3 & -5 & & & \\ -1 & 4 & & & \\ -2 & -6 & 7 & & \\ 2 & -8 & & 7 & \\ -11 & 9 & & & 7 \end{bmatrix}}^{\frac{1}{e_1}E_2E_1}\left[\begin{array}{cc|cc} \multicolumn{2}{c}{A_1} & \multicolumn{2}{c}{B} \\ 4 & 5 & 0 & 3 \\ 1 & 3 & 3 & 2 \\ 2 & 4 & 3 & 4 \\ 0 & 2 & 3 & 3 \\ 5 & 4 & 0 & 0 \end{array}\right] = \left[\begin{array}{cc|cc} & & \multicolumn{2}{c}{A_2} \\ 7 & & -15 & -1 \\ & 7 & 12 & 5 \\ & & 3 & 10 \\ & & -3 & 11 \\ & & 27 & -15 \end{array}\right]$$

The second subproblem starts with A_2 and e_2 and continues Gauss Jordan elimination to get $\frac{1}{e_3}E_4E_3$.

$$\begin{bmatrix} 9 & & 24 & -21 & \\ & 9 & -21 & 15 & \\ & & 11 & -10 & \\ & & 3 & 3 & \\ & & -36 & 45 & 9 \end{bmatrix}\begin{bmatrix} 7 & & -15 & -1 & \\ & 7 & 12 & 5 & \\ & & 3 & 10 & \\ & & -3 & 11 & \\ & & 27 & -15 & \end{bmatrix} = \begin{bmatrix} 9 & & & & \\ & 9 & & & \\ & & 9 & & \\ & & & 9 & \\ & & & & \end{bmatrix}$$

The combining phase produces the complete GaussJordan transform by multiplying the transforms of the two subproblems together.

$$\underbrace{\frac{1}{e_2}(\overbrace{\frac{1}{e_3}E_4E_3}) (\overbrace{\frac{1}{e_1}E_2E_1})} = \begin{bmatrix} -9 & -3 & 24 & -21 & \\ 9 & 6 & -21 & 15 & \\ -6 & 2 & 11 & -10 & \\ 0 & -6 & 3 & 3 & \\ 9 & -9 & -36 & 45 & 9 \end{bmatrix}$$

Lemma 2.1 assures us that all entries in each of the matrices corresponding to a bracketed subexpression will be (up to sign) minors of A bounded in dimension by m. As a further example, we could also combine the E_i's as follows:

$$\frac{1}{e_3}E_4(\underbrace{\frac{1}{e_1}(\overbrace{\frac{1}{e_2}E_3E_2})E_1})$$

The algorithms of the next sections are based on matrix multiplication with structured matrices (as in the worked example above.) In the rest of this section we study these matrix multiplications with respect to cost and structure preservation. Above we assumed that all leading minors of A were nonsingular. When this is not the case we will need to introduce a permutation P or unit upper triangular conditioning matrix C.

Correctness of the lemmas is easily verified. The cost estimates follow from Fact 1.6.

Lemma 2.2. *Let $P_2, U_1 \in \mathsf{K}^{n\times n}$ with P_2 nonsingular have the shape*

$$P_2 = \left[\begin{array}{c|c|c} I & & \\ \hline & I & \\ \hline & & * \end{array}\right] \quad \text{and} \quad U_1 = \left[\begin{array}{c|c|c} d_1 I & * & \\ \hline & * & \\ \hline & * & d_1 I \end{array}\right]$$

where the block decomposition is conformal. Then $P_2U_1P_2^{-1}$ has the same shape as U_1 and $P_2U_1P_2^{-1} = P_2(U_1 - d_1I) + d_1I$.

Lemma 2.3. *Let $C_1, C_2 \in \mathsf{K}^{n\times n}$ be unit upper triangular with the shape shown. Then*

$$\overbrace{\left[\begin{array}{c|c|c} I_{r_1} & & \\ \hline & c_{22} & c_{23} \\ \hline & & I \end{array}\right]}^{C_2}\overbrace{\left[\begin{array}{c|c|c} c_{11} & c_{12} & c_{13} \\ \hline & I_{r_2} & \\ \hline & & I \end{array}\right]}^{C_1} = \overbrace{\left[\begin{array}{c|c|c} c_{11} & c_{12} & c_{13} \\ \hline & c_{22} & c_{23} \\ \hline & & I \end{array}\right]}^{C_2+C_1-I}$$

where the block decomposition is conformal.

Lemma 2.4. *Let $C_1 \in \mathsf{K}^{n\times n}$ be as in Lemma 2.3 and $B \in \mathsf{K}^{n\times m_2}$. Then the product $C_1 B$ can be computed in $O(nm_2^{\theta-1})$ basic operations of type* Arith.

The next lemma is used to construct the second subproblem.

Lemma 2.5. *Let $d_0, d_1 \in \mathsf{K}$ be nonzero. If $P_1 B \in \mathsf{K}^{n\times m_2}$ and*

$$U_1 = \left[\begin{array}{c|c|c} d_1 I_k & a & \\ \hline & u & \\ \hline & c & d_1 I_{n-k-r_1} \end{array}\right] \quad \textit{and} \quad P_1 B = \left[\begin{array}{c} e \\ \hline f \\ \hline g \end{array}\right]$$

where e and f have row dimension k and r_1 respectively, then $\frac{1}{d_0}U_1 P_1 B = A_2$ where

$$A_2 = \left[\begin{array}{c} e_2 \\ \hline f_2 \\ \hline g_2 \end{array}\right] \quad \textit{with} \quad \begin{array}{rcl} e_2 & = & \frac{1}{d_0}(d_1 e + af) \\ f_2 & = & \frac{1}{d_0} uf \\ g_2 & = & \frac{1}{d_0}(d_1 g + cf) \end{array}$$

The computation of A_2 requires at most $O(n \max(r_1, m_2) \min(r_1, m_2)^{\theta-2})$ basic operations of type Arith *plus at most $O(n(r_1+m_2))$ basic operations of type* Div.

The next lemma is used to combine the results of the two subproblems.

Lemma 2.6. *Let $P_1, P_2 \in \mathsf{K}^{n\times n}$ with P_2 nonsingular and let $d_1, d \in \mathsf{K}$ be nonzero. If*

$$U_2 = \left[\begin{array}{c|c|c|c} dI_k & & a_2 & \\ \hline & dI_{r_1} & \bar{a}_2 & \\ \hline & & u_2 & \\ \hline & & c_2 & dI \end{array}\right] \quad \textit{and} \quad P_2 U_1 P_2^{-1} = \left[\begin{array}{c|c|c|c} d_1 I_k & a_1 & & \\ \hline & u_1 & & \\ \hline & c_1 & d_1 I_{r_2} & \\ \hline & \bar{c}_1 & & d_1 I \end{array}\right]$$

then $\frac{1}{d_1}U_2 P_2 U_1 P_1 = UP$ where $P = P_2 P_1$ and

$$U = \left[\begin{array}{c|c|c|c} dI_k & a_{11} & a_2 & \\ \hline & u_{11} & \bar{a}_2 & \\ \hline & u_{21} & u_2 & \\ \hline & c_{11} & c_2 & dI \end{array}\right] \quad \textit{with} \quad \begin{array}{rcl} a_{11} & = & \frac{1}{d_1}(da_1 + a_2 c_1) \\ u_{11} & = & \frac{1}{d_1}(du_1 + \bar{a}_2 c_1) \\ u_{21} & = & \frac{1}{d_1} u_2 c_1 \\ c_{11} & = & \frac{1}{d_1}(dc_1 + c_2 c_1) \end{array}$$

The computation of U and P requires at most $O(n \max(r_1, r_2) \min(r_1, r_2)^{\theta-2})$ basic operations of type Arith *plus at most $O(n(r_1+r_2))$ basic operations of type* Div.

2.2 The GaussJordan Transform

For now all matrices are over a field K. For convenience we extend the notion of determinant to handle also rectangular matrices as follows: if $B \in \mathsf{K}^{n\times m}$ has rank less than n, then $\det B = 0$, otherwise, if $B \in \mathsf{K}^{n\times m}$ has rank profile $(j_1, \ldots, j_r)$ with $r = n$, then $\det B$ is the determinant of the submatrix of B comprised of columns $j_1, \ldots, j_r$ and first r rows. If B has zero rows or columns then $\det B = 1$.

We continue by defining a generalization of the fraction free GaussJordan transform.

Definition 2.7. *Let $A \in \mathsf{K}^{n\times m}$, $d_0 \in \mathsf{K}$ be nonzero, and k be such that $0 \le k \le n$. A* fraction free index GaussJordan transform *of (A, k, d_0) is a 5-tuple (U, P, r, h, d) with $U \in \mathsf{K}^{n\times n}$ nonsingular and $P \in \mathsf{K}^{n\times n}$ a permutation which satisfy and can be written as*

$$\begin{array}{ccccc} \frac{1}{d}U & & \frac{1}{d_0}PA & & R \\ \left[\begin{array}{c|c|c} I_k & * & \\ \hline & * & \\ \hline & * & I_{n-k-r} \end{array}\right] & & \left[\begin{array}{c} * \\ \hline * \\ \hline * \end{array}\right] & = & \left[\begin{array}{c} * \\ \hline * \\ \hline \end{array}\right] \end{array} \quad \textit{with} \quad P = \left[\begin{array}{c|c|c} I_k & & \\ \hline & * & \\ \hline & & I_h \end{array}\right] \tag{2.4}$$

and where the block decomposition for $\frac{1}{d}U$, $\frac{1}{d_0}PA$ and R is conformal and:

- *r is the rank of rows $k+1, k+2, \ldots, n$ of A;*
- *h is maximal such that rows $k+1, k+2, \ldots, n-h$ of A have rank r;*
- *d is equal to d_0 times the determinant of the middle block of $\frac{1}{d_0}PA$;*
- *The middle blocks of $\frac{1}{d_0}PA$ and R have full row rank r. The middle block of R is in reduced row echelon form. Entries in columns $j_1, \ldots, j_r$ of the upper block of R are zero, where $(j_1, \ldots, j_r)$ is the rank profile of the middle block of R.*

We call R the index k Gauss Jordan form of $\frac{1}{d_0}A$. For brevity, we will say "index transform" to mean "fraction free index GaussJordan transform".

Theorem 2.9. *Algorithm 2.8 is correct.*

Algorithm 2.8. `GaussJordan`(A, k, d_0)
Input: (A, k, d_0) with $A \in \mathsf{K}^{n\times m}$, $0 \le k \le n$ and $d_0 \in \mathsf{K}$ nonzero.
Output: (U, P, r, h, d), a fraction free index Gauss transform.
if $(A[i,*] = 0$ for $k < i \le n)$ **then**
  $(U, P, r, h, d) := (d_0 I, I, 0, n-k, d_0)$
else if $m = 1$ **then**
  $i :=$ minimal index with $i > k$ and $A[i, 1] \ne 0$;
  $P :=$ the permutation matrix which interchanges rows k and i;
  $(r, h, d) := (1, n-i, (PA)[k, 1])$;
  $U := dI_n$; $U[*, k] := -(PA)[*, 1]$; $U[k, k] := d_0$;
else
  Choose positive m_1 and m_2 with $m_1 + m_2 = m$;
  $A_1 :=$ the first m_1 columns of A;
  $B :=$ the last m_2 columns of A;
  $(U_1, P_1, r_1, h_1, d_1) :=$ `GaussJordan`(A_1, k, d_0);
  $A_2 := \frac{1}{d_0} U_1 P_1 B$;
  $(U_2, P_2, r_2, h_2, d) :=$ `GaussJordan`$(A_2, k + r_1, d_1)$;
  $(U, P, r, h, d) := (\frac{1}{d_1} U_2(P_2(U_1 - d_1 I) + I), P_2 P_1, r_1 + r_2, \min(h_1, h_2), d)$;
fi;
return (U, P, r, h, d);

Proof. We need to show that the tuple (U, P, r, h, d) returned by the algorithm satisfies Definition 2.7. Use induction on (m, r). It is easy to verify by comparison with this definition that the two base cases are correct (when $r = 0$ and/or $m = 1$).

Now assume $m > 1$ and $r > 0$ and choose positive m_1 and m_2 with $m_1 + m_2 = m$. Let B, $(U_1, P_1, r_1, h_1, d_1)$, (U_2, P_2, r_2, h_2, d) and (U, P, r, h, d) be as computed in the algorithm. By induction $(U_1, P_1, r_1, h_1, d_1)$ and $(U_2, P_2, r_2, h_1, d_1)$ are computed correctly. Then

$$\frac{1}{d_1} U_1 P_1 \begin{array}{c} \begin{array}{cc} \frac{1}{d_0}A_1 & \frac{1}{d_0}B \end{array} \\ \left[\begin{array}{c|c} * & * \\ \hline * & * \\ \hline * & * \end{array}\right] \end{array} = \begin{array}{c} \begin{array}{cc} R_1 & \frac{1}{d_1}A_2 \end{array} \\ \left[\begin{array}{c|c} * & * \\ \hline * & * \\ \hline & * \end{array}\right] \end{array}$$

where R_1 is the index k Gauss Jordan form of $\frac{1}{d_0}A_1$ and the matrices written in block decomposition have upper block with k rows and center block with r_1 rows.

Because of the structure of U_2, P_2 and R_1, we have $U_2 P_2 R_1 = R_1$.

Then

$$\frac{1}{d} U_2 P_2 \frac{1}{d_1} U_1 P_1 \frac{1}{d_0} A = \frac{1}{d} U_2 P_2 \begin{array}{c} \begin{array}{cc} R_1 & \frac{1}{d_1}A_2 \end{array} \\ \left[\begin{array}{c|c} * & * \\ \hline * & * \\ \hline & * \end{array}\right] \end{array} = \begin{array}{c} \begin{array}{cc} R_1 & R_2 \end{array} \\ \left[\begin{array}{c|c} * & * \\ \hline * & * \\ \hline & * \end{array}\right] \end{array}$$

where R_2 is the index $k + r_1$ echelon form of $\frac{1}{d_1}A_2$. By Lemma 2.2 we have $\frac{1}{d}UP = \frac{1}{d}U_2 P_2 \frac{1}{d_1} U_1 P_1$ where U and P have the shape required by Definition 2.7. It is easy to see that h and r are correct. Correctness of d is also not difficult to show. □

Theorem 2.10. *Assume the choices $m_1 = \lfloor m/2 \rfloor$ and $m_2 = \lceil m/2 \rceil$. Then the cost of Algorithm 2.8 is $O(nmr^{\theta-2})$ basic operations of type* Arith *plus $O(nm \log r)$ basic operations of type* Div.

Proof. Let $f_{n,k}(m, r)$ to be the number of basic operations of type Arith required. Recall that "comparison with zero" is counted as a basic operation of type Arith. The two base cases of the algorithm yields $f_{n,k}(m, 0) = O(nm)$ and $f_{n,k}(1, 1) = O(n)$.

Similarly, let $g_{n,k}(m, r)$ be the number of basic operations of type Div required. Then $g_{n,k}(m, 0) = 0$ and $g_{n,k}(1, 1) = 0$.

Now assume $m > 1$ and $r > 0$. By Lemmas 2.5 and 2.6 we get $f_{n,k}(m, r) \le f_{n,k}(m_1, r_1) + f_{n,k+r_1}(m_2, r_2) + O(nmr^{\theta-2})$ and $g_{n,k}(m, r) \le g_{n,k}(m_1, r_1) + g_{n,k+r_1}(m_2, r_2) + O(nm)$ for some nonnegative r_1 and r_2 with $r_1 + r_2 = r$.

The result now follows from Section 1.3. □

Note that if (U, P, r, h, d) is an index transform for $(A, 0, 1)$ then (U, P, r, d) is a fraction free GaussJordan transform for A. As discussed in the previous section, on input with $(A, 0, 1)$ all intermediate entries in the algorithm will be (up to sign) minors of A bounded in dimension by r. This gives:

Proposition 2.11. *Let $A \in \mathsf{R}^{n\times m}$, R an integral domain. A fraction free GaussJordan transform (U, P) for A can be recovered in $O(nmr^{\theta-2})$ basic operations of type* Arith *plus $O(nm \log r)$ basic operations of type* Div.

When $\mathsf{R} = \mathbb{Z}$, Hadamard's inequality bounds the magnitude of every $r \times r$ minor of A by $\beta = (\sqrt{r}||A||)^r$.

Corollary 2.12. *Let $\mathsf{R} = \mathbb{Z}$. The complexity bound of Proposition 2.11 becomes $O(nmr^{\theta-2}(\log \beta) + nm(\log r)\,\mathrm{B}(\log \beta))$ word operations where $\beta = (\sqrt{r}||A||)^r$.*

2.3 The Gauss Transform

Let R be an integral domain. Let A be an $n \times n$ nonsingular matrix over R. Assume for now that all leading minors of A are nonzero. Then we can apply fraction free Gaussian elimination to recover

$$\overset{F}{\begin{bmatrix} d_0 & & & \\ * & d_1 & & \\ \vdots & & \ddots & \\ * & * & \cdots & d_{n-1} \end{bmatrix}} \overset{A}{\begin{bmatrix} * & * & \cdots & * \\ * & * & & * \\ \vdots & & \ddots & \vdots \\ * & * & \cdots & * \end{bmatrix}} = \overset{T}{\begin{bmatrix} d_1 & * & \cdots & * \\ & d_2 & \cdots & * \\ & & \ddots & \vdots \\ & & & d_n \end{bmatrix}}$$

where d_i is the principal i-th minor of A ($d_0 = 1$). If we have a $B \in \mathsf{R}^{n\times m}$ with rank profile $(j_1, j_2, \ldots, j_n)$, and A is the submatrix of B comprised of columns $j_1, j_2, \ldots, j_n$, then F is also the triangularizing adjoint of B.

We can now defining a generalization of the fraction free Gauss transform. For now, all matrices are over a field K.

Definition 2.13. *Let $A \in \mathsf{K}^{n\times m}$, $d_0 \in \mathsf{K}$ be nonzero. A* fraction free Gauss transform *of (A, d_0) is a 5-tuple (U, P, r, h, d) with $U \in \mathsf{K}^{n\times n}$ nonsingular and $P \in \mathsf{K}^{n\times n}$ a permutation which satisfy and can be written as*

$$\overset{\frac{1}{d}U}{\left[\begin{array}{c|c} \frac{d_0}{d}F & \\ \hline * & I_{n-r} \end{array}\right]} \overset{\frac{1}{d_0}PA}{\left[\begin{array}{c} * \\ \hline * \end{array}\right]} = \overset{R}{\left[\begin{array}{c} * \\ \hline \end{array}\right]} \quad \text{with} \quad P = \left[\begin{array}{c|c} * & \\ \hline & I_h \end{array}\right] \tag{2.5}$$

and where the block decomposition for $\frac{1}{d}U$, $\frac{1}{d_0}PA$ and R is conformal and:

- *r is the rank of A;*
- *h is maximal such that first $n-h$ row of A have rank r;*
- *d is equal to d_0 times the determinant of the principal block of $\frac{1}{d_0}PA$;*
- *The principal block of $\frac{1}{d_0}PA$ has full row rank r. F is equal to d_0 times the triangularizing adjoint of this block.*

Algorithm 2.14 (`Gauss`) is almost identical to Algorithm 2.8 (`GaussJordan`). Note that if (U, P, r, h, d) is a fraction free index Gauss transform for $(A, 0, 1)$ then (U, P, r, d) is a fraction free Gauss transform for A.

Algorithm 2.14. `Gauss`(A, d_0)
Input: (A, d_0) with $A \in \mathsf{K}^{n\times m}$ and $d_0 \in K$ nonzero.
Output: (U, P, r, h, d), a fraction free index GaussJordan transform.
if $A = 0$ **then**
  $(U, P, r, h, d) := (d_0 I, I, 0, n, d_0)$
else if $m = 1$ **then**
  $(U, P, r, h, d) :=$ `GaussJordan`$(A, 0, d_0)$
else
  Choose positive m_1 and m_2 with $m_1 + m_2 = m$;
  $A_1 :=$ the first m_1 columns of A;
  $(U_1, P_1, r_1, h_1, d_1) :=$ `Gauss`(A_1, d_0);
  $A_2 :=$ the trailing $(n - r_1) \times m_2$ submatrix of $\frac{1}{d_0}U_1 P_1 A$;
  $(U_2, P_2, r_2, h_2, d) :=$ `Gauss`(A_2, d_1);
  $\bar{P}_2 := \mathrm{diag}(I_{r_1}, P_2)$;
  $\bar{U}_2 := \mathrm{diag}(d_1 I_{r_1}, U_2)$;
  $(U, P, r, h, d) := (\frac{1}{d_1}\bar{U}_2(\bar{P}_2(U_1 - d_1 I) + d_1 I, \bar{P}_2 P_1, r_1 + r_2, \min(h_1, h_2), d)$;
fi;
return (U, P, r, h, d);

Proposition 2.15. *Let $A \in \mathsf{R}^{n\times m}$, R an integral domain. A fraction free Gauss transform (U, P) for A can be recovered in $O(nmr^{\theta-2})$ basic operations of type* Arith *plus $O(nm \log r)$ basic operations of type* Div.

Corollary 2.16. *Let $\mathsf{R} = \mathbb{Z}$. The complexity bound of Proposition 2.15 becomes $O(nmr^{\theta-2}(\log\beta) + nmr(\log r)\,\mathrm{B}(\log\beta))$ word operations where $\beta = (\sqrt{r}||A||)^r$.*

2.4 The Modified Gauss Transform

Algorithms 2.8 (`GaussJordan`) and 2.14 (`Gauss`) used permutation matrices (i.e. row interchanges) during the elimination to ensure that pivots would be nonzero. In some applications it is desirable to actually "condition" the pivot. For this purpose we use a subroutine `Cond` that takes as input a nonzero $A \in \mathsf{K}^{n\times 1}$ and returns a unit upper triangular

$$\begin{array}{ccc} C & A & \\ \begin{bmatrix} 1 & c_2 & c_3 & \cdots & c_n \\ & 1 & & & \\ & & 1 & & \\ & & & \ddots & \\ & & & & 1 \end{bmatrix} & \begin{bmatrix} a_1 \\ a_2 \\ a_3 \\ \vdots \\ a_n \end{bmatrix} & = \begin{bmatrix} \bar{a}_1 \\ a_2 \\ a_3 \\ \vdots \\ a_n \end{bmatrix} \end{array} \tag{2.6}$$

with $\bar{a}_1 = a_1 + c_2 a_2 + \cdots + c_n a_n$ nonzero and possibly satisfying an additional property. What the additional property should be will depend on application (see Section 5.3).

The next definition is identical to Definition 2.13 except that the role of the permutation matrix P is replaced by a unit upper triangular C and the paramater h is omitted.

Definition 2.17. *Let $A \in \mathsf{K}^{n\times m}$, $d_0 \in \mathsf{K}$ be nonzero.* A modified fraction-free Gauss transform *of (A, d_0) is a 4-tuple (U, C, r, d) with $U \in \mathsf{K}^{n\times n}$ nonsingular and $C \in \mathsf{K}^{n\times n}$ unit upper triangular which satisfy and can be written using a conformal block decomposition as*

$$\begin{array}{ccc} \frac{1}{d}U & \frac{1}{d_0}CA & R \\ \left[\begin{array}{c|c} \frac{d_0}{d}F & \\ \hline * & I_{n-r} \end{array}\right] & \left[\begin{array}{c} * \\ \hline * \end{array}\right] & = \left[\begin{array}{c} * \\ \hline \end{array}\right] \end{array} \quad \textit{with} \quad C = \left[\begin{array}{c|c} * & * \\ \hline & I_{n-r} \end{array}\right] \tag{2.7}$$

and where:

- *r is the rank of A;*
- *d is equal to d_0 times the determinant of the principal block of $\frac{1}{d_0}CA$;*
- *The principal block of $\frac{1}{d_0}CA$ has full row rank r. F is equal to d_0 times the triangularizing adjoint of this block.*

Algorithm 2.18 (`CondGauss`) is a straightforward modification of Algorithm 2.14 (`Gauss`). The merging of the two subproblems is now based on Lemmas 2.3 and 2.4 instead of Lemma 2.5. Note that if (U, C, r, d) is a modified fraction free index Gauss transform for $(A, 0, 1)$, then r is the rank of A and (U, I) is a fraction free Gauss transform for CA.

Proposition 2.19. *Let $A \in \mathsf{R}^{n\times m}$, R an integral domain. Not counting the calls to subroutind* `Cond`*, a modified fraction free Gauss transform (U, C) for A can be recovered in $O(nmr^{\theta-2})$ basic operations of type* Arith *plus $O(nm \log r)$ basic operations of type* Div.

Algorithm 2.18. `CondGauss`(A, d_0)
Input: (A, d_0) with $A \in \mathsf{K}^{n\times m}$ and $d_0 \in K$ nonzero.
Output: (U, C, r, d) a modified fraction free index Gauss transform.
if $A = 0$ **then**
  $(U, C, r, h, d) := (d_0 I, I, 0, n, d_0)$
else if $m = 1$ **then**
  $C :=$ `Cond`(A);
  $(U, *, r, *, d) :=$ `GaussJordan`$(CA, 0, d_0)$
else
  Choose positive m_1 and m_2 with $m_1 + m_2 = m$;
  $A_1 :=$ the first m_1 columns of A;
  $(U_1, C_1, r_1, d_1) :=$ `CondGauss`(A_1, d_0);
  $A_2 :=$ the trailing $(n - r_1) \times m_2$ submatrix of $\frac{1}{d_0}U_1 C_1 A$;
  $(U_2, C_2, r_2, d) :=$ `CondGauss`(A_2, d_1);
  $\bar{C}_2 := \operatorname{diag}(I_{r_1}, C_2)$;
  $\bar{U}_2 := \operatorname{diag}(d_1 I_{r_1}, U_2)$;
  $(U, C, r, d) := (\frac{1}{d_1}\bar{U}_2(\bar{C}_2(U_1 - d_1 I) + d_1 I), C_1 + C_2 - I, r_1 + r_2, d)$;
fi;
return (U, C, r, d);

Triangular Factorization over a Stable PIR

In this subsection we show how to apply algorithm `CondGauss` to a problem over a stable PIR R. Given a unimodular $V \in \mathsf{R}^{n\times n}$, we want to recover a unit lower triangular L such that V can be expressed as the product $L *_1 *_2$ for some upper triangular $*_1$ and lower triangular and $*_2$. (We neglect to name the matrices $*_1$ and $*_2$ because we won't need them.) The algorithm makes use of Subroutine 2.20 (`Cond`).

Subroutine 2.20. `Cond`(A)
Input: Nonzero $A \in \mathsf{R}^{n\times 1}$, R a stable PIR.
Output: $C \in \mathsf{R}^{n\times n}$ as in (2.6) with $(a_1, \ldots, a_n) = (a_1 + c_2 a_2 + \cdots + c_n a_n)$.
$C := I_n$;
$a := A[1, 1]$;
for i **from** 2 **to** n **do**
  $C[1, i] := \operatorname{Stab}(a, A[i, 1], 0)$;
  $a := a + C[1, i]\, A[i, 1]$
od;
return C;

Lemma 2.21. *Let $V \in \mathsf{R}^{n\times n}$ be unimodular, R a stable ring. Then V can be expressed as the product $L *_1 *_2$ where L is unit lower triangular, $*_1$ is upper triangular and $*_2$ is lower triangular. A matrix L satisfying these requirements can be computed in $O(n^\theta)$ basic operations of type* {Arith, Stab, Unit}.

Corollary 2.22. *Let $\mathsf{R} = \mathbb{Z}/(N)$. The complexity bound of Lemma 2.21 becomes $O(n^\theta(\log\beta) + n^2(\log n)\,\mathrm{B}(\log\beta))$ word operations where $\beta = nN$.*

Proof. Compute a modified Gauss transform $(F, R, *, *, *)$ of A^T and let $T = FRA^T$. Let D be the diagonal matrices with same diagonal entries as T. Then

$$A = \overbrace{(D^{-1}T)^T}^{L}\overbrace{((D^{-1}FD)^T)^{-1}}^{*_1}\overbrace{((DR)^T)^{-1}}^{*_2}.$$

□

2.5 The Triangularizing Adjoint

Let $A \in \mathsf{R}^{n\times n}$, R an integral domain. It is a classical result that the determinant of A can be computed in $O(n^\theta)$ ring operations. Here we show that all leading minors of A can be recovered in the same time.

Recall that the adjoint of A is the matrix A^{adj} with entry A^{adj}_{ij} equal to $(-1)^{i+j}$ times the determinant of the submatrix obtained from A by deleting row j and column i. The adjoint satisfies $A^{\mathsf{adj}}A = AA^{\mathsf{adj}} = dI$ where $d = \det A$. The point is that the adjoint is defined even when A is singular and might even be nonzero in this case.

We define the *triangularizing adjoint* F of A to be the lower triangular matrix with first i entries in row i equal to the last row of the adjoint of the principal i-th submatrix of A. F and the corresponding *adjoint triangularization* T of A satsify

$$\overbrace{\begin{bmatrix} d_0 & & & \\ * & d_1 & & \\ \vdots & & \ddots & \\ * & * & \cdots & d_{n-1} \end{bmatrix}}^{F}\overbrace{\begin{bmatrix} * & * & \cdots & * \\ * & * & & * \\ \vdots & & \ddots & \vdots \\ * & * & \cdots & * \end{bmatrix}}^{A} = \overbrace{\begin{bmatrix} d_1 & * & \cdots & * \\ & d_2 & \cdots & * \\ & & \ddots & \vdots \\ & & & d_n \end{bmatrix}}^{T}$$

where d_i is the principal i-th minor of A ($d_0 = 1$).

We present an algorithm that requires $O(n^\theta)$ multiplication and subtractions plus $O(n^2 \log n)$ exact divisions from R to compute F. In addition to solving the "all leading minors" problem, recovering F and T is the main computational step in randomized algorithms for computing normal forms of matrices over various rings, including the integer Smith form algorithm of Giesbrecht (1995a), the parallel algorithms for polynomial matrices by Kaltofen *et al.* (1987) and the sequential version thereof for the ring $\mathbb{Q}[x]$ given by Storjohann and Labahn (1997).

Notes

In the special case when all the d_i's are nonzero, F and T can be recovered using standard techniques like Gaussian elimination, asymptotically fast LU decomposition described in Aho *et al.* (1974) or the algorithm for Gauss transform of the previous chapter. In the general case these algorithms are not applicable since the d_i's, which correspond to pivots during the elimination, are required to be nonzero. The obvious approach to deal with the general case is to compute the adjoint of all leading submatrices of A. This costs $O(n^{\theta+1})$ ring operations, and, as far as we know, is the best previous complexity for recovering all leading minors of A.

Algorithms which require $O(n^3)$ operations for computing the adjoint of A in the general case (when A may be singular) have been given by Shapiro (1963) and Cabay (1971). There, the application was to avoid the problem of bad primes when solving a nonsingular linear system. Inspired by these results, we developed in (Storjohann and Labahn, 1997) an $O(n^3)$ field operations algorithm for computing the triangularizing adjoint. In (Storjohann, 1996b) we reduce this to $O(n^\theta \log n)$ ring operations, but the algorithm is not fraction free.

The Adjoint Transform

Let K be a field. Given a square matrix A over K and $d_0 \in F$ nonzero, we define the *adjoint transform* of (A, d_0) to be the matrix obtained by multiplying the triangularizing adjoint of $\frac{1}{d_0}A$ by d_0. Note that the triangularizing adjoint of A is given by the adjoint transform of $(A, 1)$. Our recursive algorithm for computing the adjoint transform is completely described by the following technical lemma.

Lemma 2.23. *For $A \in \mathsf{K}^{n\times n}$ and $d_0 \in \mathsf{K}$ nonzero, let F be the adjoint transform of (A, d_0). Let A_1 be the first m columns of A, and let*

(U, P, r, h, d) be a fraction-free Gauss transform for (A_1, d_0). Let F_1 be the adjoint transform of (B_1, d_0) where B_1 is the principal $\min(m, r+1)$*th submatrix of A.*

1) The principal $\min(m, r+1)$*th submatrix of F equals F_1.*

2) If $P = I_n$ then the principal $\min(m, r+1)$*th submatrix of U equals F_1.*

3) If $r < m$ then the last $n - r - 1$ rows of F are zero.

Assume henceforth that $r = m$. Let F_2 be the adjoint transform of (B_2, d) where B_2 is the trailing $(n-m) \times (n-m)$ submatrix of UPA. Then:

4) Rows $m+1, m+2, \ldots, n-h-1$ of F are zero.

5) The last $\min(n-m, h+1)$ *rows of F are given by the last* $\min(n-m, h+1)$ *rows of* $F_2 \left[\begin{array}{c|c} V & I \end{array} \right] P \det P$ *where V is the submatrix of $\frac{1}{d}U$ comprised of the last $n-m$ rows and first m columns.*

Proof. We prove each statement of the lemma in turn.

1 & 2) Obvious

3) If $r < m$, then for $i = m+1, \ldots, n$ the principal $i \times (i-1)$ submatrix of $\frac{1}{d_0}A$ is rank deficient; since entries in row i of $\frac{1}{d_0}F$ are $(i-1) \times (i-1)$ minors of these submatrices, they must be zero.

4) Now assume that $r = m$. By definition 2.13, the principal $(n - h - 1) \times m$ submatrix of A is rank deficient. Now apply the argument used to prove statement 3).

5) By definition of an index echelon transform, we have

$$\left[\begin{array}{c|c} I_m & \\ \hline V & I \end{array} \right] \overset{PA}{\left[\begin{array}{c|c} * & * \\ \hline * & * \end{array} \right]} = \left[\begin{array}{c|c} * & * \\ \hline & (1/d)B_2 \end{array} \right].$$

Since $\frac{1}{d}F_2$ is the triangularizing adjoint of $\frac{1}{d}B_2$ and d is d_0 times the determinant of the upper left block of PA, we may deduce that

$$\frac{1}{d_0} F_2 \left[\begin{array}{c|c} V & I \end{array} \right] \tag{2.8}$$

is equal to the last $n-m$ rows of the triangularizing adjoint of $\frac{1}{d_0}PA$. Now fix some i with $n-h-1 \le i \le n$, and let $\bar{A}$ and $\bar{P}$ be the principal $i \times i$ submatrices of A and P respectively. Since $P = \operatorname{diag}(*, I_h)$ we have that $\bar{P}\bar{A}$ is equal to principal ith submatrix of PA. By definition, the

first i entries in row i of the triangularizing adjoint of PA is given by the last row of $(\bar{P}\bar{A})^{\mathsf{adj}}$. Similarly, the first i entries in row i of $\frac{1}{d_0}F$ are given by the last row of $\bar{A}^{\mathsf{adj}}$. The result now follows from the claim about (2.8) and by noting that $(\bar{P}\bar{A})^{\mathsf{adj}}\bar{P}\det\bar{P} = \bar{A}^{\mathsf{adj}}$. □

Note that if $A \in \mathsf{K}^{1\times 1}$, then the adjoint transform of (A, d_0) is $[d_0]$. Correctness of Algorithm 2.24 (`AdjointTransform`) follows as a corollary to Lemma 2.23. The analysis of the algorithm is straightforward.

Algorithm 2.24. `AdjointTransform`(A, d_0)
Input: (A, d_0) with $A \in \mathsf{K}^{n\times n}$ and $d_0 \in K$ nonzero.
Output: $F \in \mathsf{K}^{n\times n}$, the adjoint transform of (A, d_0).

if $m = 1$ **then**
  $F := [d_0]$;
else
  $F :=$ the $n \times n$ zero matrix;
  Choose m satisfying $1 \le m < n$;
  $A_1 :=$ the first m columns of A;
  $(U, P, r, h, d) :=$ `Gauss`(A_1, d_0);
  if $P = I_n$ **then**
    $F_1 :=$ the principal $\min(m, r+1)$th submatrix of U;
  else
    $B_1 :=$ the principal $\min(m, r+1)$th submatrix of A;
    $F_1 :=$ `AdjointTransform`(B_1, d_0);
  fi;
  Set principal $\min(m, r+1)$th submatrix of F equal to F_1;
  if $r = m$ **then**
    $B_2 :=$ the trailing $(n-m)$th submatrix of UPA;
    $F_2 :=$ `AdjointTransform`(B_2, d);
    $k := \min(n-m, h+1)$;
    Set last k rows of F to last k rows of $\frac{1}{d}\operatorname{diag}(0, F_2)UP\det P$;
  fi;
fi;
return F;

Proposition 2.25. *Let $A \in \mathsf{R}^{n\times n}$, R an integral domain. The triangularizing adjoint of A can can be computed in $O(n^\theta)$ basic operations of type* Arith *plus $O(n^2 \log n)$ basic operations of type* Div.

Corollary 2.26. *Let $\mathsf{R} = \mathbb{Z}$. The complexity bound of Proposition 2.25 becomes $O(n^\theta(\log\beta) + n^2(\log n)\,\mathrm{B}(\log\beta))$ word operations where $\beta = (\sqrt{r}||A||)^r$.*

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 $0 \le i \le r$ 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 $1 \le k < i \le r$.

(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 $A[r+1, k] \ne 0$ **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 $A[r, k] \ne 0$ **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 $O(nmr + n^2r)$ 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 $O(nmr + n^2r)$ 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

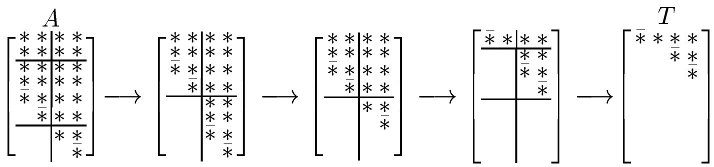


Figure 3.1: Subroutine for transformation to echelon form

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

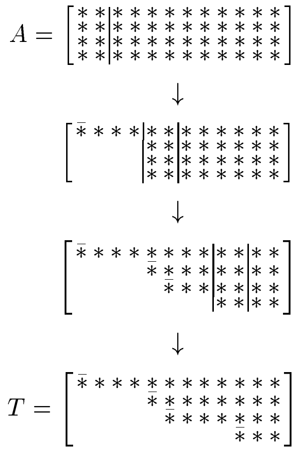


Figure 3.2: Left-to-right transformation to echelon form

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

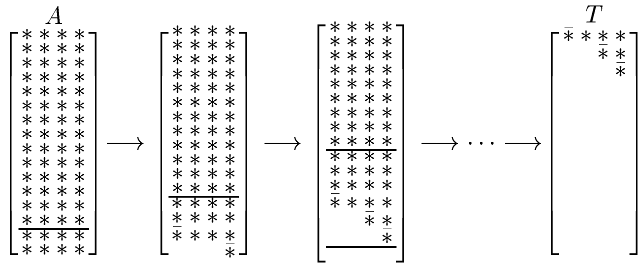


Figure 3.3: Bottom-to-top transformation to echelon form

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. □

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

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

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 $d_i + r_i \leq 2d_i$. 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 $r_{i+1} \geq r_i$.

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 $d_i \leq r$, 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 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 aforementioned algorithm on input $A \in \mathsf{R}^{n\times m}$. The function $r(A)$ is well defined because the algorithm is deterministic.

For $r \geq 0$, let $f_{m,r}(n)$ be a bound on the number of basic operations required by the algorithm with input from $\{A \in \mathsf{R}^{n\times m} \mid r(A) \leq r\}$. 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

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

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

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

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

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

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

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

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 $\mathbb{Z}$.*

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

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 $\{Arith, \mathrm{Gcdex}, \mathrm{Stab}\}$.

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 $\mathrm{assoc}(\mathsf{R})$.

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 4

The 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.

Let R be a PIR. For a matrix $A \in \mathsf{R}^{n \times m}$ we write $\mathrm{S}(A)$ to mean the set of all R-linear combinations of rows of A and $\mathrm{S}_j(A)$ to mean the subset of $\mathrm{S}(A)$ comprised of all rows which have first j entries zero. Corresponding to every $A \in \mathsf{R}^{* \times m}$ is an $H \in \mathsf{R}^{* \times m}$ that satisfies:

(r1) Let r be the number of nonzero rows of H. Then the first r rows of H are nonzero. For $0 \le i \le r$ 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 $1 \le k < i \le r$.

(r4) Rows $i+1, i+2, \ldots, r$ of H generate $\mathrm{S}_{j_i}(A)$.

H is the Howell canonical form of A. The first r rows of H — the Howell basis of A — give a canonical generating set for $\mathrm{S}(A)$.

Assume that $n \ge r$. A Howell transform for A is a tuple (Q, U, C, W, r) which satisfies and can be written using a conformal block decomposition as

$$\begin{array}{c} W \\ \left[\begin{array}{c|c} * & \\ \hline & I \end{array}\right] \end{array} \quad \text{and} \quad \begin{array}{c} Q \\ \left[\begin{array}{c|c} I & \\ \hline * & I \end{array}\right] \end{array} \begin{array}{c} U \\ \left[\begin{array}{c|c} * & \\ \hline & I \end{array}\right] \end{array} \begin{array}{c} C \\ \left[\begin{array}{c|c} I & * \\ \hline & I \end{array}\right] \end{array} \begin{array}{c} A \\ \left[\begin{array}{c} * \\ \hline * \end{array}\right] \end{array} = \begin{array}{c} T \\ \left[\begin{array}{c} H \\ \hline \\ \end{array}\right] \end{array} \tag{4.1}$$

with $U \in \mathsf{R}^{n \times n}$ unimodular, H the Howell basis for A and W a kernel for T, that is $W \in \mathsf{R}^{n \times n}$ and $\mathrm{S}(W) = \{v \in \mathsf{R}^n \mid vA = 0\}$. In this chapter we give an algorithm for Howell transform that requires $O(nmr^{\theta-2})$ basic operations of type {Arith, Gcdex, Unit, Quo Stab}. This assumes that A has been augmented (if necessary) to have at least r zero rows. The need for r zero rows disappears if R is a stable ring.

Being able to compute a Howell transform leads directly to fast algorithms for solving a variety of linear algebra problems over R. To motivate the definition of the Howell transform we consider some of these now. Let $A_i \in \mathsf{R}^{n_i \times m}$ be given for $i = 1, 2$. By augmenting with 0-rows we may assume without loss of generality that $n = n_1 \ge n_2 \ge m$. Let $(Q_i, U_i, C_i, W_i, r_i)$ be a Howell transform for A_i and H_i the Howell basis of A_i for $i = 1, 2$. For convenience, let (A, Q, U, C, W, r) denote $(A_1, Q_1, U_1, C_1, W_1, r_1)$.

Kernel computation [Find a kernel $Y \in \mathsf{R}^{n \times n}$ for A.]
$WQUC$ is a kernel for A, but this product may be dense and requires $O(n^2 m^{\theta-2})$ operations to compute. Alternatively, return the decomposition

$$Y = \begin{array}{c} WQU \\ \left[\begin{array}{c|c} * & \\ \hline * & I_{n-r} \end{array}\right] \end{array} \begin{array}{c} C \\ \left[\begin{array}{c|c} I_r & * \\ \hline & I_{n-r} \end{array}\right] \end{array}. \tag{4.2}$$

Equality of spans. [Determine if $\mathrm{S}(A_1) = \mathrm{S}(A_2)$.]
The spans will be equal precisely when $H_1 = H_2$. If this is the case, a transformation matrix P such that $A_1 = PA_2$ and $P^{-1}A_1 = A_2$ is given by $P = (Q_1 U_1 C_1)^{-1} Q_2 U_2 C_2$. A straightforward multiplication will verify that

$$P = \begin{array}{c} (2I - C_1) \\ \left[\begin{array}{c|c} I_r & * \\ \hline & I_{n-r} \end{array}\right] \end{array} \begin{array}{c} ((Q_2 - Q_1)U_1 + I) \\ \left[\begin{array}{c|c} I_r & \\ \hline * & I_{n-r} \end{array}\right] \end{array} \begin{array}{c} U_1^{-1}U_2 \\ \left[\begin{array}{c|c} * & \\ \hline & I_{n-r} \end{array}\right] \end{array} \begin{array}{c} C_2 \\ \left[\begin{array}{c|c} I_r & * \\ \hline & I_{n-r} \end{array}\right] \end{array}.$$

Sum of modules. [Find the Howell basis for $\mathrm{S}(A_1) + \mathrm{S}(A_2)$.]
Return the Howell basis for $\left[\begin{array}{c} A_1 \\ \hline A_2 \end{array}\right]$.

Intersection of modules. [Find the Howell basis for $\mathrm{S}(A_1) \cap \mathrm{S}(A_2)$.]
Compute a kernel Y for $\left[\begin{array}{c} A_1 \\ \hline A_2 \end{array}\right]$. Return the Howell basis for $Y \left[\begin{array}{c} A_1 \\ \hline \\ \end{array}\right]$.

Testing containment. [Determine whether or not $b \in \mathrm{S}(A)$.]
Recover a row vector y such that

$$\left[\begin{array}{c|c} 1 & y \\ \hline & I \end{array}\right] \left[\begin{array}{c|c} 1 & b \\ \hline & H \end{array}\right] = \left[\begin{array}{c|c} 1 & b' \\ \hline & H \end{array}\right]$$

with the right hand side in Howell form. Then $b \in \mathrm{S}(A)$ if and only if $b' = 0$. If $b \in \mathrm{S}(A)$, then $xA = b$ where $x \leftarrow [\, y \mid 0 \,]UC$.

Notes

Existence and uniqueness of the Howell form was first proven by Howell (1986) for matrices over $\mathbb{Z}/(N)$. Howell (1986)'s proof is constructive and leads to an $O(n^3)$ basic operations algorithm. Buchmann and Neis (1996) give a proof of uniqueness over an arbitrary PIR and propose efficient algorithms for solving a variety of linear algebra problems over rings. Buchmann and Neis call the first r rows of H a standardized generating set for $\mathrm{S}(A)$.

This chapter is joint work with Thom Mulders. An earlier version appears in (Mulders and Storjohann, 1998).

4.1 Preliminaries

Let $A \in \mathsf{R}^{n\times m}$. We say A is in weak Howell form if A satisfies (r1) and (r3) but not necessarily (r2).

Lemma 4.1. *Let*

$$H = \left[\begin{array}{c|c} H_1 & F \\ \hline & H_2 \end{array}\right] \quad \text{and} \quad K = \left[\begin{array}{c|c} K_1 & -S \\ \hline & K_2 \end{array}\right].$$

where K_i is a kernel for H_i, $i = 1, 2$. If $K_1F = SH_2$ then K is a kernel for H. If in addition H_1 and H_2 are in weak Howell form and H_1 has no zero row, then H is in weak Howell form.

Lemma 4.2. *If $(Q_1, U_1, C_1) =$*

$$\left(\left[\begin{array}{c|c|c|c} I_k & & & \\ \hline & I_{r_1} & & \\ \hline & \bar{q}_1 & I_{r_2} & \\ \hline & q_1 & & I \end{array}\right], \left[\begin{array}{c|c|c|c} I_k & & & \\ \hline & u_1 & & \\ \hline & & I_{r_2} & \\ \hline & & & I \end{array}\right], \left[\begin{array}{c|c|c|c} I_k & & & \\ \hline c_1 & I_{r_1} & \bar{d}_1 & d_1 \\ \hline & & I_{r_2} & \\ \hline & & & I \end{array}\right] \right),$$

and $(Q_2, U_2, C_2) =$

$$\left(\left[\begin{array}{c|c|c|c} I_k & & & \\ \hline & I_{r_1} & & \\ \hline & & I_{r_2} & \\ \hline & & q_2 & I \end{array}\right], \left[\begin{array}{c|c|c|c} I_k & & & \\ \hline & I_{r_1} & & \\ \hline & & u_2 & \\ \hline & & & I \end{array}\right], \left[\begin{array}{c|c|c|c} I_k & & & \\ \hline & I_{r_1} & & \\ \hline c_2 & \bar{c}_2 & I_{r_2} & d_2 \\ \hline & & & I \end{array}\right] \right)$$

and

$$W_1 = \left[\begin{array}{c|c|c|c} I_k & w_1 & & \\ \hline & \bar{w}_1 & & \\ \hline & & I_{r_2} & \\ \hline & & & I \end{array}\right]$$

are all in $\mathsf{R}^{n\times n}$ and the block decomposition is conformal, then

$$Q_2U_2((C_2 - I)W_1 + I)Q_1U_1C_1 = QUC$$

where $(Q, U, C) =$

$$\left(\left[\begin{array}{c|c|c|c} I_k & & & \\ \hline & I_{r_1} & & \\ \hline & & I_{r_2} & \\ \hline & q_1 & q_2 & I \end{array}\right], \left[\begin{array}{c|c|c|c} I_k & & & \\ \hline & u_1 & u_{12} & \\ \hline & u_{21} & u_{22} & \\ \hline & & & I \end{array}\right], \left[\begin{array}{c|c|c|c} I_k & & & \\ \hline c_{11} & I_{r_1} & & c_{12} \\ \hline c_2 & & I_{r_2} & d_2 \\ \hline & & & I \end{array}\right] \right)$$

with

$$\begin{aligned} c_{11} &= c_1 - \bar{d}_1c_2 \\ c_{12} &= d_1 - \bar{d}_1d_2 \\ u_{21} &= u_2(\bar{q}_1 + d_2q_1 + c_2w_1 + \bar{c}_2\bar{w}_1)u_1 \\ u_{12} &= u_1\bar{d}_1 \\ u_{22} &= u_2 + u_{21}\bar{d}_1 \end{aligned}$$

Moreover, there exists a subrouting `Combine` *that takes as input (Q_1, U_1, C_1, Q_2, U_2, C_2), produces as output (Q, U, C), and has cost bounded by $O(n \max(r_1, r_2) \min(r_1, r_2)^{\theta-1})$ arithmetic operations.*

4.2 The Howell Transform

We begin by defining a generalization of the Howell transform.

Definition 4.3. *Let $A \in \mathsf{R}^{n\times m}$ and $k \in \mathbb{N}$ satisfy $0 \le k \le n$ and $n - k \ge r$ where r is the number of nonzero rows in the Howell form of A. An* index weak Howell transform *of (A, k) is an 5-tuple (Q, U, C, W, r) which satisfies and can be written using a conformal block decomposition as*

$$\overset{W}{\left[\begin{array}{c|c|c} I & -S & \\ \hline & K & \\ \hline & & I \end{array}\right]} \quad \text{and} \quad \overset{Q}{\left[\begin{array}{c|c|c} I & & \\ \hline & I & \\ \hline & * & I \end{array}\right]} \overset{U}{\left[\begin{array}{c|c|c} I & & \\ \hline & * & \\ \hline & & I \end{array}\right]} \overset{C}{\left[\begin{array}{c|c|c} I & & \\ \hline * & I & * \\ \hline & & I \end{array}\right]} \overset{A}{\left[\begin{array}{c} \bar{A} \\ \hline * \\ \hline * \end{array}\right]} = \overset{T}{\left[\begin{array}{c} \bar{A} \\ \hline H \\ \hline \ \end{array}\right]} \tag{4.3}$$

with U unimodular, $\bar{A}$ the first k rows of A, H a weak Howell basis for A, W a kernel for T, K a kernel for H and S such that $\bar{A} = SH$.

Algorithm 4.4 (`Weak Howell`) is given on page 76.

Theorem 4.5. *Algorithm 4.4 is correct.*

Proof. Assume for now that R is a stable ring. We need to show that the tuple (Q, U, C, W, r) returned by the algorithm satisfies Definition 4.3. Use induction on (m, r). It is easy to verify by comparing with Definition 4.3 that the two base cases are correct (when $r = 0$ and/or $m = 1$).

Now assume $m > 1$ and $r > 0$ and choose positive m_1 and m_2 with $m_1 + m_2 = m$. Let B, $(Q_1, U_1, C_1, W_1, r_1)$, $(Q_2, U_2, C_2, W_2, r_2)$

and (Q, U, C, W, r) be as computed in the algorithm. By induction $(Q_1, U_1, C_1, W_1, r_1)$ and $(Q_2, U_2, C_2, W_2, r_2)$ are computed correctly. Then

$$\begin{array}{cccccc} & \begin{array}{cc} A_1 & B \end{array} & & W_1 & Q_1U_1C_1A & & A_2 \\ W_1Q_1U_1C_1 & \left[\begin{array}{c|c} \bar{A}_1 & E \\ \hline * & * \\ \hline * & * \end{array}\right] = & \left[\begin{array}{c|c|c} I & -S_1 & \\ \hline & K_1 & \\ \hline & & I \end{array}\right] & \left[\begin{array}{c|c} \bar{A}_1 & E \\ \hline H_1 & F \\ \hline & * \end{array}\right] = & \left[\begin{array}{c|c} & E - S_1F \\ \hline & K_1F \\ \hline & * \end{array}\right] \end{array} \tag{4.4}$$

where H_1 is in weak Howell form with no zero rows and E and F are new labels.

W_1 and W_2 can be partitioned conformally and satisfy

$$\begin{array}{ccc} W_2 & W_1 & W_1 + W_2 - I \\ \left[\begin{array}{c|c|c|c} I & & -S_{21} & \\ \hline & I & -S_{22} & \\ \hline & & K_2 & \\ \hline & & & I \end{array}\right] & \left[\begin{array}{c|c|c|c} I & -S_1 & & \\ \hline & K_1 & & \\ \hline & & I & \\ \hline & & & I \end{array}\right] = & \left[\begin{array}{c|c|c|c} I & -S_1 & -S_{21} & \\ \hline & K_1 & -S22 & \\ \hline & & K_2 & \\ \hline & & & I \end{array}\right] \end{array}.$$

Let

$$H = \left[\begin{array}{c|c} H_1 & F \\ \hline & H_2 \end{array}\right], \quad K = \left[\begin{array}{c|c} K_1 & -S_{22} \\ \hline & K_2 \end{array}\right] \text{ and } S = \left[\begin{array}{c|c} S_1 & S_{21} \end{array}\right].$$

Then $SH = \bar{A}$ (direct computation). Let T be the matrix in (4.3). From Lemma 4.1 we get that H is a weak Howell basis for T, K is a kernel for H and W is a kernel for T.

Note that $(C_2 - I)W_1 + I$ has the same block structure as C_2 and thus is unimodular. A straightforward multiplication verifies that $Q_2U_2((C_2 - I)W_1 + I)Q_1U_1C_1A = T$. Then T is left equivalent to A hence H is a weak Howell basis also for A. This also shows that $r_1 + r_2 = r$.

Now assume that the input matrix A has rows $k+1, k+2, \ldots, k+r$ zero. Using induction we can show that all subproblems will satisfy the same condition and hence $A[k+1, 0]$ will also be zero. In this case no operations of type Stab will be required. □

Theorem 4.6. *Let $A \in \mathsf{R}^{n\times m}$ and k be such that (A, k) is valid input to algorithm* `WeakHowell`. *The algorithm requires $O(nmr^{\theta-2})$ basic operations of type* {Arith, Gcdex, Ann, Stab}.

Proof. Similar to the proof of Theorem 2.10. □

Corollary 4.7. *Let $\mathsf{R} = \mathbb{Z}_N$. The complexity bound of Theorem 4.6 becomes $O(nmr^{\theta-2}(\log\beta) + nm(\log r)\,\mathrm{B}(\log\beta))$ word operations where $\beta = rN$.*

Proposition 4.8. *Let $A \in \mathsf{R}^{n\times m}$. If either R is a stable ring or A has at least first r rows zero (where r is the number of rows in the Howell basis of A) then a Howell transform for A can be computed in $O(nmr^{\theta-2})$ basic operations of type* {Arith, Gcdex, Ann, Quo, Stab}.

Proof. Compute an index weak Howell transform (Q, U, C, W, r) for $(A, 0)$. Now recover an upper triangular and unimodular

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

such that $RQ\bar{U}CA$ is in Hermite (and Howell) form. (See the proof of Theorem 3.15.) Then $(RQR^{-1}, RU, C, WR^{-1}, r)$ is a Howell transform for A. The inverse of R can be computed in the allotted time using Theorem 3.15. □

Corollary 4.9. *Let $A \in \mathbb{Z}_N^{n\times m}$ have $n \geq r$ where r is the number of rows in the Howell basis of A. A Howell transform (Q, U, C, W, r) for A can be recovered in $O(nmr^{\theta-2}(\log\beta) + nm(\log r)\,\mathrm{B}(\log\beta))$ word operations where $\beta = rN$.*

Algorithm 4.4. `WeakHowell`(A, k)
Input: (A, k) with $A \in \mathsf{R}^{n\times m}$ and $0 \le k \le n$.
Output: (Q, U, C, W, r), an index weak Howell transform for (A, k).
Caveat: The inequality $k + r \le n$ must be satisfied. Either R should be stable ring or A have rows $k+1, k+2, \ldots, k+r$ zero.

if $A = 0$ **then**
  $(Q, U, C, W, r) := (I_n, I_n, I_n, I_n, 0)$;
else if $m = 1$ **then**
  $Q,\ U,\ C,\ W,\ r,\ a := I_n,\ I_n,\ I_n,\ I_n,\ 1,\ A[k+1, 1]$;
  if $A[k+1, 1] = 0$ **then**
    $E :=$ the $1 \times n$ matrix as in Proposition 3.5
  fi;
  for i **to** n **do**
    if $i = k+1$ **then next fi**;
    if $A[k+1, 1] = 0$ **then**
      $c := E[i]$
    else
      $c := \mathrm{Stab}(a, A[i, 1], 0)$
    fi;
    $C[k+1, i] := c$;
    $a := a + cA[i, 1]$;
  od;
  $W[k+1, k+1] := \mathrm{Ann}(a)$;
  for i **to** k **do** $W[i, k+1] := -\mathrm{Div}(A[i, 1], a)$ **od**;
  for i **from** $k+2$ **to** n **do** $Q[i, k+1] := -\mathrm{Div}(A[i, 1], a)$ **od**;
else
  Choose positive m_1 and m_2 with $m_1 + m_2 = m$;
  $A_1 :=$ the first m_1 columns of A;
  $B :=$ the last m_2 columns of A;
  $(Q_1, U_1, C_1, W_1, r_1) :=$ `WeakHowell`(A_1, k);
  $A_2 := W_1 Q_1 U_1 C_1 B$;
  $(Q_2, U_2, C_2, W_2, r_2) :=$ `WeakHowell`$(A_2, k + r_1)$;
  $(Q, U, C) :=$ `Combine`$(Q_1, U_1, C_1, Q_2, U_2, C_2, W_1)$;
  $W := W_1 + W_2 - I$;
  $r := r_1 + r_2$;
fi;
return (Q, U, C, W, r);

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.

Throughout the chapter:

- R is a PID,
- $A \in \mathsf{R}^{n\times m}$ has rank profile $(j_1, j_2, \ldots, j_r)$, and
- H is the Hermite form of A.

Let $\bar{\mathsf{R}}$ denote the fraction field of R. Note that for nonzero $N \in \mathsf{R}$, the residue class ring $\mathsf{R}/(N)$ will be a PIR but not, in general, a PID. Recall the relationship between these rings in Figure 5. We are going to explore

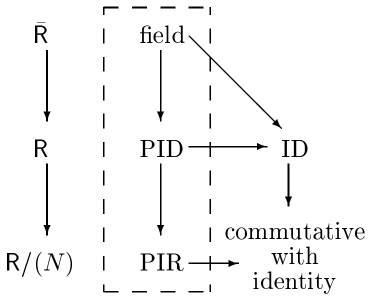


Figure 5.1: Relationship between rings

the relationship between these rings from the point of view of recovering various matrix invariants over R. The concrete rings which will most interest us are $\mathsf{R} = \mathbb{Z}$ (whereby $\bar{\mathsf{R}} = \mathbb{Q}$ and N is composite) and $\mathsf{R} = \mathbb{Q}[x]$ (whereby $\bar{\mathsf{R}} = \mathbb{Q}(x)$ and $N \in \mathbb{Q}[x]$ has a nontrivial factorization).

The rest of this chapter is organised as follows. First we establish some notation and recall sundry useful facts about vector spaces and lattices. Then in Section 5.1 we give a self contained treatment of the well known modular determinant method for computing the Hermite basis H over the residue class ring $\mathsf{R}/(N)$ for well chosen N. In Section 5.2 we show how to recover H by computing, in a fraction free way, over the fraction field R. In other words, we view R as an ID (in other words we don't use operation Gcdex) and compute an echelon form of A as considered over the fraction field of R. The algorithm of Section 5.2 is then applied in Section 5.3 to get an efficient algorithm for solving systems of linear diophantine equations over a PID such as $\mathbb{Q}[x]$.

We begin with some definitions. The lattice $\mathcal{L}(A)$ is the set of all R-linear combinations of rows of A. A notion of lattice for matrices over PIDs is analogous to that of vector spaces for matrices over fields. Some more discussion about lattices can be found in (Newman, 1972), but the presentation here is self contained. Let B over R have the same dimension as A. B is a (left) multiple of A if there exists an $M \in \mathsf{R}^{n\times n}$ such that $MA = B$. Note that M need not be unimodular or even

nonsingular. But it is easy to show that

Lemma 5.1. *If A and B are multiples of each other then $\mathcal{L}(A) = \mathcal{L}(B)$.*

In what follows we will write $A \cong B$ to mean that A and B have the same dimension and $\mathcal{L}(A) = \mathcal{L}(B)$. In other words, A and B are left equivalent to each other.

Lattice Determinant

Define $\det \mathcal{L}(A) \stackrel{\text{def}}{=} \prod_{1\le i\le r} H[1, j_i]$. Then $\mathcal{L}(A) = \mathsf{R}^m$ precisely when $r = m$ and $\det \mathcal{L}(A) = 1$. The following facts will also be useful.

Lemma 5.2. $\det \mathcal{L}(A)$ *is the gcd of all $r \times r$ minors of A comprised of columns $j_1, j_2, \ldots, j_r$.*

Lemma 5.3. *If B is a multiple of A, then $B \cong A$ if and only if* rank $B =$ rank A *and* $\det \mathcal{L}(B) = \det \mathcal{L}(A)$.

Extended Matrix GCD

If the rows of $G \in \mathsf{R}^{r\times m}$ provide a basis for the lattice of A (i.e. if $\mathcal{L}(G) = \mathcal{L}(A)$) then we simply say that G is a basis for A. For example, the nonzero rows of H are the canonical "Hermite basis" for A. Also:

Lemma 5.4. *The first r rows of any echelon form of A are a basis for A.*

A basis G for A is sometimes called a (right) matrix gcd.

Lemma 5.5. *If $r = m$, and G is a basis for A, then all entries in AG^{adj} are divisible by* $\det G$, *and* $\mathcal{L}(AG^{\mathsf{adj}}(1/\det G)) = \mathsf{R}^m$.

Consider for the moment that $n = 2m$, $r = m$, and A can be written as

$$\left[\begin{array}{c} A_1 \\ \hline A_2 \end{array}\right]$$

where both A_1 and A_2 are square nonsingular. Let $E \in \mathsf{R}^{r\times n}$ be such that $EA = G$. Then E is called a solution to the extended matrix gcd problem. If E_1 is the first m and E_2 the last m colums of E then $E_1A_1 + E_2A_2 = G$. We will use "extended matrix GCD" to mean, more generally, the problem of recovering the first r rows of a unimodular transforming matrix which transforms to echelon form an $A \in \mathsf{R}^{n\times m}$ with rank r.

Consider the scalar case of the extended matrix GCD problem, when $m = 1$. Then

$$\overset{E}{\begin{bmatrix} e_1 & e_2 & \cdots & e_n \end{bmatrix}} \overset{A}{\begin{bmatrix} a_1 \\ a_2 \\ \vdots \\ a_n \end{bmatrix}} = \overset{G}{\begin{bmatrix} g \end{bmatrix}}$$

with $g = \gcd(a_1, a_2, \ldots, a_n)$. Proposition 3.5 (transformation to echelon form) gives an $O(n)$ basic operations algorithm to recover the vecotr E. (It is already know that $O(n)$ operations suffice, see for example (Majewski and Havas, 1994).)

The analogy between a basis $G \in \mathsf{R}^{r\times m}$ and scalar gcd g is closest when $r = m$. But assume $r < m$ and we have the matrix $\bar{G} \in \mathsf{R}^{r\times r}$ comprised of only the r columns $j_1, j_2, \ldots, j_r$ of G. Then the other columns of G can be reconstructed from any r linearly independent rows in $\mathcal{L}(A)$. For example, in the next lemma B could be any r linearly independent rows of A.

Lemma 5.6. *Let $B \in \mathsf{R}^{r\times m}$ have rank r and satisfy $\mathcal{L}(B) \subseteq \mathcal{L}(A)$. Let $\bar{B} \in \mathsf{R}^{r\times r}$ be comprised of the rank profile columns of B. If $\bar{G} \in \mathsf{R}^{r\times r}$ is the submatrix comprised of the rank profile columns of a basis for A, then $(1/\det \bar{B})\bar{G}\bar{B}^{\mathsf{adj}}B$ is a basis for A.*

Many Hermite form algorithms require that A have full column rank (cf. Section 5.1). Lemma 5.6 shows that this is no essential difficulty. Compute a fraction free GaussJordan transform (U, P, r, d) for A and let B be the first r rows of UPA. Compute $\bar{G} \in \mathsf{R}^{r\times r}$ to be the Hermite basis of the full column rank submatrix comprised of the rank profile columns of A. Then a Hermite basis for A is given by $(1/d)\bar{G}B$.

Nullspaces

The (left) nullspace of A is the lattice $\{v \in \mathsf{R}^{1\times m} \mid vA = 0\}$. We say simply that a matrix $N \in \mathsf{R}^{(n-r)\times m}$ is a basis for the nullspace for A if $\mathcal{L}(N)$ equals the nullspace of A. Any basis for the nullspace of A necessarily has full row rank $n - r$.

Lemma 5.7. *Let $M \in \mathsf{R}^{(n-r)\times n}$ have rank $n - r$ and satisfy $MA = 0$. Then M is a basis for the nullspace of A if and only if $\mathcal{L}(M^T) = \mathsf{R}^{n-r}$.*

We get the following as a corollary to Lemma 5.5.

Corollary 5.8. *Let $M \in \mathsf{R}^{(n-r)\times n}$ have rank $n - r$ and satisfy $MA = 0$. Let $G^T \in \mathsf{R}^{(n-r)\times(n-r)}$ be a basis for M^T. Then $(1/\det G)G^{\mathsf{adj}}M$ is a nullspace for A.*

Corollary 5.5 says that we may deduce a basis for the nullspace of A given any full rank multiple of such a basis. The essential point is that such a multiple can be recovered by computing over the fraction field of R. For example, choose M to be the last $n - r$ rows of UP where $(U, P, r, *)$ is a GaussJordan transform for A.

Or consider the case when A is large and sparse. A suitable M can be constructed by computing random vectors in the left nullspace of A using iterative methods as proposed by (Kaltofen and Saunders, 1991). This should be efficient when $n - r$ is small compared to n. An alternative solution to this problem, that of construcing a nullspace for a sparse matrix when $n - r$ is small compared to n, is proposed and analysed by Squirrel (1999).

The Pasting Lemma

Let $G \in \mathsf{R}^{r\times m}$ be a basis for A. Consider matrices $U \in \mathsf{R}^{n\times n}$ which can be partitioned as and satisfy

$$\overset{U}{\left[\begin{array}{c} E \\ \hline \\ M \\ \\ \end{array}\right]} \overset{A}{\left[\begin{array}{c} * \\ \hline \\ * \\ \\ \end{array}\right]} = \left[\begin{array}{c} G \\ \hline \\ \\ \\ \end{array}\right] \tag{5.1}$$

where $EA = G$ and $MA = 0$.

Lemma 5.9. *The matrix U of (5.1) is unimodular if and only if M is a basis for the nullspace for A.*

We call Lemma 5.9 the "pasting lemma". Any $E \in \mathsf{R}^{r\times m}$ such that EA is a basis for A and any basis $M \in \mathsf{R}^{(n-r)\times n}$ for the nullspace of A can be pasted together to get a unimodular matrix.

5.1 Modular Computation of an Echelon Form

Assume that $A \in \mathsf{R}^{n\times m}$ has full column rank. Then an echelon basis for A (eg. the Hermite basis) can be recovered by computing modulo a carefully chosen $d \in \mathsf{R}$.

Lemma 5.10. *Let $A \in \mathsf{R}^{n\times m}$ have full column rank. If $\det \mathcal{L}(A) | d$ then*

$$\left[\begin{array}{c} A \\ \hline \\ \end{array}\right] \cong \left[\begin{array}{c} A \\ \hline dI \end{array}\right].$$

Let B over R have the same dimension as A. For $d \in \mathsf{R}$ nonzero we write $A \cong_d B$ if there exists a $U \in \mathsf{R}^{n\times n}$ with $\det U \perp d$ and $UA \equiv B \bmod d$. This condition on the determinant of U means that there also exists a $V \in \mathsf{R}^{n\times n}$ with $\det V \perp d$ and $VB \equiv A \bmod d$.

Lemma 5.11. *Let $A, B \in \mathsf{R}^{n\times m}$. If $A \cong_d B$ then* $\left[\begin{array}{c} A \\ \hline dI \end{array}\right] \cong \left[\begin{array}{c} B \\ \hline dI \end{array}\right]$.

The next two observations captures the essential subtlety involved in computing the Hermite form "modulo d".

Lemma 5.12. *Let $a, d, h \in \mathsf{R}$ be such that $a|d$ and h only has prime divisors which are also divisors of d. If $(d^2, h) = (a)$ then $(h) = (a)$.*

Lemma 5.13. *Let $a, d, h \in \mathsf{R}$ be such that $a|d$ and h only has prime divisors which are also divisors of d. If $(d, h) = (a)$ then we may not conclude that $(h) = (a)$. An example over $\mathbb{Z}$ is $a = 2$, $d = 2$ and $h = 4$.*

Recall that $\phi = \phi_{d^2}$ is used to denote the canonical homomorphism from R to $\mathsf{R}/(d^2)$. In what follows we assume the choices of $\mathcal{A}(\cdot)$ and $\mathcal{R}(\cdot,\cdot)$ over $\mathsf{R}/(d^2)$ are made consistently with the choices over R (see Section 1.1). This assumption is crucial.

Proposition 5.14. *Let $A \in \mathsf{R}^{n\times m}$ have full column rank and let $d \in \mathsf{R}$ satisfy $\det \mathcal{L}(A) | d$, $d \in \mathsf{R} \setminus \mathsf{R}^*$. If $\bar{H}$ is a Hermite form of $\phi_{d^2}(A)$ over $\mathsf{R}/(d^2)$, then $\phi^{-1}(\bar{H})$ is the Hermite form of A over R.*

Proof. Let $H = \phi^{-1}(\bar{H})$. Then H is in Hermite form over R. We need to show that H is the Hermite form of A. By Lemmas 5.10 and 5.11

$$\left[\begin{array}{c} A \\ \hline \\ \end{array}\right] \cong \left[\begin{array}{c} H \\ \hline d^2 I \end{array}\right]. \tag{5.2}$$

Thus H is left multiple of A. By Lemma 5.3 it will suffice to show that $\det \mathcal{L}(A) = \det \mathcal{L}(H)$.

From (5.2) and Lemma 5.2 we have that $\det \mathcal{L}(A)$ is equal to the gcd of all $m \times m$ minors of the matrix on the right hand side of (5.2). All such minors which involve a row from $d^2 I$ will be a multiple of d^2. We may deduce that H has rank m and that $(d^2, \det \mathcal{L}(H)) = (\det \mathcal{L}(A))$. By construction each diagonal entry of H is a divisor of d^2. The result now follows from Lemma 5.12. ☐

As a corollary to Proposition 5.14 and Corollary 3.16 (computation of Hermite form over $\mathbb{Z}/(d^2)$ with transforming matrix) we get the following.

Corollary 5.15. *Let $A \in \mathbb{Z}^{n\times n}$ be nonsingular and d be a positive multiple of $\det A$. Given d together with a matrix $B \in \mathbb{Z}^{n\times n}$ that satisfies $B \cong_d A$, the Hermite form H of A together with a $U \in \mathbb{Z}^{n\times n}$ such that $UB \equiv H \bmod d$ can be computed in $O(n^\theta(\log \alpha) + n^2(\log n)\,\mathrm{B}(\log \alpha))$ word operations where $\alpha = ||B|| + d$.*

Let d be as in Proposition 5.14. One quibble with the approach of Proposition 5.14 is that we work with the more expensive modulus d^2 instead of d. If we don't require a transforming matrix as in Corollary 5.15 the approach can be modified to allow working modulo d.

Our eventual goal is to transform A to echelon form. Say we have achieved

$$B = \left[\begin{array}{c|c} t & a \\ \hline & B' \end{array}\right]$$

where $t \in \mathsf{R}$ and $B \cong_d A$. For example, B could be obtained by applying unimodular row operations and arbitrarily reducing entries of the work matrix modulo d.

Compute $(h, s, v, *, *) := \mathrm{Gcdex}(t, d)$. Then

$$\left[\begin{array}{c|c|c|c} s & & v & \\ \hline & I & & \\ \hline d/h & & -t/h & \\ \hline & & & I \end{array}\right] \left[\begin{array}{c|c} t & a \\ \hline & B' \\ \hline d & \\ \hline & dI \end{array}\right] = \left[\begin{array}{c|c} h & sa \\ \hline & B' \\ \hline & (d/h)a \\ \hline & dI \end{array}\right] \tag{5.3}$$

where the transforming matrix on the left is unimodular by construction.

From Lemmas 5.10, 5.11 and equation (5.3) we get

$$\mathcal{L}(A) = \mathcal{L}\left(\left[\begin{array}{c|c} h & sa \\ \hline & B' \\ \hline & (d/h)a \\ \hline & dI \end{array}\right]\right). \tag{5.4}$$

The first row of the matrix on the right hand side must be the first row of an echelon form for A. To compute the remaining rows we may recursivley apply the steps just described (transformation of A to B) to the $(n+m-1)\times(m-1)$ submatrix

$$\left[\begin{array}{c} B' \\ \hline (d/h)a \\ \hline\hline dI \end{array}\right]. \tag{5.5}$$

We now describe an enhancement. From (5.4) we see that h is an associate of the first diagonal entry in the Hermite basis of A, and thus the lattice determinant of the submatrix (5.5) must be a divisor of d/h. Applying Lemma 5.10 gives

$$\left[\begin{array}{c} B' \\ \hline (d/h)a \\ \hline dI \end{array}\right] \cong \left[\begin{array}{c} B' \\ \hline (d/h)a \\ \hline (d/h)I \end{array}\right] \cong \left[\begin{array}{c} B' \\ \hline \\ \hline (d/h)I \end{array}\right].$$

Thus, we may neglect the middle row $(d/h)a$ of (5.5) and find the remaining rows in the echelon form by recursing on the $(n+m-2)\times(m-1)$ matrix

$$\left[\begin{array}{c} B' \\ \hline (d/h)I \end{array}\right].$$

The method just described has been called the "decreasing modulus" method by Domich *et al.* (1987). Alternatively, compute an echelon form B such that $B \cong_d A$. Then use the decreasing modulus approach just described to transform B to echelon form T which satisfies $T \cong A$; the reconstruction phase costs n basic operations of type Gcdex and $O(n^2)$ basic operations of type Arith.

Notes

Kannan and Bachem (1979) point out with an example that computing the canonical form of an integer matrix modulo the determinant d *does* *not* work. The subtleties are explained above. The key observation we make (in Proposition 5.14) is that computing mod d^2 *does* work. This is important since it allows us to recover a U such that $UB \equiv H \bmod d$ where $\det U \perp d$. This does not seem possible when working with the modulus d.

All of the theory described above (except for Proposition 5.14) for computing modulo d is exposed very nicely for integer matrices by Domich *et al.* (1987). The "mod d" approach is used also by Domich (1989), Iliopoulos (1989a, 1989b) and Hafner and McCurley (1991).

5.2 Fraction Free Computation of an Echelon Form

An echelon form of a full column rank A over R can be recovered from a modified Gauss transform of A. The idea is to combine fraction free Gaussian elimination with the decreasing modulus method discussed in the previous section. This is useful when working over rings such as $\mathbb{Q}[x]$ where expression swell is nasty. Using the approach described here all intermediate expressions will be minors of the input matrix.

Recall Subroutine 2.20 (`Cond`), used by Algorithm 2.18 (`CondGauss`). We need to modify Subroutine 2.20 (`Cond`) slightly. Given a nonzero $A = \left[\begin{array}{ccc} a_1 & a_2 \cdots & a_n \end{array}\right]^T \in \mathsf{R}^{n\times 1}$ the matrix C returned by `Cond` should satisfy

$$(a_1, a_2, \ldots, a_n, d^2) = (a_1 + c_2a_2 + \cdots + c_na_n, d^2). \tag{5.6}$$

When $\mathsf{R} = \mathbb{Q}[x]$ there exist choices for the c_i's which are nonnegative integers bounded in magnitude by $1 + \deg d$. When $\mathsf{R} = \mathbb{Z}$ there exist choices for the c_i's which are nonnegative integers with $c_i = O((\log d)^2)$. Algorithm 5.16 (FFEchelon) requires such small choices for the c_i's. Efficient deterministic algorithms for computing minimal norm solutions over $\mathbb{Z}$ and $\mathsf{K}[x]$ are given in (Storjohann, 1997) and (Mulders and Storjohann, 1998).

One thing we will need to show is that the division in step 3 are exact. To prove the algorithm correct we need the following observation.

Lemma 5.17. *Let $A, B \in \mathsf{R}^{n\times m}$ and $h \in \mathsf{R}$ be nonzero. Then $hA \cong hB$ if and only if $A \cong B$.*

For convenience let us make some observations. Consider the case when A is square nonsingular. Then:

Algorithm 5.16. `FFEchelon`(A)
Input: $A \in \mathsf{R}^{n\times m}$ with rank r.
Output: (E,T), T an echelon basis T of A and $E \in \mathsf{R}^{r\times n}$ such that $EA = T$.

1. Compute a fraction free GaussJordan transform $(\bar{U}, P, r, d)$ of A. Let U_1 be the first r rows of $\bar{U}$ and and T_1 the first r rows of $\bar{U}PA$. Let $(j_1, j_2, \ldots, j_r)$ be the rank profile of A.

2. Compute a modified fraction free Gauss transform $(U, C, *, *)$ of PA using a suitable subroutine `Cond` which produces output satisfying 5.6. Let U_2 be the first r rows of U and T_2 be the first r rows of $UCPA$.

3. Let $S_1, S_2 \in \mathsf{R}^{n\times n}$ be diagonal with $(h_i^*, S_1[i,i], S_2[i,i], *, *) :=$ Gcdex$(d^2, T_1[i, j_i])$. Let $D \in \mathsf{R}^{n\times n}$ be diagonal with $D[i,i] := h_{i-1}^*$, $h_0^* = 1$. Set $E := D^{-1}(S_1 dU_1 + S_2U_2C)P$ and $T := D^{-1}(S_1 dT_1 + S_2T_2)$. Return (E,T).

- d is the determinant of PA.
- $U = U_1$ is the adjoint of PA and $T_1 = dI$.
- $U = U_2$ is the triangularizing adjoint and T_2 the adjoint triangularization of CPA.
- $T[i,i] = h_i^*/h_{i-1}^*$ for $1 \le i \le n$.
- All entries in row i of $(S_1dU_1 + S_2U_2C)P$ will be divisible by the product of the first i diagonal entries of any echelon basis of A.

Now consider when A has rank profile $j_1, j_2, \ldots, j_r$. On input with the full column rank submatrix of A comprised of column $j_1, j_2, \ldots, j_r$ algorithm `FFEchelon` will produce exactly the same E.

Theorem 5.18. *Algorithm 5.16 is correct.*

Proof. We may assume without loss of generality that A has full column rank. It is easy to verify that $EA = T$ and $T[i,i] = h_i^*/h_{i-1}^*$. The output will be correct if and only if h_i^* is the product of the first i diagonal entries in some echelon basis of A. In this case we have that E is over R, and thus T is a multiple A and the result follows from Lemma 5.3.

Note that (U, I) is a Gauss transform of CPA. Consider triangularizing CPA column by column using fraction free Gaussian elimination. After the first k columns have been eliminated the work matrix can be written as

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

where the principal block is the principal $k \times k$ submatrix of T_2 and the trailing block B has dimension $(n-k)\times(m-k)$. Note that all information required to construct h_i^* for $1 \le i \le k$ is now determined. The construction of h_i^* for $i > k$ depends essentially on the subsequent triangularization of B. Now use induction on k to show that the h_i^*'s are correct.
Induction Hypothesis: There are two.
(a) $T_2[i,i] = c_i h_i^*$ where $c_i \perp d$ and h_i^* is the product of the first i entries in some echelon form of A, $1 \le i \le k$.
(b) The trailing $(n-k)\times(m-k)$ submatrix of an echelon form for A can be found by transforming

$$\left[\begin{array}{c} B \\ \hline\hline d^2 I \end{array}\right] \tag{5.7}$$

to echelon form and dividing by h_k^*.
Base Case: When $k = 0$ (a) is vacuous and (b) follows from Lemma 5.10.
Induction Step: By construction the principal entry of B is $T_2[k+1, k+1]$ and $h_{k+1}^* = \text{Gcd}(d^2, T_2[k+1, k+1])$. Because of the preconditioning with C we have h_{k+1}^* equal to the gcd of all entries in the first column of B. Using (b) we conclude that h_{k+1}^* is the product of the first $k+1$ entries in an echelon form of A. Let $T_2[k+1, k+1] = c_{k+1}h_{k+1}^*$. Then $(d^2/h_{k+1}^*, c_{k+1}) = (1)$. Since $h_{k+1}^* \mid d$ we may conclude that $c_{k+1} \perp d$. This gives the inductive step for (a).

Using (a) we also have that $T_2[k,k] = c_k h_k^*$ with $c_k \perp d$. Now we are going to combine fraction free Gaussian elimination of B with the decreasing modulus approach of the previous section. Let $h_{k+1}^* = h_{k+1}h_k^*$. The principal entry of $\frac{1}{h_k^*}B$ is $c_{k+1}h_{k+1}$. Consider applying the following sequence of row operations to $\frac{1}{h_k^*}B$. Multiply all but the first row by c_{k+1}. Then add appropriate multiples of the first row to the lower rows to zero out entries below the principal entry. Now multiply all rows but the first row by $1/c_k$. Since c_{k+1} and c_k are $\perp d^2/h_k^*$ we have (Lemma 5.11) that

$$\left[\begin{array}{c} \frac{1}{h_k^*}B \\ \hline \frac{1}{h_k^*}d^2 I \end{array}\right] \cong \left[\begin{array}{c|c} c_{k+1}h_{k+1} & \frac{1}{h_k^*}a \\ \hline & \frac{1}{h_k^*}B' \\ \hline\hline \frac{1}{h_k^*}d^2 & \\ \hline & \frac{1}{h_k^*}d^2 I \end{array}\right].$$

Now proceed as in (5.3) using the decreasing modulus approach. This gives that the remaining $n-k-1$ rows of an echelon form for A can be found by transforming

$$\left[\begin{array}{c} B' \\ \hline\hline (d^2/h_{k+1})I \end{array}\right]$$

to echelon form and dividing by h_k^*. Multiply by h_{k+1}. By Lemma 5.17 the trailng $(n-k-1)\times(m-k-1)$ submatrix of an echelon form for A can be found by transforming

$$\left[\begin{array}{c} (h_{k+1}/c_k)B' \\ \hline\hline d^2 I \end{array}\right]$$

to echelon form and dividing by h_{k+1}^*. The key observation is now that the principal block of this matrix is precisely what we would have obtained by continuing fraction free Gaussian elimination for one column on B. This gives the inductive step for (b). □

5.3 Solving Systems of Linear Diophantine Equations

Given $b \in \mathsf{R}^m$, the problem of solving a linear diophantine system is to find a vector $v \in \mathsf{R}^n$ such that $vA = b$ or prove that no such vector exists. This problem can be generalized somewhat using the following observation.

Lemma 5.19. *The set of all $c \in \mathsf{R}$ such that $vA = cb$ admits a solution for v is an ideal of R.*

In (Mulders and Storjohann, 1999) we call a solution $\frac{1}{c}v$ which solves $\frac{1}{c}vA = b$ in the sense of Lemma 5.19 a *solution with minimal denominator*. It is well known how to find such a solution by transforming the $(n+1)\times(m+1)$ matrix

$$B = \left[\begin{array}{c|c} A & \\ \hline -b & 1 \end{array}\right].$$

to echelon form, see for example Blankinship (1966b). Kannan (1985) first showed the problem of solving linear systems over $\mathbb{Q}[x]$ was in $\mathcal{P}$ using this approach. Here we observe that we can apply algorithm `FFEchelon` to the problem at hand.

Proceed as follows. Let U_1, U_2, T_1, T_2, d, r be as produced by steps 1 and 2 of algorithm `FFEchelon` with input A. (But don't perform step 3.) Then the system will be consistent only if the last row of T_2 has first m entries zero. If this is the case then compute $(c, s, t, *, *) := \mathrm{Gcd}(d^2, T_2[r+1, j_{r+1})$ and construct the last row of E as in step 3. Thus only a single operation of type Gcdex is required.

Let us estimate the complexity of this approach when $\mathsf{R} = \mathbb{Q}[x]$. Assume A and b are over $\mathbb{Z}[x]$ and let d be a bound on the degrees of polynomial coefficients and α a bound on the magnitudes of integer coefficients in A and b. For simplicity consider the diagonal case: Let $s = n + m + d + \log\alpha$.

The dominant cost will almost certainly be the fraction free Gaussian elimination in steps 1 and 2. Not counting the calls to subroutine `Cond` this is bounded by $O\tilde{~}(s^{4+\theta})$ word operations (assuming FFT-based integer arithmetic). This estimate is obtained by noting that polynomials over $\mathbb{Z}[x]$ can be represented as integers by writing the coefficients as a binary lineup.

We show in (Mulders and Storjohann, 1998) that the cost of all calls to `Cond` will be equivalent to computing computing $O(s^2)$ gcds of polynomials over $\mathbb{Z}[x]$ which have both degrees and magnitude integer coefficients bounded by $O(s^2)$. If we allow randomization (Las Vegas) the gcd computations can be accomplished using the algorithm of Schönhage (1988) in the time bound stated above for steps 1 and 2. The derivation of a good worst case deterministic complexity bounds for these gcd computations is more challenging.

Let us compare the approach described above using `FFEchelon` with the usual method which is to compute a solution to the matrix extended gcd problem, that is, compute a complete echelon form T of B together with the first r rows E of a unimodular transform matrix. The total size of such an E and T will be $O\tilde{~}(s^8)$ words; this bound is probably tight. A detailed discussion is given in (Storjohann 1994, Chapter 4). An E and T can be recovered in $O\tilde{~}(s^{6+\theta})$ bit operations (assuming FFT-based integer arithmetic) by combining (Storjohann, 1994, Theorem 6.2) with Proposition 3.15. This complexity for producing a solution to the matrix extended gcd problem is close to optimal, but considerably more than the $O\tilde{~}(n^{4+\theta})$ bound derived for the alternative method sketched above.

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.

Let $A \in \mathbb{Z}^{n\times m}$ have rank r. Let G be the first r rows of the Hermite form of A — the *Hermite basis* of A. Consider matrices $U \in \mathbb{Z}^{n\times n}$ with the property that UA equals the Hermite form of A. Any such U can be partitioned using a conformal block decomposition as

$$\overset{U}{\left[\begin{array}{c} E \\ \hline \\ M \\ \\ \end{array}\right]} \overset{A}{\left[\begin{array}{c} * \\ \hline \\ * \\ \\ \end{array}\right]} = \left[\begin{array}{c} G \\ \hline \\ \\ \\ \end{array}\right] \tag{6.1}$$

where $EA = G$. Such a matrix U will be unimodular precisely when M is a basis for the nullspace for A. Let $\beta = (\sqrt{r}||A||)^r$. We show how to recover G together with an E in $O(nmr^{\theta-2}(\log\beta) + nm(\log r)\,\mathrm{B}(\log\beta))$

word operations. A nullspace M can be recovered in $O(nmr^{\theta-2}(\log 2n/r)(\log\beta) + nm(\log n)\,\mathrm{B}(\log\beta))$ word operations.

The main contribution here is to produce an E and M (in the time stated above) with good size bounds on the entries. We get $||E|| \leq \beta$ and $||M|| \leq r\beta^2$. Furthermore, E will have at most $r + (r/2)\log_2 r + r\log_2 ||A||$ nonzero columns. We also show that $||G|| \leq \beta$.

Preliminaries

The bounds claimed in the next lemma are easily verified.

Lemma 6.1. *Let $A \in \mathbb{Z}^{n\times n}$ be nonsingular with Hermite form H. Then $\det H = |\det A|$ and the unique U such that $UA = H$ is given by $(1/\det A)HA^{\mathsf{adj}}$. Moreover:*

(a) $\max_i\{\sum_j H_{ij}\} \leq \det H$.

(b) $||H^{\mathsf{adj}}|| \leq n^{n/2}\det H$.

(c) $||U|| \leq ||A^{\mathsf{adj}}||$.

Notes

The form origninates with Hermite (1851). Early algorithms for triangularizing integer matrices given by Barnette and Pace (1974), Blankinship (1966a), Bodewig (1956), Bradley (1971) and Hu (1969) are not known to admit polynomial time running bounds — the main problem being the potential for rapid growth in the size of intermediate integer coefficients. Fang and Havas (1997) prove that a well defined variant of the standard elimination algorithm — an example of such an algorithm was given in the notes section of Chapter 3 — leads to exponential growth when $\mathsf{R} = \mathbb{Z}$. A doubly exponential lower bound on the size (or norm) of operands over certain Euclidean rings is demonstrated by Havas and Wagner (1998).

Table 6.1 summarizes polynomial time complexity results for the case of a nonsingular $n \times n$ input matrix A. The Time and Space columns give the exponents e_1 and f_1 such that the corresponding algorithm has running time bounded by $O\tilde{~}(n^{e_1}(\log ||A||)^{e_2})$ bit operatons and intermediate space requirments bounded by $O\tilde{~}(n^{f_1}(\log ||A||)^{f_2})$ bits. We neglect to give the exponents e_2 and f_2 (but remark that they are small for all the algorithms, say on the order of 1.) We use this simplified

(Time, Space) "one-paramter" complexity model only when summarizing asymptotic complexity — our primary interest is the complexity in the paramater n. Kannan and Bachem (1979) give the first (proven)

	Citation	Time	Space
	Hermite form of dense square integer matrix		
1	Kannan and Bachem (1979)	finite	finite
2	Chou and Collins (1982)	6	3
3	Domich (1985)	4	3
4	Domich *et al.* (1987)	4	3
5	Iliopoulos (1989a)	4	3
6	Hafner and McCurley (1991)	4	3
7	Storjohann and Labahn (1996)	$\theta + 1$	3
8	Storjohann (200x)	4	2

Table 6.1: Complexity bounds for Hermite form computation

polynomial time algorithm. This is later improved by Chou and Collins (1982), with an emphasis on the problem on solving a linear diophantine system. Algorithms 1 and 2 perform the elimination directly over $\mathbb{Z}$ — most of the effort is spent bounding the growth of entries. We sidestep the problem of expression swell by working modulo the determinant of a square nonsingular matrix; this is the technique used in Algorithms 3–7. The Time bounds given for algorithms 3–6 assume $\mathrm{M}(k) = O\tilde{~}(k)$ while that for algorithm 7 assumes $\mathrm{M}(k) = O\tilde{~}(k^{\theta-1})$. Algorithm 8 assumes $\mathrm{M}(k) = k^2$.

Although some of the algorithms cited in Table 6.1 are presented for the general case, some assume the matrix has full column rank or even that the input matrix is nonsingular. This is no essential difficulty, since any algorithm for transforming a nonsingular matrix to Hermite form may be adapted to handle the case of a rectangular input matrix and, moreover, to recover also a transforming matrix. This is dealt with also by Wagner (1998). We mention one method here. Let $A \in \mathbb{Z}^{n\times r}$ have rank r. Recover the rank profile $[j_1, j_2, \ldots, j_r]$ of A together with a permutation matrix P such that PA has first r rows linearly independant. Let $\bar{A}$ be the matrix obtained by augmenting columns $[j_1, \ldots, j_r]$ of PA with the last $n - r$ columns of I_n. Then $\bar{A}$ is square nonsingular. Compute the Hermite form $\bar{H}$ of $\bar{A}$. Set $U = (1/\det A)\bar{H}\bar{A}^{\mathsf{adj}}P^{-1}$. Then U is unimodular and $H = UA$ is the Hermite form of A.

The method just sketched, which is mentioned also by Hafner and

McCurley (1991), is fine when $n \leq m$, but does not lead to good running time bounds when n is significantly larger than m, cf. Figure 1.1 on page 7. Table 6.2 summarizes results for this case. We assume that the

	Citation	Time	$\log_2 \|\|U\|\|$
	Transform for rectangular integer matrix		
1	Hafner and McCurley (1991)	n^3	$m \log_2(\sqrt{m}\|\|A\|\|)$
2	Storjohann and Labahn (1996)	nm^θ	$O((\log 2n/m)m(\log m\|\|A\|\|))$
3	Storjohann (here)	nm^θ	$\log_2 m + 2m \log_2(\sqrt{m}\|\|A\|\|)$

Table 6.2: Soft-Oh complexity bounds for transform computation

input matrix has full column rank m. The first citation is the method sketched above. Algorithm 2 works by adapting Hafner and McCurley's (1991) algorithm for recovering a transform matrix over an abstract ring to the case $\mathbb{Z}$. Finally, the third citation is the asymptotically fast method of this chapter. For Methods 1 and 3 we get good explicit bounds on the bit-length of entries. In (Storjohann and Labahn, 1996) we gave an asymptotic bound for the bit-length $\log_2 ||U||$.

Our primary goal here is to obtain a good worst case asymptotic complexity bound under the tri-parameter model for producing a transform. Our secondary goal is to get good explicit bounds on the sizes of entries in the transform. We have chosen to make this secondary goal subordinate to the first, but we remark that because the transform is highly non-unique, there are many different directions that research can take.

Different approaches include heuristic methods which attempt to achieve much better bounds on the sizes of both intermediate numbers and those appearing in the final output. See for example Havas and Majewski (1994). Moreover, since the last $n-r$ rows of U are a nullspace basis, these can be reduced using integer lattice basis reduction. This will typically result in much smaller entries, but at the price of an increased running time bound. See the discussion in the books by Sims (1984) and Cohen (1996). More recently, Havas *et al.* (1998) show that the Hermite form can be deduced by reducing, via lattice basis reduction, a well chosen integer lattice.

6.1 Extended Matrix GCD

Let $A \in \mathbb{Z}^{n\times n}$ be nonsingular with Hermite form H. Assume A and H can be written using a conformal block decomposition as

$$A = \left[\begin{array}{c|c} B & \\ \hline D & I_{n-r} \end{array}\right] \quad \text{and} \quad H = \left[\begin{array}{c|c} H_1 & H_3 \\ \hline & H_2 \end{array}\right] \tag{6.2}$$

for some r, $1 \leq r \leq n$. Let E be the first r and M the last $n-r$ rows of the unique U which satisfies $UA = H$. Our goal in this section is to produce E. Solving for U gives

$$U = \left[\begin{array}{c} E \\ \hline M \end{array}\right] = \left[\begin{array}{c|c} (1/\det B)(H_1 - H_3 D)B^{\mathsf{adj}} & H_3 \\ \hline -(1/\det B)H_2 D B^{\mathsf{adj}} & H_2 \end{array}\right]. \tag{6.3}$$

Note that entries in A^{adj} will be minors of A bounded in dimension by r. This gives the bound $||U|| \leq (\sqrt{r}||A||)^r$ using Lemma 6.1c. We are going to construct E using (6.3). The main step is to recover H_1 and H_3.

The key result of this chapter is:

Lemma 6.2. *Let $A \in \mathbb{Z}^{n\times n}$ be as in (6.2). Given $\det B$, the matrices H_1 and H_3 can be recovered in $O(nr^{\theta-1}(\log\beta) + nr(\log r)\,\mathrm{B}(\log\beta))$ word operations where $\beta = ||A|| + |\det B|$.*

Proof. By extending D with at most $r-1$ rows of zeroes, we may assume without loss of generality that n is a multiple of r, say $n = kr$. All indexed variables occuring henceforth will be $r \times r$ integer matrices. Write A as

$$A = \begin{bmatrix} B & & & & \\ D_2 & I & & & \\ D_3 & & I & & \\ \vdots & & & \ddots & \\ D_k & & & & I \end{bmatrix}.$$

We first give the algorithm and then bound the cost later. Let G_k denote B. For $i = k, k-1, \ldots, 2$ in succesion, compute $(S_i, T_i, V_i, H_i, B_i)$ such that

$$\overset{U_i}{\left[\begin{array}{c|c} S_i & T_i \\ \hline V_i & H_i \end{array}\right]} \left[\begin{array}{c|c} G_i & \\ \hline D_i & I \end{array}\right] = \left[\begin{array}{c|c} G_{i-1} & T_i \\ \hline & H_i \end{array}\right]$$

with U_i unimodular and the right hand side in Hermite form. The principal block G_1 of the last Hermite form computed will be the Hermite basis of A.

Expand each U_i with identity matrices to get

$$\overset{U_2}{\begin{bmatrix} S_2 & T_2 & & & \\ V_2 & H_2 & & & \\ & & I & & \\ & & & \ddots & \\ & & & & I \end{bmatrix}} \overset{U_3}{\begin{bmatrix} S_3 & & T_3 & & \\ & I & & & \\ V_3 & & H_3 & & \\ & & & \ddots & \\ & & & & I \end{bmatrix}} \cdots \overset{U_k}{\begin{bmatrix} S_k & & & & T_k \\ & I & & & \\ & & I & & \\ & & & \ddots & \\ V_k & & & & H_k \end{bmatrix}} \overset{A}{\begin{bmatrix} B & & & & \\ D_2 & I & & & \\ D_3 & & I & & \\ \vdots & & & \ddots & \\ D_k & & & & I \end{bmatrix}}$$

$$= \overset{T}{\begin{bmatrix} G_1 & * & * & \cdots & * \\ & H_2 & * & \cdots & * \\ & & H_3 & \cdots & * \\ & & & \ddots & \vdots \\ & & & & H_k \end{bmatrix}}$$

where T is a unimodular triangularization of A. For any block $*$ there exists a unique solution (R_i, E_i) to

$$\left[\begin{array}{c|c} I & R_i \\ \hline & H_i \end{array}\right] \left[\begin{array}{c|c} I & * \\ \hline & H_i \end{array}\right] = \left[\begin{array}{c|c} I & E_i \\ \hline & H_i \end{array}\right]$$

such that the matrix on the right is in Hermite form. We are going to compute (R_i, E_i) in succession for $i = 1, 2, \ldots, k$ such that

$$\overset{F_k}{\begin{bmatrix} I & & & & R_k \\ & I & & & \\ & & I & & \\ & & & \ddots & \\ & & & & I \end{bmatrix}} \cdots \overset{F_3}{\begin{bmatrix} I & & R_3 & & \\ & I & & & \\ & & I & & \\ & & & \ddots & \\ & & & & I \end{bmatrix}} \overset{F_2}{\begin{bmatrix} I & R_2 & & & \\ & I & & & \\ & & I & & \\ & & & \ddots & \\ & & & & I \end{bmatrix}} \overset{T}{\begin{bmatrix} G_1 & * & * & \cdots & * \\ & H_2 & * & \cdots & * \\ & & H_3 & \cdots & * \\ & & & \ddots & \vdots \\ & & & & H_k \end{bmatrix}}$$

$$= \begin{bmatrix} G_1 & E_2 & E_3 & \cdots & E_k \\ & H_2 & * & \cdots & * \\ & & H_3 & \cdots & * \\ & & & \ddots & \vdots \\ & & & & H_k \end{bmatrix},$$

with each E_i reduced with respect to H_i. Note that we cannot produce all off-diagonal entries of T explicitly in the allotted time. To recover

(R_i, E_i) we need to pipeline the computation. Proceed as follows. Initialize S to be the $r \times r$ identity. Compute

$$\left[\begin{array}{c|c} I & R_i \\ \hline & I \end{array}\right] \left[\begin{array}{c|c} I & ST_i \\ \hline & H_i \end{array}\right] = \left[\begin{array}{c|c} I & E_i \\ \hline & H_i \end{array}\right] \quad \text{and update} \quad S = S_i S + R_i V_i$$

for $i = 2, \ldots, k$. Induction on k (base case $k = 2$) shows that the first r rows of $F_k \cdots F_3 F_2 U_2 U_3 \cdots U_k A$ is equal to $\begin{bmatrix} G_1 & E_2 & E_3 & \cdots & E_k \end{bmatrix}$.

The cost estimate follows from Corollary 5.15. (Note that it is sufficient that the matrix equalities given above hold modulo $2|\det B|$.) $\square$

Proposition 6.3. *Let $A \in \mathbb{Z}^{n\times m}$ have rank r. An $E \in \mathbb{Z}^{r\times n}$ such that EA equals the Hermite basis of A can be recovered in $O(nr^{\theta-1}(\log\beta) + nr(\log r)\,\mathrm{B}(\log\beta))$ word operations where $\beta = (\sqrt{r}||A||)^r$. At most $r + \lfloor \log_2 \beta \rfloor$ columns of E will be nonzero and $||E|| \le \beta$.*

Proof. Extract those columns from A which don't belong to the rank profile of A. Premultiply A by a permutation matrix so that the principal $r \times r$ submatrix is nonsingular. (Later postmultiply the computed E by this permutation matrix.) The augmented input matrix can now be written as in (6.2). Recover B^{adj} and $\det B$. Everything so far is accomplished in one step by computing a fraction free GaussJordan transform of A.

Now recover H_1 and H_3 using Lemma 6.2. Finally, contruct the principal $r \times r$ block of E as in (6.3) using matrix multiplication. $\square$

Lemma 6.4. *Let G be the Hermite basis of $A \in \mathbb{Z}^{n\times m}$. Then $||G|| \le (\sqrt{r}||A||)^r$ where r is the rank of A.*

Proof. Let B be the first r and D the last $n - r$ rows of A. Without loss of generality assume that B has rank r. Let $\bar{A}$, $\bar{B}$, $\bar{D}$ and $\bar{G}$ be the submatrices of A, B, D and G respectively comprised of the columns corresponding to the rank profile of A. Considering (6.3) we deduce that $G = (1/\det \bar{B})\bar{H}\bar{B}^{\mathsf{adj}}B$. The key observation is now that entries in $\bar{B}^{\mathsf{adj}}B$ are $r \times r$ minors of B, and thus bounded in magnitude by β. The result now follows from Lemma 6.1a. $\square$

We end this section by giving an application of Proposition 6.3 when $n = r = 1$.

Corollary 6.5. *Let $A \in \mathbb{Z}^{n\times 1}$ be nonzero. An $E \in \mathbb{Z}^{1\times n}$ with $EA = \begin{bmatrix} g \end{bmatrix}$, g the gcd of entries in A, can be recovered in $O(n\,\mathrm{B}(\log ||A||))$ bit operations. At most $1 + \log_2 ||A||$ entries in E will be nonzero and $||E|| \le ||A||$.*

6.2 Computing a Null Space

Let $A \in \mathbb{Z}^{n\times r}$ with rank r be given. We are going to construct a nullspace for A in a novel way. Assume that $n > r$ and let $n = n_1 + n_2$ where $n_1 > r$. Assume we can partition A as

$$A = \left[\begin{array}{c} A_1 \\ \hline A_2 \end{array}\right] = \left[\begin{array}{c} A_{11} \\ \hline A_{12} \\ \hline A_{21} \\ \hline A_{22} \end{array}\right] \in \mathbb{Z}^{(n_1+n_2)\times r} \tag{6.4}$$

where A_1 is $n_1 \times r$ with rank r and A_2 is $n_2 \times r$ with rank r_2. Assume further that A_1 and A_2 can be partitioned as shown where A_{11} is $r \times r$ nonsingular and A_{21} is $r_2 \times r$ with rank r_2.

Let $M_1 \in \mathbb{Z}^{(n_1-r)\times n_1}$ and $M_2 \in \mathbb{Z}^{(n_2-r_2)\times n_2}$ be nullspaces for A_1 and A_2 respectively. Partition M_1 and M_2 as as $M_1 = \left[\begin{array}{c|c} M_{11} & M_{12} \end{array}\right]$ and $M_2 = \left[\begin{array}{c|c} M_{21} & M_{22} \end{array}\right]$ where M_{11} is $(n_1 - r)\times r$ and M_{21} is $(n_2 - r_2)\times r_2$. Then M_i can be completed to a unimodular matrix U_i such that

$$\overbrace{\left[\begin{array}{c|c} * & * \\ \hline M_{i1} & M_{i2} \end{array}\right]}^{U_i} \left[\begin{array}{c} A_{i1} \\ \hline A_{i2} \end{array}\right] = \left[\begin{array}{c} * \\ \hline \end{array}\right].$$

Now let $\bar{A}_2 \in \mathbb{Z}^{n_2\times r_2}$ be comprised of those columns of A_2 corresponding to the rank profile of A_2. Embedding U_1 and U_2 into an $n \times n$

matrix yields

$$\left[\begin{array}{c|c||c|c} * & & * & \\ \hline & * & & * \\ \hline\hline M_{11} & & M_{12} & \\ \hline & M_{21} & & M_{22} \end{array}\right] \overbrace{\left[\begin{array}{c|c} A_{11} & \\ \hline A_{21} & \bar{A}_{21} \\ \hline\hline A_{12} & \\ \hline A_{22} & \bar{A}_{22} \end{array}\right]}^{\bar{A}} = \left[\begin{array}{c|c} * & \\ \hline * & * \\ \hline\hline & \\ \hline & \end{array}\right] \tag{6.5}$$

where the transforming matrix is unimodular by construction. It follows that the trailing $n - (r + r_2)$ rows of this transforming matrix comprise a nullspace for $\bar{A}$. By Lemma 5.9, the transforming matrix will remain unimodular if we replace the first $r + r_2$ rows with any $E \in \mathbb{Z}^{(r+r_2)\times n}$ such that EA equals the Hermite basis of $\bar{A}$. Partition such an E as

$$E = \left[\begin{array}{c|c||c|c} E_{11} & E_{12} & E_{13} & E_{14} \\ \hline E_{21} & E_{22} & E_{23} & E_{24} \end{array}\right].$$

Then

$$\overbrace{\left[\begin{array}{c|c||c|c} E_{11} & E_{12} & E_{13} & E_{14} \\ \hline E_{21} & E_{22} & E_{23} & E_{24} \\ \hline\hline M_{11} & & M_{12} & \\ \hline & M_{21} & & M_{22} \end{array}\right]}^{U} \overbrace{\left[\begin{array}{c|c} A_{11} & \\ \hline A_{21} & \bar{A}_{21} \\ \hline\hline A_{12} & \\ \hline A_{22} & \bar{A}_{22} \end{array}\right]}^{\bar{A}} = \left[\begin{array}{c|c} H & * \\ \hline & * \\ \hline\hline & \\ \hline & \end{array}\right]$$

where U is unimodular and the right hand side is in Hermite form with H the Hermite basis for A. We conclude that

$$M = \left[\begin{array}{c|c|c|c} E_{21} & E_{23} & E_{22} & E_{24} \\ \hline M_{11} & M_{12} & & \\ \hline & & M_{21} & M_{22} \end{array}\right] \tag{6.6}$$

is a nullspace for A.

Proposition 6.6. *Let $A \in \mathbb{Z}^{n\times *}$ have column dimension bounded by m and rank $\bar{r}$ bounded by r. Let $\beta = (\sqrt{r}||A||)^r$. A nullspace $M \in \mathbb{Z}^{(n-\bar{r})\times n}$ for A which satisfies $||M|| \leq r\beta^2$ can be recovered in $O(nmr^{\theta-2}(\log 2n/r)(\log\beta) + nm(\log n)\,\mathrm{B}(\log\beta))$ word operations.*

Proof. Let A be as in the statement of the proposition. We first do some preprocessing. Recover the row rank profile of A and extract from A those columns which are linearly dependent; we assume henceforth that A has full column rank $\bar{r}$. Identify $\bar{r}$ linearly independent rows and premultiply A by a suitable permutation matrix so that the principal $\bar{r}\times\bar{r}$ submatrix of A is nonsingular. (Later postmultiply the computed nullspace by this permutation.) The steps so far are accomplished in the allotted time by computing a fraction free Gauss transform of A. Because of this preconditioning, it will be sufficient to prove the proposition for those cases when $m = \bar{r} \leq r$.

Let $T_r(n)$ be the number of bit operations required to compute a nullspace M which satisfies the conditions of the proposition for an $A \in \mathbb{Z}^{n\times\bar{r}}$ with rank $\bar{r}$, $\bar{r} \leq r$.

The algorithm is recursive. The base case occurs when $n < 2r$; assume this is the case. Extend A with $I_{n-\bar{r}}$ to get an $n\times n$ nonsingular matrix

$$B = \left[\begin{array}{c||c} A & \\ * & \\ \hline * & I_{n-\bar{r}} \end{array}\right]. \tag{6.7}$$

Use Lemma 6.2 to compute an $E \in \mathbb{Z}^{n\times n}$ such that EB is in Hermite form. Then $||M|| \leq \beta$ (Lemma 6.1c). Set M to be the last $n-r$ rows of E. This shows that

$$T_r(n) = O(r^\theta(\log\beta) + r^2(\log r)\,\mathrm{B}(\log\beta))) \quad \text{if} \quad n < 2r.$$

Now assume that $n \geq 2r$. The result will follow if we show that

$$T_r(n) = T_r(\lfloor n/2\rfloor) + T_r(\lceil n/2\rceil) + O(nr^{\theta-1}(\log\beta) + nr\,\mathrm{B}(\log\beta)) \quad \text{if} \quad n \geq 2r.$$

Let $n_1 = \lfloor n/2\rfloor$ and $n_2 = \lceil n/2\rceil$. Let A_1 be the first n_1 and A_2 the last n_2 rows of A. Let r_2 be the rank of A_2. Compute the column rank profile of A_2 to identify r_2 linearly independent rows. Premultiply A by a suitable permutation so that the principal $r_2\times r_1$ submatrix of A_2 is nonsingular. The input matrix A can now be written as in (6.4) and satisfies the rank conditions stated there.

Recursively compute nullspaces $M_1 \in \mathbb{Z}^{(n_1-\bar{r})\times n_1}$ and $M_2 \in \mathbb{Z}^{(n_2-r_2)\times n_2}$ which satisfy the requirements of the proposition with respect to A_1 and A_2. Construct the $n\times(\bar{r}+r_2)$ matrix $\bar{A}$ shown in (6.5). Using Proposition 6.3, compute an $E \in \mathbb{Z}^{(\bar{r}+r_2)\times n}$ such that $E\bar{A}$ equals the Hermite basis for $\bar{A}$. Finally, construct the nullspace for A as in (6.6).

Let B be the $n\times n$ nonsingular extended matrix as in (6.7). Lemma 6.1c bounds $||E||$ by $||B^{\mathrm{adj}}||$. It is straightforward to derive the bound $||B^{\mathrm{adj}}|| \leq r\beta^2$ by considering the two cases $r_2 = \bar{r}$ and $r_2 < \bar{r}$. We don't belabor the details here. $\square$

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 $1 \le i \le r-1$. 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

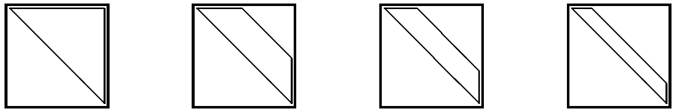


Figure 7.1: Banded Reduction

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

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

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 $j \ge i + b$, that is, if A can be written as

$$A = \begin{bmatrix} * & \cdots & * & & & & \\ & \ddots & & \ddots & & & \\ & & \ddots & & \ddots & & \\ & & & \ddots & & \ddots & \\ & & & & \ddots & & * \\ & & & & & \ddots & \vdots \\ & & & & & & * \end{bmatrix}. \tag{7.1}$$

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.

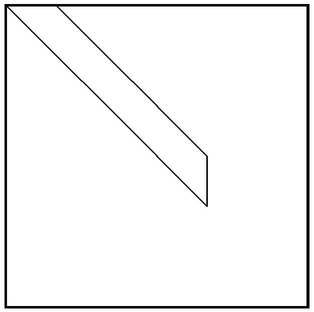
Proposition 7.1. *For $b > 2$, there exists an algorithm that takes as input an upper b-banded $A \in \mathsf{R}^{n\times n}$, and produces as output an upper $(\lfloor b/2 \rfloor + 1)$-banded matrix A' that is equivalent to A.*

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

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

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

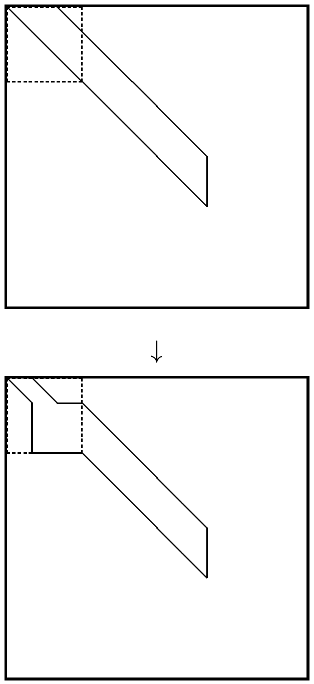
$$B = \left[\begin{array}{c|c} B_1 & B_2 \\ \hline & B_3 \end{array}\right]$$

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

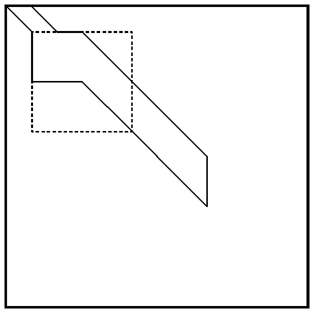
$$B' = \left[\begin{array}{c|c} B_1 & B_2 \\ \hline & B_3 \end{array}\right] \left[\begin{array}{c|c} I_{s_1} & \\ \hline & W \end{array}\right]. \tag{7.2}$$

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:



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

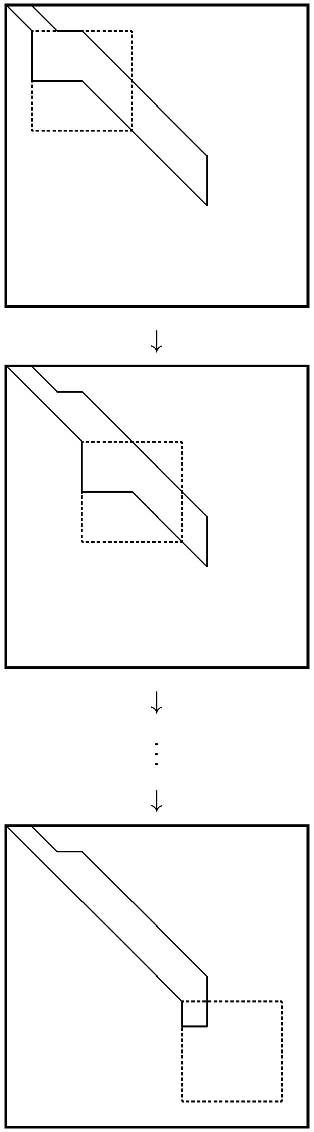
$$C = \left[\begin{array}{c|c} C_1 & C_2 \\ \hline & C_3 \end{array}\right]$$

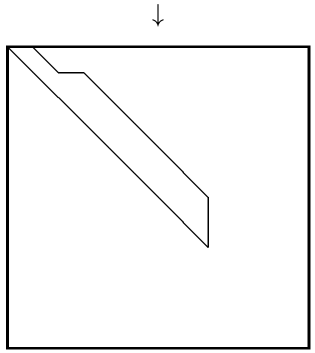
where each block is $s_2 \times s_2$. Use the algorithm of Lemma 3.1 to compute, in succession, a principal transform U^T such that $C_1^T U^T$ is lower triangular, and then a principal transform V such that $(UC_2)V$ is lower triangular. Set

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

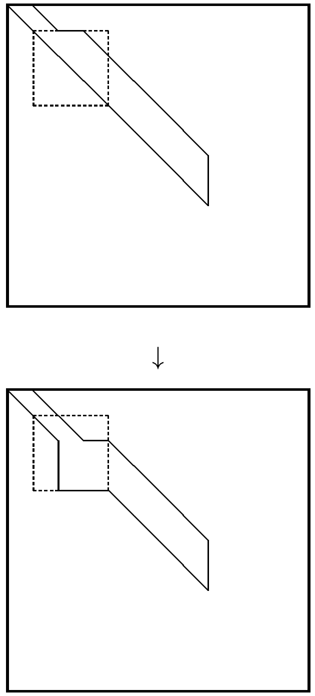
Since $n_2 < 2b$, the cost is as stated. $\square$

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.





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.



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 $f_n(b) \le f_n(b/2) + O(n^2 b^{\theta-2})$ 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 $O(n^\theta (\log \beta) + n^2 (\log n)\, \mathrm{B}(\log \beta))$ 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 $f(n) \leq 2f(n/2) + O(n^\theta)$.

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. □

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

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

□

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 $f(n) \leq 4f(n/2) + O(n^\theta)$. 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 $\mathrm{last}(A_1)$ divides all entries in $\mathrm{diag}(A_2, B_1, B_2))$. This condition on $\mathrm{last}(A_1)$ 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 $\mathrm{diag}(A_2, B_2)$ 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 $\mathrm{last}(A_2)$ divides all entries in $\mathrm{diag}(B_1, B_2)$; this condition on $\mathrm{last}(A_2)$ 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 accomplished 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**
     $(g, s, t, u, v) := \mathrm{Gcdex}(A[i-1, i], A[i, i])$;
     $$\begin{bmatrix} A[i-1,*] \\ A[i,*] \end{bmatrix} := \begin{bmatrix} s & t \\ u & v \end{bmatrix} \begin{bmatrix} A[i-1,*] \\ A[i,*] \end{bmatrix}$$

od;
for i **from** n_1+2 **to** $n-1$ **do**
    $(g,s,t,u,v) := \mathrm{Gcdex}(A[i,i], A[i+1,i])$;
    $\begin{bmatrix} A[i,*] \\ A[i+1,*] \end{bmatrix} := \begin{bmatrix} s & t \\ u & v \end{bmatrix} \begin{bmatrix} A[i,*] \\ A[i+1,*] \end{bmatrix}$
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 $n_1 + 1 + n_2 = n$.

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

$$\downarrow$$

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

$$\downarrow$$

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

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 $(\mathrm{diag}(U_1, I_1, U_2), \mathrm{diag}(V_1, I_1, V_2))$ to get

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

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**
   $(s_i, s, t, u, v) := \mathrm{Gcdex}(A[i,i], A[i,k]);$
   $\left[\begin{array}{cc} A[*,i] & A[*,k] \end{array}\right] := \left[\begin{array}{cc} A[*,i] & A[*,k] \end{array}\right] \left[\begin{array}{cc} s & u \\ t & v \end{array}\right]$
   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

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

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

$$\left[\begin{array}{cccccc} s_1 & & & & & \\ & s_2 & & & & \\ & & s_3 & & & \\ & & & \ddots & & \\ & & & & s_{k-1} & \\ & & & & & s_k \\ & & & & & \end{array}\right].$$

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 $\beta = (\sqrt{r}||A||)^r$.*

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.

Let $A \in \mathbb{Z}^{n\times m}$ have rank r. Let $D \in \mathbb{Z}^{r\times r}$ be the principal $r \times r$ submatrix of the Smith form of A. Consider matrices $U \in \mathbb{Z}^{n\times n}$ and $V \in \mathbb{Z}^{m\times m}$ such that UAV equals the Smith form of A. Any such U and V can be partitioned using a conformal block decomposition as

$$\overset{U}{\left[\begin{array}{c} E \\ \hline \\ M \\ \\ \end{array}\right]} \overset{A}{\left[\begin{array}{c|c} * & * \\ \hline & \\ * & * \\ & \end{array}\right]} \overset{V}{\left[\begin{array}{c|c} F & N \end{array}\right]} = \left[\begin{array}{c|c} D & \\ \hline & \\ & \\ & \end{array}\right] \tag{8.1}$$

where $EAF = D$. The matrices U and V will be unimodular precisely when M is a basis for the left and N a basis for the right nullspace of A. Let $\beta = (\sqrt{r}||A||)^r$. We show how to recover U and V in $O\tilde{}(nmr^{\theta-2}(\log \beta) + nmB(\log \beta))$ bit operations.

In general, the transforms U and V are highly nonunique. The main contribution here is to produce U and V (in the time stated above) with good bounds on the size of entries. We get $||M||, ||N|| \leq r\beta^2$, $||F|| \leq r^3\beta^3$ and $||E|| \leq r^{2r+5}\beta^4 \cdot ||A||$.

This chapter is organized into two sections. In Section 8.1 we demonstrate how to construct a small postmultipler for the Smith form of a full columns rank input matrix.

Notes

Many algorithms have been proposed and substantial progress has been made bounding from above the bit complexity of this problem. Table 8.1 summarizes results for the case of an $n \times n$ input matrix A by giving for each algorithm a triple (Time, Space, Type). The Time and Space columns give the exponents e_1 and f_1 such that the corresponding algorithm has running time bounded by $O\tilde{}(n^{e_1}(\log ||A||)^{e_2})$ bit operatons and intermediate space requirments bounded by $O\tilde{}(n^{f_1}(\log ||A||)^{f_2})$ bits. We neglect to give the exponents e_2 and f_2 (but remark that they are small for all the algorithms, say ≤ 3.) We use this simplified (Time, Space, Type) "one-paramter" complexity model only when summarizing — our primary interest is the complexity in the paramater n.

The Time bounds given for algorithms 5, 7, 8, 9 and 12 allow $\mathrm{M}(k) = k^2$, that for algorithm 11 assumes $\mathrm{M}(k) = O(k^{\theta-1})$, and the other bounds assume $\mathrm{M}(k) = O\tilde{}(k)$. Algorithms 8 and 10 require the input matrix to be nonsingular. Algorithm 9 calls for a comment since there is no citation. The algorithm is a variation of algorithm 7 — the improvment in space complexity is achieved by incorporating p-adic lifting *ala* Dixon (1982), see also (Mulders and Storjohann, 1999), into the algorithm of Storjohann (1996a) and applying software pipeling between that algorithm and the one in (Storjohann, 1998a). We will present this in a future paper.

The actual time complexity of algorithm 12 depends on the cost a matrix vector product involving A; the stated Time bound is valid (for example) for an input matrix which has $O\tilde{}(n)$ nonzero entries.

We suggest that algorithms 8, 9, 11 and 12 are emminently suitable for implementation.

	Citation	Time	Space	Type
	Smith form of a dense integer matrix			
1	Kannan and Bachem (1979)	finite	finite	DET
2	Iliopoulos (1989a)	5	3	DET
3	Hafner and McCurley (1991)	5	3	DET
4	Giesbrecht (1995a)	4	3	LV
5	Giesbrecht (1995a)	$\theta+1$	2	MC
6	Storjohann (1996c)	$\theta+1$	3	DET
7	Storjohann (1998a)	4	3	DET
8	Eberly, Giesbrecht and Villard (2000)	3.5	2	MC
9	Storjohann (200x)	4	2	DET
	Transforms for Smith form of a dense integer matrix			
10	Iliopoulos (1989b)	$\theta+2$	4	DET
11	Storjohann (here)	$\theta+1$	3	DET
	Smith form of a sparse integer matrix			
12	Giesbrecht (1996)	3	2	MC

Figure 8.1: Complexity bounds for Smith form computation

The comments we make in the second last paragraph on page 94 are applicable here as well. Heuristic algorithms for Smith form computation are given by Havas *et al.* (1993) and Havas and Majewski (1997). Also, the lattice basis reduction based techique mentioned on page 94 for reducing the size of numbers in the transforming matrices are applicable here as well.

8.1 Computing a Smith Conditioner

In this section we show how to construct a small post-multiplier matrix for the Smith form of an integer matrix. We take a more abstract approach an present the main results over an abstract PIR. Let R be a PIR.

Let $A \in \mathsf{R}^{n\times n}$ have Smith form $S = \mathrm{diag}(s_1, s_2, \ldots, s_n)$. Let $e_i = \mathrm{Div}(s_i, s_{i-1})$ for $1 \leq i \leq n$, $s_0 = 1$. Then $S = \mathrm{diag}(e_1, e_1e_2, \ldots, e_1e_2\cdots e_n)$.

The next lemma is obvious.

Lemma 8.1. *Assume A is left equivalent to S. Then the matrix obtained from A by*

- *adding any multiple of a latter to a former column, or*

- *adding a multiple of $e_{i+1}e_{i+2}\cdots e_j$ times column i to column j $(i<j)$*

is also left equivalent to S.

Definition 8.2. *We say that A is in* triangular Smith form *if A is upper triangular with each diagonal entry an associate of the corresponding diagonal entry in S.*

A matrix in triangular Smith form is easy to transform to Smith form.

Lemma 8.3. *If A is in triangular Smith form, there exists a unit upper triangular V such that AV is in Smith form.*

The next lemma is analogous to Lemma 8.1 but for triangular Smith forms.

Lemma 8.4. *Assume A is left equivalent to triangular Smith form. Then the matrix obtained from A by*

- *ading any multiple of a former to a latter column, or*
- *adding a multiple $e_{j+1}e_{j+2}\cdots e_i$ times column i to column j $(i>j)$*

is also left equivalent to a triangular Smith form.

Proof. Assume, without loss of generality, that A is in triangular Smith form. Since we may restrict our attention the the symmetric submatrix comprised of rows and column $i,\ldots,j$, which is also in triangular Smith form, we may assume that $i=n$ and $j=1$. Since A can be written as $\bar{A}e_1$ with $\bar{A}[1,1]=1$, we may assume that $e_1=1$.

Let $d=e_2\ldots e_n$. Then there exists a $v\in \mathsf{R}^{1\times n}$ such that $vA=\begin{bmatrix} d & 0 & \cdots & 0\end{bmatrix}$. To see this, let D be the diagonal matrix with $D[i,i]=e_{i+1}\cdots e_n$. Then DA has all diagonal entries equal to d and each off-diagonal entry divisible by d. The existence of v follows easily.

Let $w\in\mathsf{R}^{n\times 1}$ be the last column of A. Then for any $t\in\mathsf{R}$,

$$(I_n - twv)A\begin{bmatrix} 1 & & & \\ & 1 & & \\ & & \ddots & \\ td & & & 1\end{bmatrix} = A.$$

The determinant of the matrix on the left is given by $1-tvw$. By construction of v and w, we have $vw=0$, and thus the left transformation is unimodular. $\square$

Lemma 8.5. *Assume A is left equivalent to a triangular Smith form. Then every Hermite form of A is in triangular Smith form.*

Proof. Let H be a Hermite form of A. Then H is left equivalent to a triangular Smith form of A; for each i, there exists an R-linear combination of rows of H which is equal to row i of a triangular Smith form of A. The only row of H with principal entry nonzero is row one. It follows that $H[1,1]$ is the first diagonal entry in the Smith form of A. Since all entries in $H[1,*]$ are divisible by $H[1,1]$, the only multiple of $H[1,*]$ which has principal entry zero is the zero vector. We must have $H[2,2]$ equal to the second diagonal entry in the Smith form of A. Now use induction on i. $\square$

Definition 8.6. *A unit lower triangular matrix C is called a* Smith conditioner for *A if AC is left equivalent to a triangular Smith form.*

Proposition 8.7. *Let $A\in\mathsf{R}^{n\times n}$ with Smith form $S=\mathrm{diag}(e_1,e_1e_2,\ldots,e_1e_2\cdots e_n)$ be given. Given a Smith conditioner L for A, the following matrices:*

- *a unit lower triangular C with $C[i,j]\in\mathcal{R}(\mathsf{R},e_{j+1}e_{j+2}\cdots e_i)$ for $i>j$, and*
- *a unit upper triangular R with $R[i,j]\in\mathcal{R}(\mathsf{R},e_{i+1}e_{i+2}\cdots e_j)$ for $i<j$,*

such that ACR is left equivalent to S, can be computed in $O(n^\theta)$ basic operations of type {Arith, Quo}.

Corollary 8.8. *Let $\mathsf{R}=\mathbb{Z}/(N)$. The complexity bound of Proposition 8.7 becomes $O(n^\theta(\log\beta)+n^2(\log n)\,\mathrm{B}(\log\beta))$ word operations where $\beta=nN$.*

Proof. Set E to be the strictly lower triangular with $E_{i,j}=e_{j+1}e_{j+2}\cdots e_i$ for $i>j$ and compute a matrix V as in Example 3.13. Set $C=LV$. By Lemma 8.4, C will also be a Smith conditioner for A.

Compute a Hermite form H of AC. Then H is a triangular Smith form of A (Lemma 8.5) and there exists a unit upper triangular T such that HT is in Smith form (Lemma 8.3). Write H as $S\bar{H}$ where $\bar{H}$ is unit upper triangular and compute $T=\bar{H}^{-1}$ using a Hermite reduction transform. Then ACT is left equivalent to S.

Set E to be the strictly upper triangular with $E_{i,j}=e_{i+1}e_{i+2}\cdots e_j$ for $i<j$ and compute a matrix V as in Example 3.14. Set $R=TV$. By Lemma 8.1, ACR will also be left equivalent to S. $\square$

Proposition 8.7 starts with a Smith conditioner L of A. If the following two conditions are met

- R is a stable ring, and
- the Smith form of A has all diagonal entries nonzero

then we can recover such an L from any transform $(*, V)$ such that AV is left equivalent to a Smith form of A. The idea is to use Lemma 2.21 to produce a lower triangular L such that V can be expressed as the product $L *_1 *_2$ for some upper triangular $*_1$ and lower triangular $*_2$. Such an L will be a Smith conditioner since $AL*_1$ is left equivalent to a Smith form (Lemma 8.1) and AL is left equivalent to a triangular Smith form (Lemma 8.4).

Notes

The idea of a "Smith conditioner" was first used by Kaltofen, Krishnamoorthy & Saunders (1987, 1990) for polynomial matrices; there C was chosen randomly. By first postmultiplying an input matrix by a Smith conditioner, the problem of computing the Smith form is reduced to that of computing an echelon form. Villard's (1995) algorithm for Smith form over $\mathbb{Q}[x]$ — which first showed the problem to be in $\mathcal{P}$ — recovers the entries of C one by one while transforming the input matrix to echelon form. Villard also uses the term "triangular Smith form". Giesbrecht's (1995a) randomized Smith normal form algorithm for integer matrices computes the columns of C by, in essence, using a Las Vegas probabilistic algorithm to obtain solutions to the modulo N extended gcd problem. In (Storjohann, 1997) we show how to produce a Smith conditioner for an integer input matrix which has $\log ||C|| = O((\log n + \log\log ||A||)^2)$. In (Mulders and Storjohann, 1998) the modulo N extended gcd problem for polynomials is studied and applied to get a small Smith conditioner when $\mathsf{R} = \mathsf{K}[x]$.

Worked Example over $\mathbb{Z}$

Our goal is to recover the Smith form of

$$A = \begin{bmatrix} 14 & 8 & -26 & -14 & 13 & 7 \\ 6 & -30 & 16 & -14 & -17 & 13 \\ -8 & -20 & 14 & 20 & 20 & 2 \\ 46 & -14 & 0 & 18 & -15 & 3 \\ -6 & -18 & -18 & 18 & -39 & -3 \\ 8 & -4 & 6 & -36 & 6 & -24 \end{bmatrix}$$

over $\mathbb{Z}$. We compute the determinant of this matrix to be $d = 3583180800$. Working modulo d we recover the Smith form $S = \operatorname{diag}(1, 2, 6, 12, 48, 518400)$ of A. Now we want to recover a transform (U, V) such that $UAV = S$. Recall that $\cong_N$ means left equivalent modulo N (see Section 5.1). Working modulo $N = 2 \cdot 514800$ we recover a

$$V = \begin{bmatrix} 610263 & 739906 & 924658 & 295964 & 566625 & 460949 \\ 842660 & 884554 & 744338 & 476716 & 573129 & 717178 \\ 811572 & 434451 & 242395 & 581038 & 600751 & 755646 \\ 302520 & 914154 & 532124 & 430481 & 214158 & 150757 \\ 843932 & 720594 & 858593 & 51322 & 732251 & 81872 \\ 258189 & 13326 & 683869 & 875588 & 385203 & 622164 \end{bmatrix} \in \mathbb{Z}^{6\times 6}$$

such that $AV \cong_N S$. Note that this V satisfies $\det V \perp N$ but not $\det V = \pm 1$. As discussed above, use Lemma 2.21 to recover from V a unit lower triangular L which will be a Smith conditioner for A. We get

$$L = \begin{bmatrix} 1 & & & & & \\ 942681 & 1 & & & & \\ 254870 & 704138 & 1 & & & \\ 228897 & 218885 & 568609 & 1 & & \\ 1013963 & 3368 & 240655 & 559257 & 1 & \\ 914176 & 513121 & 747963 & 326960 & 862874 & 1 \end{bmatrix}.$$

Now construct C and R as in Proposition 8.7 so that $S \cong_N ACR$.

$$C = \begin{bmatrix} 1 & & & & & \\ 1 & 1 & & & & \\ 4 & 2 & 1 & & & \\ 11 & 5 & 1 & 1 & & \\ 1 & 8 & 7 & 1 & 1 & \\ 278066 & 227881 & 50475 & 20416 & 9674 & 1 \end{bmatrix}$$

and

$$R = \begin{bmatrix} 1 & 1 & 0 & 3 & 27 & 201749 \\ & 1 & 0 & 0 & 18 & 191429 \\ & & 1 & 1 & 3 & 62592 \\ & & & 1 & 2 & 581 \\ & & & & 1 & 5166 \\ & & & & & 1 \end{bmatrix}.$$

Note that CR is unimodular over $\mathbb{Z}$. We conclude that ACR is left equivalent (over $\mathbb{Z}$) to S.

Some Bounds

Let $A \in \mathbb{Z}^{n \times m}$ have full column rank. Let $D = \mathrm{diag}(s_1, s_2, \ldots, s_m)$ be the principal $m \times m$ submatrix of the Smith form of A. Above we saw how to recover a unimodular $V \in \mathbb{Z}^{m \times m}$ such that AV is left equivalent to D. The V recovered using the technique there can be expressed as the product of a unit lower triangular C and unit upper triangular R such that

- $0 \leq C[i,j] < s_i/s_j$ for $i > j$, and
- $0 \leq R[i,j] < s_j/s_i$ for $j > i$.

Row i of C has entries bounded in magnitude by s_i. Similarly, column j of R has entries bounded in magnitude by s_j. Part (a) of the next lemma follows. Part (b) follows from Hadamard's inequality. Part (c) from part (b).

Lemma 8.9. *Let C and R satisfy the bounds given above. Then*

(a) $||V|| < ns_m^2$. *The total size of V is $O(n^2 \log n + n \log \beta)$.*

(b) $||C^{-1}||, ||R^{-1}|| \leq n^{n/2}\beta$.

(c) $||V^{-1}|| \leq nn^n\beta^2$.

8.2 The Algorithm

Let $A \in \mathbb{Z}^{n \times m}$ have rank r. By transposing A if necessary, we may assume, without loss of generality, that $m \leq n$. Let D be the principal $r \times r$ submatrix of the Smith form of A. The nullspaces M and N are recovered using Proposition 6.6. The following procedure shows how to recover an E and F such that $EAF = D$. Then we paste to get a Smith transform as in 8.1.

1. By computing a GaussJordan transform for A recover the following quantities: the rank r of A; permutation matrices P and Q such PAQ has principal $r \times r$ submatrix B nonsingular; the determinant d and adjoint B^{adj} of B.

$$PAQ = \left[\begin{array}{c|c} B & * \\ \hline * & * \end{array}\right]$$

2. Apply Corollary 3.16 to recover an $E_1 \in \mathbb{Z}^{r \times n}$ such that $E_1 PAQ$ is in Hermite form. Let H_1 be the principal $r \times r$ submatrix of $E_1 PAQ$.

$$\begin{bmatrix} E_1 \\ * \end{bmatrix} \overbrace{\left[\begin{array}{c|c} B & * \\ \hline * & * \end{array}\right]}^{PAQ} = \left[\begin{array}{c|c} H_1 & * \end{array}\right]$$

3. If $r = m$ then set $(F_1, H_2) := (I_r, B)$ and goto step 4. Otherwise, apply Corollary 3.16 to recover an $F_1 \in \mathbb{Z}^{m\times r}$ such that $(F_1)^T(PAQ)^T$ is in Hermite form. Let H_2 be the principal $r \times r$ submatrix of $PAQF_1$.

$$\overbrace{\left[\begin{array}{c|c} B & * \\ \hline & \\ * & * \\ & \end{array}\right]}^{PAQ} \overbrace{\left[\begin{array}{c} \\ * \\ \\ \end{array}\right]}^{F_1} = \left[\begin{array}{c} H_2 \\ \hline \\ * \\ \\ \end{array}\right]$$

4. Let $G := (1/d)H_1B^{\mathsf{adj}}H_2$.
Note: $G = E_1PAQF_1$.

$$\overbrace{\left[\begin{array}{c} * \end{array}\right]}^{E_1} \overbrace{\left[\begin{array}{c|c} B & * \\ \hline & \\ * & * \\ & \end{array}\right]}^{PAQ} \overbrace{\left[\begin{array}{c} \\ * \\ \\ \end{array}\right]}^{F_1} = G$$

5. Let $N = 2|\det H_2|$ and use Proposition 7.14 to recover the Smith form $D = \mathrm{diag}(s_1, s_2, \ldots, s_r)$ of G together with a $V \in \mathbb{Z}^{r\times r}$ such that $D \equiv_N GV$.

6. Use the technique detailed in Section 8.1 to recover from V a unit lower triangular $C \in \mathbb{Z}^{r\times r}$ and unit upper triangular $R \in \mathbb{Z}^{r\times r}$ with GCR left equivalent to D and
   - $0 \le C[i,j] < s_i/s_j$ for $i > j$, and
   - $0 \le R[i,j] < s_j/s_i$ for $j > i$.

7. Set $F := F_1CR$.
Set $E := (1/\det(H_1H_2))DR^{-1}C^{-1}H_2^{\mathsf{adj}}BH_1^{\mathsf{adj}}E_1$.

Proposition 8.10. *Let $A \in \mathbb{Z}^{n\times m}$. Unimodular $U \in \mathbb{Z}^{n\times n}$ and $V \in \mathbb{Z}^{m\times m}$ such that UAV is in Smith form can be recovered in*

$$O(nmr^{\theta-2}(\log nm)(\log\beta) + nm(\log nm)\,\mathsf{B}(\log\beta))$$

word operations where $\beta = (\sqrt{r}||A||)^r$. Moreover

- *Entries in the last $n-r$ rows of U and last $m-r$ columns of V will be bounded in magnitude by $r\beta^2$.*
- *Entries in the first r columns of V will be bounded in magnitude by $r^3\beta^3$.*
- *Entries in the first r rows of U will be bounded in magnitude by $r^{2r+5}\beta^4 \cdot ||A||$.*

Corollary 8.11. *If $r = m$ the bound for $||V||$ becomes rs_m^2, where s_m is the last diagonal entry in the Smith form, and the total size of V will be $O(m^2(\log m + \log ||A||))$.*

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.

Let K be a field. Corresponding to any matrix $A \in \mathsf{K}^{n\times n}$ there exists an invertible U over K such that

$$U^{-1}AU = F = \begin{bmatrix} C_{f_1} & & & \\ & C_{f_2} & & \\ & & \ddots & \\ & & & C_{f_l} \end{bmatrix} \in \mathsf{K}^{n\times n}.$$

F is the Frobenius canonical form of A, also called the rational canonical form. Each block C_{f_i} is the companion matrix of a monic $f_i \in \mathsf{K}[x]$ and $f_i|f_{i+1}$ for $1 \le i \le l-1$. The minimal polynomial of A is f_l and the characteristic polynomial is the product $f_1 f_2 \cdots f_l$. The determinant of A is given by the constant coefficient of $f_1 f_2 \cdots f_l$. Recall that the companion matrix of $g = g_0 + g_1 x + \cdots + g_{r-1}x^{r-1} + x^r \in \mathsf{K}[x]$ is

$$C_g = \begin{bmatrix} 0 & \cdots & 0 & -g_0 \\ 1 & \ddots & \vdots & \vdots \\ & \ddots & 0 & -g_{r-2} \\ & & 1 & -g_{r-1} \end{bmatrix} \in \mathsf{K}^{r\times r}. \tag{9.1}$$

This chapter demonstrates that a transform for the Frobenius form can be recovered in $O(n^3)$ field operations assuming standard matrix multiplication. Our primary purpose, though, is to show how to incorporate matrix multiplication and reduce the running time to $O(n^\theta(\log n)(\log\log n))$. Both the standard and fast algorithm require only operations of type $\{+, -, \times,$ divide by a nonzero $\}$ from K. We now sketch the two algorithms.

The standard algorithm The algorithm proceeds in two stages. In

$$A = \begin{bmatrix} * & * & * & * & * & * & * & * & * & * \\ * & * & * & * & * & * & * & * & * & * \\ * & * & * & * & * & * & * & * & * & * \\ * & * & * & * & * & * & * & * & * & * \\ * & * & * & * & * & * & * & * & * & * \\ * & * & * & * & * & * & * & * & * & * \\ * & * & * & * & * & * & * & * & * & * \\ * & * & * & * & * & * & * & * & * & * \\ * & * & * & * & * & * & * & * & * & * \\ * & * & * & * & * & * & * & * & * & * \end{bmatrix} \longrightarrow Z = \begin{bmatrix} C_* & B_* & & & & & & \\ & C_* & & & & & & \\ & B_* & C_* & B_* & & & & \\ & & & C_* & & & & \\ & & & & \ddots & & & \\ & & & & & C_* & & \\ & & & & & B_* & C_* & B_* \\ & & & & & & & C_* \end{bmatrix}$$

Figure 9.1: Transformation to Zigzag form

Section 9.2 we give an $O(n^3)$ field operations algorithm for transforming A to Zigzag form Z — a matrix with an "almost" block diagonal structure and fewer than $2n$ nonzero entries, see Figure 9.1. The algorithm is naive; the approach used is Gaussian elimination but using elementary similarity tranformations (instead of only elementary row operations). Recall that an elementary similarity transformation of a square matrix A is given by $A \to EAE^{-1}$ where E is the elementary matrix corresponding to an elementary row operation. More precisely: switching rows i and j is followed by switching the same columns; multiplying row i by a is followed by multiplying the same column by a^{-1}; adding a times row j to row i is followed by subtracting a times column i from column j.

In Section 9.4 an algorithm is given for transforming a Zigzag from Z to Frobenius form F, see Figure 9.2. The approach exploits some of the theory concerning modulo isomorphism — especially the links between matrices over K (similarity transformation) and over $\mathsf{K}[x]$ (equivalence transformation). This is recalled in Section 9.1. In particular, the algorithm here is inspired by the recursive algorithm supporting Theo-

$$Z = \begin{bmatrix} C_* & B_* & & & & & & \\ & C_* & & & & & & \\ & B_* & C_* & B_* & & & & \\ & & & C_* & & & & \\ & & & & \ddots & & & \\ & & & & & C_* & & \\ & & & & & B_* & C_* & B_* \\ & & & & & & & C_* \end{bmatrix} \longrightarrow F = \begin{bmatrix} C_* & & & & & & & \\ & C_* & & & & & & \\ & & C_* & & & & & \\ & & & C_* & & & & \\ & & & & \ddots & & & \\ & & & & & C_* & & \\ & & & & & & C_* & \\ & & & & & & & C_* \end{bmatrix}$$

Figure 9.2: From Zigzag to Frobenius Form

rem 7.12 for the transformation of a bidiagonal matrix over a PIR to Smith form. The combining phase requires us to transform a block diagonal matrix to Frobenius form. This is presented in Section 9.3. As a corollary of Sections 9.2, 9.3 and 9.4 we get an $O(n^3)$ algorithm for the Frobenius form.

$$\begin{array}{ccc} \S 9.2 & & \S 9.3 \\ & \searrow \quad \swarrow & \\ & \S 9.4 & \end{array}$$

Figure 9.3: Standard Algorithm — Suggested Reading Order

The fast algorithm The last three sections of this chapter develop the fast algorithm. The first step of the algorithm is to transform the input matrix to block upper triangular shifted Hessenberg form, see Figure 9.4. The transformation to Hessenberg form costs $O(n^\theta(\log n))$ field

$$A = \begin{bmatrix} * & * & * & * & * \\ * & * & * & * & * \\ * & * & * & * & * \\ * & * & * & * & * \\ * & * & * & * & * \end{bmatrix} \longrightarrow H = \begin{bmatrix} C_* & B_* & \cdots & B_* \\ & C_* & \cdots & B_* \\ & & \ddots & \vdots \\ & & & C_* \end{bmatrix}$$

Figure 9.4: Transformation to Hessenberg form

operations using the algorithm of Keller-Gehrig (1985). Section 9.7 gives a divide and conquer algorithm for transforming a Hessenberg to Frobenius from. The key step in the algorithm, presented in Section 9.6, is the combining phase; this requires fewer than $O(\log\log n)$ applications of Keller-Gehrig's (1985) algorithm for Hessenberg form. By comparison, Giesbrecht (1993) first showed, using randomization, that computing the Frobenius form can be reduced to an expected constant number of Hessenberg computations.

$$\begin{tikzcd}
 & \S 9.1 \arrow[dl] \arrow[dr] & \\
\S 9.3 & & \S 9.5 \\
\S 9.4 \arrow[dr] & & \S 9.6 \arrow[dl] \\
 & \S 9.7 &
\end{tikzcd}$$

Figure 9.5: Fast Algorithm — Suggested Reading Order

The algorithms we develop here use ideas from Keller-Gehrig (1985), Ozello (1987), Kaltofen, Krishnamoorthy and Saunders (1990), Giesbrecht (1993), Villard (1997) and Lübeck (1998).

Notes

Many algorithms have been proposed and substantial progress has been made in bounding from above the algebraic complexity of this problem. Figure 9.6 summarizes results for the case of an $n \times n$ input matrix over a field K by giving for each algorithm a quadruple (Time, Trans, Cond, Type). Time is the asymptotic complexity. A $\bullet$ in the column labelled Trans indicates that a transition matrix is recovered. Some of the algorithms assume a condition on $q = \#\mathsf{K}$, this is indicated in column Cond. Finally, the Type of the algorithm is as before: deterministic (DET), randomized Las Vegas (LV) or randomized Monte Carlo (MC).

Augot and Camion (1994) give various specialized results. For example, knowing the factorization of the characteristic polynomial, the Frobenius form can be computed in $O(n^3 m)$ field operations where m is the number of factors in the characteristic polynomial counted with multiplicities. Over a finite field they show that $m = O(\log n)$ in the asymptotic average case. In the worst case $m = n$.

	Citation	Time	Trans	Cond	Type
	Hessenberg form of a dense matrix				
1	Keller-Gehrig (1985)	$n^\theta(\log n)$	$\bullet$		DET
2	——	n^3	$\bullet$		DET
	Frobenius form of a dense matrix				
3	Lüneburg (1987)	n^4	$\bullet$		DET
4	Ozello (1987)	n^4	$\bullet$		DET
5	Augot and Camion (1994)	n^4	$\bullet$		DET
6	Giesbrecht (1995b)	n^3	$\bullet$	$q \geq n^2$	LV
7	——	$n^3(\log_q n)$	$\bullet$		LV
8	——	$n^\theta(\log n)$	$\bullet$	$q \geq n^2$	LV
9	Steel (1997)	n^4	$\bullet$		DET
10	Storjohann (1998b)	n^3			DET
11	Storjohann and Villard (2000)	n^3	$\bullet$		DET
12	Eberly (2000)	$n^\theta(\log n)$	$\bullet$		LV
13	Storjohann (here)	$n^\theta(\log n)(\log\log n)$	$\bullet$		DET
	Frobenius form of a sparse matrix				
14	Villard (1993)	$n^{2.5}$		$q > 2$	MC
15	Eberly (2000)	n^3	$\bullet$		LV

Figure 9.6: Complexity bounds for Frobenius form computation

Now consider the randomized algorithms. Algorithms 6, 8 and 14 can be applied over a fields which are "too small" by working instead over an algebraic extension of the ground field. Use of this trick has two consequences. First, the complexity will increase by a polylogarithmic factor. Second, and more importantly, if the algorithm produces a transition matrix this might not be over the ground field but over the extension field instead. One of the questions left open by Giesbrecht, and recently solved by Eberly (2000), is whether such a transition matrix can be computing asymptotically quickly in the small field case.

Now consider the $O(n^3)$ deterministic algorithms. The first stage of the algorithm in (Storjohann, 1998b) is the same as here — transformation to Zigzag form — but the transformation from Zigzag to Frobenius form proposed there does not recover a transistion matrix. We solve that problem here by adapting the divide and conquer algorithm of Section 7.3. A more inspired algorithm for the Zigzag to Frobenius transformation phase has been proposed by Villard (1999). Villard's method is iterative. The $O(n^3)$ running time is achieved by exploiting the sparsity of Z and avoiding matrix multiplication; instead of performing operations with matrix blocks only operations with key vectors of the transformation matrix are used. By incorporating the recursive approach here the running time for the Zigzag to Frobenius phase is reduced to $O((n\log n)^2)$ field operations. This is presented in (Storjohann and Villard, 2000).

Now consider the algorithms for sparse input matrices. The running time estimates given in the table assume an input matrix which has only $O^{\tilde{}}(n)$ entries. More precisely, Eberly's algorithm requires an expected number of $O(n)$ multiplications of the input matrix A by vectors, $O(n)$ multiplications of the transpose A^T by vectors, and additional $O(kn^2)$ field operations, where k is the number of blocks in the Frobenius form. Villard's algorithm requires $O(\mu n \log n)$ multiplication of A by vectors, and additional $O(\mu n^2 \log n \log\log n)$ field operations, where μ is the number of distinct blocks in the Frobenius form. Since $k = O(n)$ and $\mu = O(n^{1/2})$ the worst case running times are as stated in the table.

Notation

A block is a submatrix comprised of a contiguous sequence of rows and columns. For $g = g_0 + g_1x + \cdots + g_{r-1}x^{r-1} + x^r \in \mathsf{K}[x]$, let $C_g \in \mathsf{K}^{r\times r}$ denote the companion matrix of g as shown in (9.2).

$$C_g = \left[\begin{array}{cccc} 0 & \cdots & 0 & -g_0 \\ 1 & \ddots & \vdots & \vdots \\ & \ddots & 0 & -g_{r-2} \\ & & 1 & -g_{r-1} \end{array}\right] \in \mathsf{K}^{r\times r}. \tag{9.2}$$

When using C_g as a block label, we allow the special case $g = 1$ in which case C_g has zero rows and columns.

Let $b = b_0 + b_1x + \cdots + b_dx^d \in \mathsf{K}[x]$. We use the label B_b to denote a block which has all entries zero except for entries in the last column row i equal to $-b_{i-1}$ for $1 \le i \le d+1$. The dimensions of a block labeled B_b will be conformal with adjacent blocks. Note that B_b may have zero columns and should have at least $1 + \deg b$ rows if $b \neq 0$.

Every matrix $A \in \mathsf{K}^{n\times n}$ can be written using a block decomposition as

$$\left[\begin{array}{cccc} C_{c_1} & B_{b_{12}} & \cdots & B_{b_{1k}} \\ B_{b_{21}} & C_{c_2} & & B_{b_{2k}} \\ \vdots & & \ddots & \vdots \\ B_{b_{k1}} & B_{b_{k2}} & \cdots & C_{c_k} \end{array}\right]$$

for some (not necessarily unique) choices of the c_*'s and b_{**}'s. In particular, we allow that some of the diagonal companion blocks in the decomposition might be 0×0. But if we specify that k should be minimal, the decomposition is unique.

9.1 Preliminaries and Subroutines

We begin with (a version of) a classical result. First recall some facts about matrices over $\mathsf{K}[x]$. Let $A(x), S(x) \in \mathsf{K}[x]^{n\times n}$ be nonsingular. $A(x)$ and $S(x)$ are equivalent if $U(x)A(x)V(x) = G(x)$ for unimodular $U(x), V(x) \in \mathsf{K}[x]^{n\times n}$. Moreover, $S(x)$ is in *Smith canonical form* if $S(x)$ is diagonal with each diagonal monic and dividing the next.

Theorem 9.1. (Fundamental Theorem of Similarity over a Field) *Suppose that $A \in \mathsf{K}^{n\times n}$ can be written as*

$$\left[\begin{array}{cccc} C_{c_1} & B_{b_{12}} & \cdots & B_{b_{1k}} \\ B_{b_{21}} & C_{c_2} & & B_{b_{2k}} \\ \vdots & & \ddots & \vdots \\ B_{b_{k1}} & B_{b_{k2}} & \cdots & C_{c_k} \end{array}\right]. \tag{9.3}$$

If the Smith form of

$$\left[\begin{array}{cccc} c_1 & b_{12} & \cdots & b_{1k} \\ b_{21} & c_2 & & b_{2k} \\ \vdots & & \ddots & \vdots \\ b_{k1} & b_{k2} & \cdots & c_k \end{array}\right] \in \mathsf{K}[x]^{k\times k} \quad \textit{is} \quad \left[\begin{array}{cccc} \bar{c}_1 & & & \\ & \bar{c}_2 & & \\ & & \ddots & \\ & & & \bar{c}_k \end{array}\right] \in \mathsf{K}[x]^{k\times k},$$

then

$$\left[\begin{array}{cccc} C_{\bar{c}_1} & & & \\ & C_{\bar{c}_2} & & \\ & & \ddots & \\ & & & C_{\bar{c}_k} \end{array}\right]$$

is the Frobenius form of A.

Note that every $A \in \mathsf{K}^{n\times n}$ can be written as in (9.3) with $k = n$, $c_i = x + A_{ii}$ and $b_{ij} = A_{ij}$. Theorem 9.1 shows that we may recover the Frobenius form of A by computing the Smith form of $xI_n - A \in \mathsf{K}[x]^{n\times n}$ (or vice versa).

Next we give some technical lemmas and then develop the analogue of Theorem 9.1 but for triangular instead of diagonal decompositions.

Lemma 9.2. *Let $A \in \mathsf{K}^{n\times n}$ and $U \in \mathsf{K}^{n\times n}$ be nonsingular. Then $U^{-1}AU$ can be written as*

$$U^{-1}AU = \begin{bmatrix} C_{c_1} & B_{b_{12}} & \cdots & B_{b_{1k}} \\ B_{b_{21}} & C_{c_2} & & B_{b_{2k}} \\ \vdots & & \ddots & \vdots \\ B_{b_{k1}} & B_{b_{k2}} & \cdots & C_{c_k} \end{bmatrix} \tag{9.4}$$

with $d_i = \deg c_i$, $1 \le j \le k$, if and only if

$$U = [v_1|Av_1|\cdots|A^{d_1-1}v_1|\cdots|v_k|Av_k|\cdots|A^{d_k-1}v_k] \tag{9.5}$$

for some $v_i \in \mathsf{K}^{n\times 1}$, $1 \le i \le k$.

Now fix the v_i's. Then $U^{-1}AU$ is block upper triangular if and only if $(d_1, d_2, \ldots, d_k)$ is the lexicographically largest such sequence for which the corresponding U of (9.5) is nonsingular.

Proof. Can be shown using elementary linear algebra. □

There is a special case of Lemma 9.2 that is important enough that we give a name to the form obtained in (9.4).

Definition 9.3. *Let $A \in \mathsf{K}^{n\times n}$. The* shifted Hessenberg form of *A is the block upper triangular matrix*

$$U^{-1}AU = \begin{bmatrix} C_{c_1} & B_{b_{12}} & \cdots & B_{b_{1n}} \\ & C_{c_2} & & B_{b_{2n}} \\ & & \ddots & \vdots \\ & & & C_{c_n} \end{bmatrix}$$

where U is constructed as in (9.5) using $[v_1|v_2|\cdots|v_n] = I_n$ and with $(d_1, d_2, \ldots, d_n)$ chosen lexicographically maximal.

We have shown the Hessenberg form with n diagonal blocks. But note that many of these blocks might be C_1, hence of dimension 0×0. The next result gives an asymptotically fast algorithm to compute the Hessenberg form.

Fact 9.4. (Keller-Gehrig, 1985) *Let $A \in \mathsf{K}^{n\times n}$ together with $v_i \in \mathsf{K}^n$ for $1 \le i \le n$ be given. The following two problems can be accomplished in $O(n^\theta(\log n))$ field operations:*

- *Find the lexicographically largest sequence $(d_1, d_2, \ldots, d_n)$ such that the U in (9.5) is nonsingular, or report that no such sequence exists.*

- *Given a sequence of integers $(d_1, d_2, \ldots, d_n)$, with $0 \le d_i \le n$ and $d_1 + d_2 + \cdots + d_n = n$, construct the matrix U in (9.5).*

Under the assumption of standard matrix multiplication, the cost of solving both problems is $O(n^3)$ field opearations.

Note that in Fact 9.4 we assume we start with a list of n vectors $\{v_*\}$. On the one hand, in the special case where A is, for example, the zero or identity matrix, we require n linearly independant $\{v_*\}$ to construct a nonsingular U. On the other hand, for an A which is not so special, if we construct $(d_1, d_2, \ldots, d_n)$ to be the (unique) lexicographically maximal degree sequence with respect to the v_i's, then typically many of the d_i's will be 0. Note that if $d_i = 0$ then v_i is not used in the construction of U — in other words $c_i = 1$.

Now recall some facts about matrices over $\mathsf{K}[x]$. Let $A(x), H(x) \in \mathsf{K}[x]^{n\times n}$ be nonsingular. $A(x)$ and $H(x)$ are right equivalent if $A(x)V(x) = H(x)$ for a unimodular $V(x) \in \mathsf{K}[x]^{n\times n}$. Moreover, $H(x)$ is in *right Hermite form* if $H(x)$ is upper triangular with $H[i,i]$ monic and $\deg(H[i,i]) > \deg(H[i,j]$ for $1 \le i < j \le n$. Theorem 9.1 showed that the Smith form over $\mathsf{K}[x]$ is the analogue of the Frobenius form. The next theorem shows that the Hermite form over $\mathsf{K}[x]$ is the analogue of the shifted Hessenberg form.

Theorem 9.5. *Suppose that $A \in \mathsf{K}^{n\times n}$ can be written as*

$$\begin{bmatrix} C_{c_1} & B_{b_{12}} & \cdots & B_{b_{1k}} \\ B_{b_{21}} & C_{c_2} & & B_{b_{2k}} \\ \vdots & & \ddots & \vdots \\ B_{b_{k1}} & B_{b_{k2}} & \cdots & C_{c_k} \end{bmatrix}. \tag{9.6}$$

If the right Hermite form of

$$\begin{bmatrix} c_1 & b_{12} & \cdots & b_{1k} \\ b_{21} & c_2 & & b_{2k} \\ \vdots & & \ddots & \vdots \\ b_{k1} & b_{k2} & \cdots & c_k \end{bmatrix} \in \mathsf{K}[x]^{k\times k} \quad \text{is} \quad \begin{bmatrix} c_1 & \bar{b}_{12} & \cdots & \bar{b}_{1k} \\ & \bar{c}_2 & & \bar{b}_{2k} \\ & & \ddots & \vdots \\ & & & \bar{c}_k \end{bmatrix} \in \mathsf{K}[x]^{k\times k},$$

then

$$\begin{bmatrix} C_{\bar{c}_1} & B_{\bar{b}_{12}} & \cdots & B_{\bar{b}_{1k}} \\ & C_{\bar{c}_2} & & B_{\bar{b}_{2k}} \\ & & \ddots & \vdots \\ & & & C_{\bar{c}_k} \end{bmatrix}$$

is the shifted Hessenberg form of A.

Note that every $A \in \mathsf{K}^{n\times n}$ can be written as in (9.6) with $k = n$, $c_i = x + A_{ii}$ and $b_{ij} = A_{ij}$. Lemma 9.5 shows that we may recover the shifted Hessenberg form of A by computing the right Hermite form of $xI_n - A \in \mathsf{K}[x]^{n\times n}$.

Example 9.6. *Let*

$$A = \begin{bmatrix} -1 & -3 \\ -2 & -3 \end{bmatrix} = \begin{bmatrix} C_{x+1} & B_3 \\ B_2 & C_{x+3} \end{bmatrix} \in \mathsf{K}^{2\times 2}.$$

Since the right Hermite form of

$$\begin{bmatrix} x+1 & 3 \\ 2 & x+3 \end{bmatrix} \in \mathsf{K}[x]^{2\times 2} \quad \textit{is} \quad \begin{bmatrix} x^2+4\,x-3 & 1/2\,x+1/2 \\ & 1 \end{bmatrix} \in \mathsf{K}[x]^{2\times 2},$$

we may conclude that the shifted Hessenberg form H of A is

$$H = \begin{bmatrix} C_{x^2+4x-3} & B_{1/2x+1/2} \\ B_0 & C_1 \end{bmatrix} = \begin{bmatrix} & 3 \\ 1 & -4 \end{bmatrix} \in \mathsf{K}^{2\times 2}.$$

The above example shows how to deduce the shifted Hessenberg from from the right Hermite form of $xI_n - A$. But note that we cannot deduce the Hermite form from the corresponding Hesssenberg form because blocks C_1 will by 0×0, and thus we lose the entries in the Hermite form which are in a column with a unit diagonal. The next result shows how to avoid this problem, and gives a method for deducing the Hermite form of certain matrices over $\mathsf{K}[x]$ from their corresonding shifted Hessenberg form over K.

Lemma 9.7. *Suppose that $A(x) \in \mathsf{K}[x]^{k\times k}$ can be written as*

$$\begin{bmatrix} c_1 & b_{12} & \cdots & b_{1k} \\ b_{21} & c_2 & & b_{2k} \\ \vdots & & \ddots & \vdots \\ b_{k1} & b_{k2} & \cdots & c_k \end{bmatrix}$$

with each c_i monic and $\deg b_{i*} < \deg c_i$, $1 \le i \le k$. Suppose that $\deg\det A(x) = n$. Let $\bar{n} = n + k$. If the $\bar{n}\times\bar{n}$ shifted Hessenberg form (over K) of

$$\begin{bmatrix} C_{xc_1} & B_{xb_{12}} & \cdots & B_{xb_{1k}} \\ B_{xb_{21}} & C_{xc_2} & & B_{xb_{2k}} \\ \vdots & & \ddots & \vdots \\ B_{xb_{k1}} & B_{xb_{k2}} & \cdots & C_{xc_k} \end{bmatrix} \quad \textit{is} \quad \begin{bmatrix} C_{x\bar{c}_1} & B_{x\bar{b}_{12}} & \cdots & B_{x\bar{b}_{1l}} \\ & C_{x\bar{c}_2} & & B_{x\bar{b}_{2l}} \\ & & \ddots & \vdots \\ & & & C_{x\bar{c}_l} \end{bmatrix},$$

then

$$\begin{bmatrix} \bar{c}_1 & \bar{b}_{12} & \cdots & \bar{b}_{1l} \\ & \bar{c}_2 & & \bar{b}_{2l} \\ & & \ddots & \vdots \\ & & & \bar{c}_l \end{bmatrix}$$

is the right Hermite form of $A(x)$.

As a corollary of Lemma 9.7 and Fact 9.4 we get the following fast method for computing right Hermite forms of certain matrices.

Proposition 9.8. *Suppose that $A(x) \in \mathsf{K}[x]^{k\times k}$ can be written as*

$$\begin{bmatrix} c_1 & b_{12} & \cdots & b_{1k} \\ b_{21} & c_2 & & b_{2k} \\ \vdots & & \ddots & \vdots \\ b_{k1} & b_{k2} & \cdots & c_k \end{bmatrix}$$

with each c_i monic and $\deg b_{i*} < \deg c_i$, $1 \le i \le k$. Then we can recover the right Hermite form of $A(x)$ using $O(n^\theta(\log n))$ field operations, where $\deg\det A(x) = n$.

We next give an example of Lemma 9.7.

Example 9.9. *Let*

$$A(x) = \begin{bmatrix} x+1 & 3 \\ 2 & x+3 \end{bmatrix} \in \mathsf{K}[x]^{2\times 2}.$$

Since the shifted Hessenberg form of

$$\left[\begin{array}{cc|cc} & 0 & & 0 \\ 1 & -1 & & -3 \\ \hline & 0 & & 0 \\ & -2 & 1 & -3 \end{array}\right] \mathsf{K}^{4\times 4} \quad \textit{is} \quad \left[\begin{array}{ccc|c} & & 0 & 0 \\ 1 & & 3 & -1/2 \\ & 1 & -4 & -1/2 \\ \hline & & & 0 \end{array}\right] \in \mathsf{K}^{4\times 4},$$

we may conclude that the right Hermite form $H(x)$ of $A(x)$ is

$$H(x) = \begin{bmatrix} x^2+4x-3 & 1/2\,x+1/2 \\ 0 & 1 \end{bmatrix} \in \mathsf{K}[x]^{2\times 2}.$$

We end this section by developing a simple result which will be used to precondition a matrix in Hessenberg form by using similarity transformation. Recall that the effect of premultiplying a vector B_b by a companion block C_c can be interpreted as the shift (multiplication by x) of b in the residue class ring $\mathsf{K}[x]/(c)$. The following example is for vectors of dimension three.

$$\overset{C_c}{\begin{bmatrix} & & -c_0 \\ 1 & & -c_1 \\ & 1 & -c_2 \end{bmatrix}} \overset{B_b}{\begin{bmatrix} -b_0 \\ -b_1 \\ -b_2 \end{bmatrix}} = \overset{B_{xb \bmod c}}{\begin{bmatrix} g_0 b_2 \\ -b_0 + g_1 b_2 \\ -b_1 + g_2 b_2 \end{bmatrix}}.$$

What is illustrated by the above example holds for arbitrary dimension. By premultiplying by a power of C_c we get:

Fact 9.10. $(C_c)^k B_b = B_{x^k b \bmod c}$.

The next result can be derived using this fact together with Lemma 9.2.

Lemma 9.11. *Let*

$$A = \begin{bmatrix} C_{c_1} & B_{b_{12}} \\ & C_{c_2} \end{bmatrix} \in \mathsf{K}^{d\times d}$$

with $d_i = \deg c_i$. For $t \in \mathsf{K}[x]$ with degree less than d_1, choose $v_1, v_2 \in \mathsf{K}^{d\times 1}$ as $(v_1, v_2) = (B_1, B_{t+x^{d_1+1}})$ and construct U as in (9.5). Then

$$U = \left[\begin{array}{c|c} I_{d_1} & * \\ \hline & I_{d_2} \end{array}\right] \quad \textit{and} \quad U^{-1}AU = \begin{bmatrix} C_{c_1} & B_{t_{12}} \\ & C_{c_2} \end{bmatrix}$$

where $t_{12} = b_{12} + tc_2 \bmod c_1$.

Lemma 9.11 was motivated by (Giesbrecht 1993, Section 2.3). There, it is noted that if b_{12} is divisible by c_2, then choosing $t = -b_{12}/c_2$ yields $t_{12} = 0$. This idea can be generalized as follows. Let $(g, *, a, *, *) \leftarrow \mathrm{Gcdex}(c_1, c_2)$. Then $ac_2 \equiv g \bmod c_1$. Choose $t \leftarrow \mathrm{Rem}(-a(b_{12}/g), c1)$, that is, choose t to be the the unique remainder of degree less than d_1 obtained by dividing $-a(b_{12}/g)$ by c_1. We get:

Corollary 9.12. *If g divides b_{12} then $t_{12} = 0$.*

9.2 The Zigzag form

A square matrix Z over K is in *Zigzag form* if

$$Z = \begin{bmatrix} C_{c_1} & B_{b_1} & & & & & & \\ & C_{c_2} & & & & & & \\ & B_{b_2} & C_{c_3} & B_{b_3} & & & & \\ & & & C_{c_4} & & & & \\ & & & & \ddots & & & \\ & & & & & C_{c_{k-2}} & & \\ & & & & & B_{b_{k-2}} & C_{c_{k-1}} & B_{b_{k-1}} \\ & & & & & & & C_{c_k} \end{bmatrix} \qquad (9.7)$$

with each b_i either zero or minus one, and $\deg c_i > 0$ for all i, $1 \le i \le k-1$.

Proposition 9.13. *There exists an algorithm which takes as input an $A \in F^{n\times n}$, and returns as output a $U \in F^{n\times n}$ such that $Z = UAU^{-1}$ is in Zigzag form. The cost of the algorithm is $O(n^3)$ field operations.*

We will describe an algorithm for reducing a matrix to Zigzag form using only elementary similarity transformations. The approach is essential that of Ozello (1987), who also uses similarity transformation. Let the block label L_b be defined as B_b except that coefficients of b appear in the first instead of last column.

Lemma 9.14. *An $A \in F^{n\times n}$ can be reduced to a similar matrix with the shape shown in (9.8) using $O(n \deg c)$ elementary similarity transformations where c is found during the reduction.*

$$\left[\begin{array}{c|c} C_c & L_b \\ \hline & * \end{array}\right] \qquad (9.8)$$

Furthermore, the only possibly required row operations involving row one are those which add a multiple of a different row to row one.

Proof. There are three stages to the reduction. After stage 1, 2 and 3 the work matrix has the shape shown in (9.10), (9.11) and (9.8) respectively.

1. We reduce column j of the work matrix to correct form for $j = 1, 2, \ldots, \deg c$ in succession. The algorithm is inductive and it is

sufficient to consider a single column. After the first $j-1$ columns have been reduced the work matrix can be written as

$$\left[\begin{array}{cccc|ccc}
0 & \cdots & 0 & A[1,j] & * & \cdots & * \\
1 & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots \\
 & \ddots & 0 & A[j-1,j] & * & \cdots & * \\
 & & 1 & A[j,j] & * & \cdots & * \\ \hline
 & & & A[j+1,j] & * & \cdots & * \\
 & & & \vdots & \vdots & \ddots & \vdots \\
 & & & A[n,j] & * & \cdots & *
\end{array}\right]. \tag{9.9}$$

Note that the input matrix can be written as in (9.9) with $j=1$. If the lower left block of the work matrix (9.9) is zero then the principal block is C_c with $\deg c = j$ and we are finished this stage. Otherwise, choose i with $j+1 \le i \le n$ and $A[i,j] \neq 0$. Perform the following (at most) $n+1$ row operations to reduce column j to correct form: switch rows i and j so that $A[j+1,j] \neq 0$; multiply row $j+1$ by $A[j+1,j]^{-1}$ so that $A[j+1,j]=1$; add appropriate multiples of row $j+1$ to the other $n-1$ rows to zero out entries above and below entry $A[j+1,j]$ in column j. Directly following each of these row operations we must also perform the corresponding inverse column operation on the work matrix. It is easily verified that none of these column operations will affect the entries in the first j columns of the work matrix. Since we perform this elimination process for columns $j = 1, 2, \ldots, \deg c$ this stage requires at most $(n+1)\deg c$ elementary similarity transformations.

2. At this point the work matrix can be written as

$$\left[\begin{array}{cccc|ccc}
0 & \cdots & 0 & A[1,j] & * & \cdots & * \\
1 & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots \\
 & \ddots & 0 & A[j-1,j] & * & \cdots & * \\
 & & 1 & A[j,j] & * & \cdots & * \\ \hline
 & & & & * & \cdots & * \\
 & & & & \vdots & \ddots & \vdots \\
 & & & & * & \cdots & *
\end{array}\right] \tag{9.10}$$

where $j = \deg c$. For $i = j, j-1, j-2, \ldots, 2$ in succession, zero out entries to the right of entry $A[i,j]$ in row i by adding appropriate multiples of column $i-1$ to the last $n-j$ columns. This requires at most $n-j$ column operations for each row for a total of $(n-j)(j-1)$ column operations. When working on row i, the corresponding inverse row operations that we must perform involve adding multiples of the last $n-j$ rows of the work matrix to row $i-1$. Because of the structure of the work matrix, the only effect these row operations can have is to change the last $n-j$ entries in the unprocessed row $i-1$. Thus, it is important that we perform the elimination in the order specified, that is, for row $i = j, j-1, j-2, \ldots, 2$ in succession.

3. At this point the work matrix can be written as

$$\left[\begin{array}{cccc|ccc}
0 & \cdots & 0 & A[1,j] & * & \cdots & * \\
1 & \ddots & \vdots & \vdots & & & \\
 & \ddots & 0 & A[j-1,j] & & & \\
 & & 1 & A[j,j] & & & \\ \hline
 & & & & * & \cdots & * \\
 & & & & \vdots & \ddots & \vdots \\
 & & & & * & \cdots & *
\end{array}\right] \tag{9.11}$$

where $j = \deg c$ and all entries below the first row of the upper right block are zero. If the entire upper right block is zero we are finished. Otherwise, choose k with $j+1 \le k \le n$ and $A[1,k] \neq 0$. Perform the following (at most) $n-j+1$ column operations to complete the reduction: switch columns $j+1$ and k; multiply column $j+1$ by $A[1,j+1]^{-1}$; add appropriate multiples of column $j+1$ to the last $n-j-1$ columns of the matrix to zero out entries to the right of entry $A[1,j+1]$. The inverse row operations corresponding to these column operations only affect the last $n-j$ rows of the work matrix.

□

We now outline our approach for reducing an $A \in F^{n\times n}$ to Zigzag form. The key idea can be understood by considering the first few steps. First apply the algorithm of Lemma 9.14 and transpose the work matrix so that it has the block lower triangular shape

$$\left[\begin{array}{c|c}
C_{c_1}^t & \\ \hline
L_{b_1} & *
\end{array}\right]. \tag{9.12}$$

Our goal now is to improve the structure of the trailing block labeled $*$ to have the shape shown in (9.8) whilst leaving the other blocks unchanged. The proof of Lemma 9.14 indicates which elementary similarity transformations should be applied to the trailing $n - \deg c_1$ rows and columns of the work matrix to effect the desired transformation. By Lemma 9.14, the only row operations involving row $\deg c_1$ are those which add a multiple of another row to row $\deg c_1$, this shows that the block labeled L_b will remain unchanged.

After applying the algorithm of Lemma 9.14 to the trailing block of (9.12), the work matrix can be written as

$$\left[\begin{array}{c|c|c} C_{c_1}^t & & \\ \hline L_{b_1} & C_{c_2} & L_{b_2} \\ \hline & & \\ & & * \\ & & \end{array}\right].$$

Now transpose again to get

$$\left[\begin{array}{c|c|c} C_{c_1} & L_{b_1} & \\ \hline & C_{c_2}^t & \\ \hline & L_{b_2} & \\ & & * \\ & & \end{array}\right].$$

The last step is to transpose the center block of the work matrix. The next lemma shows how to recover a similarity transform T for accomplishing this.

Lemma 9.15. *Let $C_c \in \mathsf{K}^{d\times d}$. If $T = [v|C_c v|\ldots|C_c^{d-1}v]$ where $v = B_{x^d} \in \mathsf{K}^{d\times 1}$, then $T^{-1}C_c^t T = C_c$. Furthermore, $L_b T = B_b$ for any L_b with d columns. T can be constructed in $O(d^2)$ field operations, $d = \deg c$.*

Proof. T can be constructed in the allotted time using matrix-vector products. Note that T will be unit lower anti-diagonal. This shows $L_b T = B_b$ and also that T is nonsingular. The result now follows from Lemma 9.2. ☐

We now return the the proof of Proposition 9.13.

Proof. (Of Proposition 9.13.) Initialize U to be the identity matrix and Z to be a copy of A. Perform the following steps:

Zig: Using the algorithm of Lemma 9.14 transform Z to have the shape shown in (9.8). Apply all row operations also to U.

Zag: Transpose Z and U. Apply the algorithm of Lemma 9.14 to the trailing $(n - \deg c_1) \times (n - \deg c_1)$ submatrix of Z. Apply all column operations also to U. Transpose Z and U. Apply the transformation specified by Lemma 9.15 to the companion block just found. Postmultiply the corresponding columns of U by T.

At this point

$$UAU^{-1} = Z = \left[\begin{array}{c|c|c} C_{c_1} & B_{b_1} & \\ \hline & C_{c_2} & \\ \hline & & \\ & B_{b_2} & * \\ & & \end{array}\right]. \tag{9.13}$$

Recursively apply the Zig and Zag steps on the lower right block $*$ of Z as shown in (9.13). Terminate when Z is in Zigzag form.

Computing and applying the transform T during a Zag step requires $O(n^2 d)$ field operations where d is the dimension of the companion block found during that step. The cost follows from Lemma 9.14 and by noting that $\deg c_1 + \deg c_2 + \cdots + \deg c_k = n$. ☐

9.3 From Block Diagonal to Frobenius Form

In this section we show how to transform a block diagonal matrix

$$A = \begin{bmatrix} C_{a_1} & & & \\ & C_{a_2} & & \\ & & \ddots & \\ & & & C_{a_k} \end{bmatrix} \in \mathsf{K}^{n\times n}. \tag{9.14}$$

to Frobenius form. For convenience, we allow companion blocks throughout this section to have dimension 0×0. Our result is the following.

Proposition 9.16. *There exists an algorithm which takes as input a block diagonal $A = \operatorname{diag}(C_{a_1}, C_{a_2}, \ldots, C_{a_k}) \in \mathsf{K}^{n\times n}$, and returns as output a $U \in \mathsf{K}^{n\times n}$ such that $F = UAU^{-1}$ is Frobenius form. The cost of the algorithm is $O(n^\theta)$ field operations.*

Proof. Apply following lemma to get a particular partial factorization of each of the a_i's. Let $d_i = \deg a_i$, $1 \le i \le k$.

Lemma 9.17. *There exists a set $\{b_1, b_2, \ldots, b_m\}$ of monic polynomials with*

- *$b_i \perp b_j$ for all $i \neq j$;*
- *$a_i = c_{i1}c_{i2}\cdots c_{im}$ with each c_{ij} a power of b_j, $1 \le i \le k$, $1 \le j \le m$.*

Moreover, the $(c_{ij})'s$ can be recovered in $O(n^2)$ field operations.

Proof. See (Bach and Shallit 1996, Section 4.8). □

The next lemma shows how to "split" a companion block C_{a_i} into m companion blocks based on the factorization $a_i = c_{i1}c_{i2}\cdots c_{im}$. A similar idea is used by Kaltofen *et al.* (1990) and Giesbrecht (1994). Let $\mathrm{col}(I_{d_i}, j)$ denote column j of the $d \times d$ identity.

Lemma 9.18. *Let $A_i = \mathrm{diag}(C_{c_{i1}}, C_{c_{i2}}, \cdots, C_{c_{im}})$ and set*

$$U_i = \left[\; v_i \;\middle|\; A_i v_i \;\middle|\; \cdots \;\middle|\; A_i^{d_i-1} v_i \;\right] \in \mathsf{K}^{d_i \times d_i}$$

where $v_i = \sum_{1\le j\le m} \mathrm{col}(I_{d_i}, 1 + c_{i1} + \cdots + c_{ij}) \in \mathsf{K}^{d_i \times 1}$. Then $U_i^{-1}A_iU_i = C_{a_i}$. The matrix U_i can be recovered in in $O(d_i^2)$ field operations.

Proof. The minimal polynomial of A_i is given by $\mathrm{lcm}(c_{i1}, c_{i2}, \ldots, c_{im}) = c_i$. This shows that C_{c_i} is the Frobenius form of A_i. It follows easily that v_i will be a cyclic vector for A_i. Since A_i contains fewer than $2d_i$ entries, U_i can be constructed in the allotted time using matrix-vector products. □

Let $V = \mathrm{diag}(U_1, U_2, \ldots, U_m) \in \mathsf{K}^{n\times n}$ where each U_i is constructed as above. Then VAV^{-1} equals

$$\mathrm{diag}(\overbrace{C_{c_{11}}, C_{c_{12}}, \cdots, C_{c_{1m}}}^{C_{a_1}}, \overbrace{C_{c_{21}}, C_{c_{22}}, \cdots, C_{c_{2m}}}^{C_{a_2}}, \cdots, \overbrace{C_{c_{k1}}, C_{c_{k2}}, \cdots, C_{c_{km}}}^{C_{a_k}}).$$

For $1 \le j \le m$, let $(c_{\bar{1}j}, c_{\bar{2}j}, \ldots, c_{\bar{k}j})$ be the list $(c_{1j}, c_{2j}, \ldots, c_{kj})$ sorted by degree (increasing). Construct a permutation matrix $P \in \mathsf{K}^{n\times n}$ based on these sorts such that $PVAV^{-1}P^{-1}$ can be written as

$$\mathrm{diag}(\overbrace{C_{c_{\bar{1}\bar{1}}}, C_{c_{\bar{1}\bar{2}}}, \cdots, C_{c_{\bar{1}\bar{m}}}}^{C_{f_1}}, \overbrace{C_{c_{\bar{2}\bar{1}}}, C_{c_{\bar{2}\bar{2}}}, \cdots, C_{c_{\bar{2}\bar{m}}}}^{C_{f_2}}, \cdots, \overbrace{C_{c_{\bar{k}\bar{1}}}, C_{c_{\bar{k}\bar{2}}}, \cdots, C_{c_{\bar{k}\bar{m}}}}^{C_{f_k}})$$

Here, f_i denotes the product $c_{i\bar{1}}c_{i\bar{2}}\cdots c_{i\bar{m}}$. By construction, we have f_i dividing f_{i+1} for $1 \le i \le k$. Now apply Lemma 9.18 to construct a block diagonal W such that $W^{-1}PVAV^{-1}P^{-1}W = \mathrm{diag}(C_{f_1}, C_{f_2}, \ldots, C_{f_m})$, the Frobenius form of A, and set $U = W^{-1}PV$. □

9.4 From Zigzag to Frobenius Form

Proposition 9.19. *There exists an algorithm that takes as input a $Z \in \mathsf{K}^{n\times n}$ in Zigzag form, and returns as output a $U \in \mathsf{K}^{n\times n}$ such that $F = UAU^{-1}$ is in Frobenius form. The cost of the algorithm is $O(n^\theta(\log n))$ field operations, or $O(n^3)$ field operations assuming standard matrix multiplication.*

Proof. Let $f(n)$ be the number of field operations required to compute a U over K which transforms a Zigzag form or the transpose of a Zigzag form to Frobenius form. Clearly, $f(0) = 0$. (Every null matrix is in Frobenius form.) Now assume $n > 0$. The result will follow we can show that

$$f(n) \le f(n_1) + f(n_2) + O(n^\theta)$$

for some choice of n_1 and n_2 both $\le n/2$.

Let Z be in Zigzag form with Frobenius form F. Using Lemma 9.15 we can construct a T over K such that $T^{-1}FT = F^t$ in $O(n^\theta)$ field operations. This shows it will be sufficient to solve one of the following: transform Z to F; transform Z^t to F; transform Z to F^t; transform Z^t to F^t. We now describe a seven step algorithm for solving one of these problems. Each step computes an invertible $n \times n$ matrix U and transforms the work matrix Z inplace as follows: $Z \to U^{-1}ZU$.

We begin with Z in Zigzag form as in (9.7).

1. At least one of Z or Z^t can be partitioned as

$$\left[\begin{array}{c|c|c} Z_1 & * & \\ \hline & * & \\ \hline & * & Z_2 \end{array}\right]$$

   where both Z_1 and Z_2 are square with dimension $\le n/2$ and the center block is a either a companion matrix or transpose thereof. Z_1 and/or Z_2 may be the null matrix but the center block should have positive dimension.

If the middle block is the transpose of a companion matrix, apply Lemma 9.15 to compute a similarity transform which transposes it. The transform can be constructed and applied in the allotted time to get

$$\left[\begin{array}{c|c|c} Z_1 & B_* & \\ \hline & C_* & \\ \hline & B_* & Z_2 \end{array}\right].$$

where each block labeled B_* zero except for possibly a single nonzero entry in the last column.

2. Recursively transform the principal and trailing block to Frobenius form. The work matrix can now be written as

$$\left[\begin{array}{c|c|c} F_1 & B_* & \\ \hline & C & \\ \hline & B_* & F_2 \end{array}\right].$$

where each block labeled B_* is zero except for possibly the last column.

Applying a similarity permutation transform which shuffles the blocks so that

$$\left[\begin{array}{c|c|c} F_1 & & * \\ \hline & F_2 & * \\ \hline & & C \end{array}\right].$$

3. Transform the principal block $\operatorname{diag}(F_1, F_2)$ to Frobenius form. By Proposition 9.16 this costs $O(n^\theta)$ field operations.

We now have

$$\begin{bmatrix} C_{a_1} & & & & & B_{b_1} \\ & C_{a_2} & & & & B_{b_2} \\ & & C_{a_3} & & & B_{b_3} \\ & & & \ddots & & \vdots \\ & & & & C_{a_{k-1}} & B_{b_{k-1}} \\ & & & & & C_{a_k} \end{bmatrix}. \tag{9.15}$$

4. We now want to "condition" the work matrix (9.15), modifying only the b_*'s, so that

$$(a_j, b_j) = (a_j, b_j, \ldots, b_k) \quad \text{for} \quad l < j < k \tag{9.16}$$

holds for $j = 1$. This is accomplished by applying the transform (U^{-1}, U) where

$$U = \begin{bmatrix} I_{d_1} & * & & & & \\ & I_{d_2} & * & & & \\ & & I_{d_3} & * & & \\ & & & \ddots & \ddots & \\ & & & & I_{d_{k-1}} & * \\ & & & & & I_{d_k} \end{bmatrix}$$

is computed using the Stab function together with the method of Lemma 9.11. The off-diagonal blocks of U are recovered in succession, from last to first, so that (9.16) holds for $j = k-1, k-2, \ldots, 1$. The method is similar to step seven of the algorithm supporting Proposition 7.12.

The work matrix can now be written as in (9.15) and satisfies (9.16).

5. Apply a similarity permutation transform to shuffle the blocks so that

$$\begin{bmatrix} C_{a_k} & & & & & \\ B_{b_{k-1}} & C_{a_2} & & & & \\ B_{b_{k-2}} & & C_{a_3} & & & \\ \vdots & & & \ddots & & \\ B_{b_2} & & & & C_{a_1} & \\ B_{b_1} & & & & & C_{a_1} \end{bmatrix}. \tag{9.17}$$

Let A be the matrix shown in (9.17). Use the method of Fact 9.4 to construct a U such that $U^{-1}AU$ is in shifted Hessenberg form (recall Definition 9.3). Applying the transform (U^{-1}, U) to get

$$\begin{bmatrix} C_{s_l} & B_{b_*} & \cdots & B_{b_*} \\ & C_{s_{l-1}} & & B_{b_*} \\ & & \ddots & \vdots \\ & & & C_{s_1} \end{bmatrix}.$$

where $s_l, s_{l-1}, \ldots, s_l$ are those diagonal entries in the right Hermite form of $xI - A \in \mathsf{K}[x]^{n \times n}$ which have positive degree (cf. Theorem 9.5).

By Lemma 9.5, together with an argument analogous to that used in step eight of the algorithm supporting Proposition 7.12, we may conclude that $\mathrm{diag}(C_{s_1}, C_{s_2}, \ldots, C_{s_l})$ is in Frobenius form and that s_j divides b_{ij} for $1 \le i < j \le n$.

6. Transpose to get

$$\begin{bmatrix} C_{s_l}^t & & & \\ B_{b_*}^t & C_{s_{l-1}}^t & & \\ \vdots & & \ddots & \\ B_{b_*}^t & B_{b_*}^t & \cdots & C_{s_1}^t \end{bmatrix}. \tag{9.18}$$

Now let A be the matrix shown in (9.18). Because of the divisibility properties of the s_*'s and b_*'s established in the previous step, the right Hermite form of $xI_n - A \in \mathsf{K}[x]^{n\times n}$ will be such that all off-diagonal entries in those columns with a positive degree diagonal entry will be zero. Use the method of Fact 9.4 to construct a U so that $U^{-1}AU$ is in shifted Hessenberg form. Apply the transform (U^{-1}, U) to get $\mathrm{diag}(C_{s_l}, C_{s_{l-1}}, \ldots, C_{s_1})$.

7. Apply a permutation similarity transform to get $\mathrm{diag}(C_{s_1}, C_{s_2}, \ldots, C_{s_l})$, a matrix in Frobenius form. We are finished.

By augmenting the input matrix Z with identity matrices as shown below, we can automatically record all similarity transforms applied to A and recover a final transform (U, U^{-1}) such that $UZU^{-1} = F$.

$$\left[\begin{array}{c|c} Z & I_n \\ \hline I_n & \end{array}\right] \to \left[\begin{array}{c|c} F & U \\ \hline U^{-1} & \end{array}\right].$$

The cost of computing and applying the similarity transforms in each step is bounded by $O(n^\theta(\log n))$ basic operations, or $O(n^3)$ field operations assuming standard matrix multiplication. The result follows. □

We get the following as a corollary of the last three sections.

Corollary 9.20. *Let $A \in \mathsf{K}^{n\times n}$. A $U \in \mathsf{K}^{n\times n}$ such that $F = U^{-1}AU$ is in Frobenius form can be computed in $O(n^3)$ field operations.*

9.5 Smith Form over a Valuation Ring

The algorithm of this section is inspired by Lübeck (1998).

This section is concerned with the problem of computing the Smith form of a matrix over a certain type of PIR. In particular, let

$$\mathsf{R} = \mathsf{K}[x]/(p^k) \text{ where } p \in \mathsf{K}[x] \text{ is irreducible of degree } d$$

We may assume that elements of R are represented as polynomials from $\mathsf{K}[x]$ with degree strictly less than $\deg p^k = kd$. We need to define both a complete set of associates $\mathcal{A}(\mathsf{R})$ of R and also a complete set of residues modulo each element of $\mathcal{A}(\mathsf{R})$, cf. Section 1.1. We choose

$$\mathcal{A}(\mathsf{R}) = \{0, p, p^2, \ldots, p^{k-1}\} \text{ and } \mathcal{R}(\mathsf{R}, p^i) = \{r \in \mathsf{K}[x] \mid \deg r < \deg p^i\}$$

It is well known that each of the basic operations {Arith, Gcdex, Ass, Rem, Ann} can be accomplshed using $O((kd)^2)$ field operations; this assumes standard polynomial arithmetic.

In this section, we adopt the convention that Smith forms of square matrices are written as $\mathrm{diag}(0, 0, \ldots, 0, s_r, \ldots, s_2, s_1)$ where $s_i | s_{i+1}$ for $1 \le i < r$. Hermite form will always mean right Hermite form, as defined on page 147 before Lemma 9.5. Let $A \in \mathsf{R}^{m\times m}$. Note that the diagonal entries of the Smith form of A over R and also of any Hermite form of A will be powers of p. On the one hand, since R is a PIR (i.e. not necessarilly a PID) the Hermite form might not be a canonical form. On the other hand, it follows from Section 8.1 that there exists a unit lower triangular conditioning matrix C such that every Hermite of CA will be in triangular Smith form, that is, have the same diagonal entries as the Smith form. In fact, the R that we consider in this section is a *valuation ring*: for every two elements $a, b \in \mathsf{R}$ either a divides b or b divides a. For this R it turns out that we may choose C to be a permutation matrix. We get the following result.

Proposition 9.21. *Let $A \in \mathsf{R}^{m\times m}$ be given, $\mathsf{R} = \mathsf{K}[x]/(p^k)$ where p is irreducible of degree d. Then there exists a permutation matrix C such that every Hermite form of CA is in triangular Smith form. Such a C can be recovered in $O(m^\theta(kd)^2)$ field operations using standard polynomial arithmetic.*

Remark: We call the matrix C of Proposition 9.21 a *Smith permutation.*

Proof. Initialize $C = I_m$, $B = A$ and $l = 0$. Perform the following steps for $i = 1, 2, \ldots, k$. Each steps modifies C and B. The paramater l is monotonically nondecreasing. At the start of each step, the trailing l

rows of B are linearly independant over $\mathcal{R}(\mathsf{R},p)$, and the trailing l rows of B and c will not be changed,

1. Let $\bar{B}$ be a copy of B with entries reduced modulo p.
2. Find the rank r of $\bar{B}$ over $\mathcal{R}(\mathsf{R},p)$. At the same time, find
   - a permutation matrix
   $$P = \left[\begin{array}{c|c} * & \\ \hline & I_l \end{array}\right]$$
   such that $P\bar{B}$ has last r rows with rank r, and
   - an upper triangular
   $$U = \left[\begin{array}{c|c} I & * \\ \hline & I_l \end{array}\right] \in \mathcal{R}(\mathsf{R},p)^{m\times m}$$
   such that $UP\bar{B}$ has first r rows all divisible by p.
3. Replace C with PC. Replace B with the matrix obtained from UPB by dividing each entry in the first $n-r$ rows by p.

It is easy to verify that upon completion C will indeed be a Smith permutation for A. The cost of step 1 is $O(m^2(kd^2))$ field operations, since reducing a degree $dk-1$ polynomial modulo a degree $d-1$ polynomial can be accomplished in $O(kd\cdot d)$ field operations. Step 2 can be accomplished in $O(m^\theta d^2)$ field operations by computing a Gauss transform for $\bar{B}$ over $\mathcal{R}(\mathsf{R},p)$. Step 3 costs $O(m^\theta(d\cdot kd))$ field operations, since U and B have entries bounded in degree by d and kd respectively. □

An Extension

Now consider the situation where

$$\mathsf{R} = \mathsf{K}[x]/(f^k) \text{ where } f \in \mathsf{K}[x] \text{ has degree } d,$$

that is, where f might not be irreducible. We would like to compute a Smith permutation for an $A \in \mathsf{R}^{m\times m}$. Unfortunately, this might not be possible, since R is not a valuation ring; one reason is that entries in the Smith form of A may contain only partial factors of f. But initialize $p = f$. What happens if we pretend that p is irreducible and we run the algorithm supporting Proposition 9.21? There is precisely one place

where the algorithm might break down — during the computaton of the rank r and permuation P in step 1. Since these are computed by Gaussian elminination, we might get an error when attempting to divide by a nonzero (but not necessarily invertible) element of R. If such a "side exit" error happens, we can produce (in $O(d^2)$ field operations) a nontrivial factorization for p: *either* $p = \bar{p}^{\bar{e}}$ with $\bar{e} > 1$ *or* $p = \bar{p}\hat{p}$ with, say, $\deg \bar{p} \geq \deg \hat{p}$ and $\bar{p} \perp \hat{p}$. The idea is to reduce all entries in the work matrices U and $\bar{B}$ modulo $\bar{p}$ and continue with the Gaussian elmination. By keeping track of all the factorizations that occur we can produce the following.

- A gcd-free basis $\{p, f_1, f_2, \ldots, f_k\}$ for f, say $f = p^e f_1^{e_1} f_2^{e_2} \cdots f_k^{e_k}$, such that
  - $\deg p^e f_1^{e_1} f_2^{e_2} \cdots f_i^{e_i} \geq \deg f_{i+1}^{e_{i+1}}$ for $0 \leq i < k$, and
  - each diagonal entry of the Smith form of A over R can be written as $s\bar{s}$ where s is a power of p and $\bar{s} \perp p$.
- A Smith permutation C for A over $\mathsf{R}/(p^k) \cong \mathsf{K}[x]/(p^k)$.

The cost analysis is straightforward. As the "work modulus" p is decreasing in degree, the cost of all the steps of the algorithm supporting Proposition 9.21 decreases. It remains only to bound the cost of reducing entries in the work matrices when the modulus changes. But for some monontonically decreasing degree sequence $d = d_0 > d_1 > d_2 > \cdots > d_{k+1}$, the total cost of these reductions is bounded by

$$O(m^2 \cdot (d_0(d_0 - d_1) + d_1(d_1 - d_2) + \cdots)) = O(m^2 d^2)$$

field operations. (Recall that computing the remainder of a degree $d_1 - 1$ polynomial with respect to a degree $d_2 - 1$ polynomial costs $O(d_0(d_0 - d_1))$ field operations when using standard arithmetic.) We get:

Corollary 9.22. *Let $A \in \mathsf{K}[x]/(f^k)$ be given. A factorization of f and a Smith permutation C as described above can be recovered in $O(m^\theta(kd)^2)$ field operations using standard polynomial arithmetic.*

9.6 Local Smith Forms

Let $A \in \mathsf{K}[x]^{m\times m}$ be given. We abuse notation very slightly, and sometimes consider A to be over a residue class ring $\mathsf{K}[x]/(p^k)$ for a given

$p \in \mathsf{K}[x]$, $k \in \mathbb{N}$. Assume for now that p is irreducible. Then each diagonal entry of the Smith form of A over $\mathsf{K}[x]$ can be written as $s\bar{s}$ where s is a power of p and $\bar{s} \perp p$. If n is an upper bound on the highest power of p in any diagonal entry of the Smith form of A, then we can recover the Smith form of A locally at p by computing the Smith form of A over $\mathsf{K}[x]/(p^{n+1})$.

This section is concerned with the problem of computing a Smith permutation. Specifically, we are given as input

- an irreducible $p \in \mathsf{K}[x]$ of degree d, and
- a nonsingular Hermite form $A \in \mathsf{K}[x]^{m\times m}$ which has each diagonal entry a multiple of p.

Our goal is to produce the following:

- A Smith permutation C for A over $\mathsf{K}[x]/(p^{n+1})$, where $n = \deg\det A$.

Thus, the problem we consider here reduces to the problem we dealt with in the previous section. But if we apply Proposition 9.21 directly, we get a complexity of $O(m^\theta(nd)^2)$ field operations. Because, by assumption, p divides each diagonal entry of A, we have $m \le n/d$. Using $m \le n/d$ and $\theta > 2$ leads to the complexity bound of $O(n^{\theta+2})$ field operations for recovering C. In this section, we establish the following result.

Proposition 9.23. *Let p be irreducible of degree p and $A \in \mathsf{K}[x]^{m\times m}$ be in Hermite form with each diagonal entry a multiple of p. Suppose $n = \deg\det A$. Then a Smith permutaion C for A over $\mathsf{K}[x]/(p^{n+1})$ can be recovered in $O(n^\theta(\log n)(\log\log n))$ field operations using standard polynomial arithmetic.*

Proof. Initialize $C = I_m$. The key idea is to work in stages for $k = 1, 2, \ldots$ Each stage applies row permutations to C to achieve that C is a Smith permutation for A over $\mathsf{K}[x]/(p^*)$ for higher and higher powers of p. After stage $k-1$ and at the start of stage k, the matrix C will be a Smith permutation for A over $\mathsf{K}[x]/(p^{\phi(k-1)})$ (we define ϕ below) and the Hermite form of CA can be written as

$$\left[\begin{array}{c|ccccc} p^{\phi(k-1)}T & * & * & \cdots & * \\ \hline & H_{k-1} & * & \cdots & * \\ & & H_{k-2} & & * \\ & & & \ddots & \vdots \\ & & & & H_1 \end{array}\right], \tag{9.19}$$

where

- all entries in the principal block $p^{\phi(k-1)}T$ are divisible by $p^{\phi(k-1)}$, and
- the work matrix is in triangular Smith form over $\mathsf{K}[x]/(p^{\phi(k-1)})$.

Subsequent stages need only work with the smaller dimensional submatrix T. The function ϕ determines the amount of progress we make at each stage. What is important for correctness of the algorithm is that ϕ is monotonically increasing, that is $\phi(k) > \phi(k-1)$ for $k \in \mathbb{N}$. In fact, if we choose $\phi(k) = k+1$, then $\phi(k) - \phi(k-1) = 1$ and the algorithm will require at most n stages. If we chose $\phi(k) = 2^k$, the algorithm will require at most $\lceil \log_2 n\rceil$ stages. We define

$$\phi(k) = \begin{cases} 1 & \text{if } k = 0 \\ \lceil 2^{((\theta/2)^k)} \rceil & \text{if } k > 0 \end{cases}$$

The following is easily verified:

Lemma 9.24. *With the following definition of ϕ, we have*

- *$\phi(k) < (\phi(k-1))^{\theta/2} + 1$ for $k > 0$, and*
- *if $k > \lceil \log_2(\log_2(n))/((\log_2\theta) - 1)\rceil$, then $\phi(k) > n$*

In other words, with this definition of ϕ, and with the assumption that $\theta > 2$, the algorithm will require only $O(\log\log n)$ stages. Now we give the algorithm. Initialize $k = 1$, $C = I_m$ and $T = A$. Do the following.

1. Let B be a copy of T. Reduce entries in B modulo $p^{\phi(k)-\phi(k-1)}$. Note: B will have dimension less than $\lfloor n/(\phi(k-1)d)\rfloor$.
2. Use Proposition 9.21 to recover a Smith permutation L for B over $\mathsf{K}[x]/(p^{\phi(k)-\phi(k-1)})$.
3. Update C as follows:
$$C \longrightarrow \left[\begin{array}{c|c} L & \\ \hline & I \end{array}\right] C.$$
4. Use Proposition 9.8 to compute the Hermite form H of LB over $\mathsf{K}[x]$. Replace T by $1/p^{\phi(k)}$ times the principle square submatrix of H of maximal dimension such that all entries in the submatrix are divisible by $p^{\phi(k)}$.

5. If T is the null matrix then quit.

6. Increment k and go to step 1.

Now we turn our attention to the complexity. From Lemma 9.24 we know that we jump back up to step 1 at most $O(\log \log n)$ times. The result will follow if we show that each of the other steps are bounded by $O(n^\theta(\log n))$ field operations.

Step 4 costs $O(n^\theta(\log n))$ field operations. Now consider step 2. Each application of Proposition 9.21 requires

$$O\left(\left(\frac{n}{\phi(k-1)d}\right)^\theta \cdot ((\phi(k)-\phi(k-1))d)^2\right)$$

field operations. Since $\phi(k)-\phi(k-1) < \phi(k)$, and using Lemma 9.24, we may substitute $\phi(k)-\phi(k-1) \to (\phi(k-1))^{\theta/2}$. Since $\theta > 2$ the overall bound becomes $O(n^\theta)$. Now consider step 1. The sum of the degrees of the diagonal entris in T is less than n. By amortizing the cost of reducing the entries in each row of B, we get the bound

$$O\left(\left(\frac{n}{\phi(k-1)d}\right) n \cdot (\phi(k)-\phi(k-1))d\right)$$

field operations. We have already seen, while bounding step 2, that this is less than $O(n^\theta)$. □

An Extension

Now consider that we are not given an irreducible p, but rather

- an $f \in \mathsf{K}[x]$ of degree d, and
- a nonsingular Hermite form $A \in \mathsf{K}[x]^{m\times m}$ which has each diagonal entry a multiple of f.

Now our goal is to produce the following:

- A gcd-free basis $\{p, f_1, f_2, \ldots, f_k\}$ for f, say $f = p^e f_1^{e_1} f_2^{e_2} \cdots f_k^{e_k}$, such that
  - $\deg p^e f_1^{e_1} f_2^{e_2} \cdots f_i^{e_i} \geq \deg f_{i+1}^{e_{i+1}}$ for $0 \leq i < k$, and
  - each diagonal entry of the Smith form of A over $\mathsf{K}[x]$ can be written as $s\bar{s}$ where s is a power of p and $\bar{s} \perp p$.

- A Smith permutation C for A over $\mathsf{K}[x]/(p^{n+1})$, where $n = \deg\det A$.

Using Corollary 9.22, we get the following:

Corollary 9.25. *Let $A \in \mathsf{K}[x]^{m\times m}$ be nonsingular, in Hermite form, and with each diagonal entry a multiple of f. Suppose that* $\deg\det A = n$. *Then a partial factorization of f together with a Smith conditioner C for A as described above can be recovered in $O(n^\theta(\log n)(\log\log n))$ field operations.*

9.7 The Fast Algorithm

Let $A \in \mathsf{K}^{n\times n}$. We give a recursive algorithm for computing the Frobenius form of A. The approach is a standard divide and conquer paradigm: four recursive calls on matrices with dimension at most $\lfloor n/2 \rfloor$. First we use two recursive calls and some additonal massaging to arrive at a Hessenberg form T which has each diagonal entry a power of f, for some $f \in \mathsf{K}[x]$. We will be finished if we can recover the Frobenius form of T. Because of the special shape of T, it will be sufficient to recover the Frobenius form locally for some $p \in \mathsf{K}[x]$, a factor of f. The other components of T (corresponding to factors of the characteristic polynomial of T which are relatively prime to p) will be split off from T and incorporated into the final two recursive calls of the algorithm. Before the splitting of T begins, the final two recursive calls have dimension n_1 and n_2 respectivley, $n_1, n_2 \leq \lfloor n/2 \rfloor$ and $n_1 + t + n_2 = n$ where t is the dimension of T. We need to ensure that the size of the final two recursive calls remains bounded by $n/2$. Suppose that we split the block T into two blocks of dimenision t_1 and t_2 respectively, $t_2 \leq t_1$. Then

Lemma 9.26. *Either $n_1 + t_2 \leq \lfloor n/2 \rfloor$ or $n_2 + t_2 \leq \lfloor n/2 \rfloor$.*

Now we give the algorithm.

1. Use Fact 9.4 to recover a shifted Hessenberg form of A. Partition as

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

where the principal and trailing blocks are in shifted Hessenberg form with dimension n_1 and n_2 respectively, $n_1, n_2 \leq n/2$. Recursively transform these blocks to Frobenius form.

2. Note: the principal $n-n_2$ submatrix of the work matrix is the submatrix demarked by the double lines. Apply steps 3–7 of Proposition 9.19 to transform this submatrix to Frobenius form. The work matrix now looks like
$$\left[\begin{array}{c|c} F_* & * \\ \hline & F_* \end{array}\right]$$

3. Compute a gcd-free basis $\{b_1, b_2, \ldots, b_m\}$ for the set of all diagonal companion blocks in the work matrix. Apply Lemma 9.18 to the principal and trailing submatrices to split according the gcd-free basis. After applying a similarity permutation the work matrix can be written as
$$\left[\begin{array}{cccc|cccc} *_1 & & & & * & * & \cdots & * \\ & *_2 & & & * & * & & * \\ & & \ddots & & \vdots & \vdots & \ddots & \vdots \\ & & & *_m & * & * & \cdots & * \\ \hline & & & & *_1 & & & \\ & & & & & *_2 & & \\ & & & & & & \ddots & \\ & & & & & & & *_m \end{array}\right]$$
where $*_i$ denotes a (possibly null) matrix in Frobenius form which has each diagonal entry a power of b_i.

4. Because
   - $b_i \perp b_j$ for $i \neq j$,
   - the observation of Corrollary 9.12,

   we can construct, using Fact 9.4, a transistion matrix that transforms the work matrix to look like
$$\left[\begin{array}{cccc|cccc} *_1 & & & & * & & & \\ & *_2 & & & & * & & \\ & & \ddots & & & & \ddots & \\ & & & *_m & & & & * \\ \hline & & & & *_1 & & & \\ & & & & & *_2 & & \\ & & & & & & \ddots & \\ & & & & & & & *_m \end{array}\right].$$

5. Apply a similarity permutation to achieve
$$\left[\begin{array}{cc|cc|c|cc} *_1 & * & & & & & \\ & *_1 & & & & & \\ \hline & & *_2 & * & & & \\ & & & *_2 & & & \\ \hline & & & & \ddots & & \\ \hline & & & & & *_m & * \\ & & & & & & *_m \end{array}\right]. \tag{9.20}$$
.

6. Partition the work matrix as
$$\left[\begin{array}{c|c|c} A_1 & & \\ \hline & T & \\ \hline & & A_2 \end{array}\right] \tag{9.21}$$
such that the principal and trailing block have dimension n_1 and n_2 respectively, where $n_1, n_2 \leq n/2$ are chosen maximal. Note that T will be one of the blocks of the matrix in (9.20), say
$$T = \left[\begin{array}{c|c} *_f & * \\ \hline & *_f \end{array}\right]$$
with each of $*_f$ in Frobenius form, each diagonal a power of f.

7. Denote by $T(x) \in \mathsf{K}[x]^{m\times m}$ the Hermite form which corresponds to the Hessenberg form T. Now apply the algorithm of the previous section to recover from $T(x)$ and f the following:
   - A gcd-free basis $\{p, f_1, f_2, \ldots, f_k\}$ for f, say $f = p^e f_1^{e_1} f_2^{e_2} \cdots f_k^{e_k}$, such that
     - $\deg p^e f_1^{e_1} f_2^{e_2} \cdots f_i^{e_i} \geq \deg f_{i+1}^{e_{i+1}}$ for $0 \leq i < k$, and
     - each diagonal entry of the Smith form of $T(x)$ over $\mathsf{K}[x]$ can be written as $s\bar{s}$ where s is a power of p and $\bar{s} \perp p$.
   - A Smith permutation L for $T(x)$ over $\mathsf{K}[x]/(p^{n+1})$.

   Split the block T according to the gcd-free basis $\{p, f_1, f_2, \ldots, f_k\}$ using the technique of steps 3–5 above. According to the Smith permutation L, apply a similarity permutation to the block T_p

corrsponding to p. Transform T_p to Hessenberg form and then apply step 6 of the algorithm supporting Proposition 9.19 to transform T_p to Frobenius form.

After the splitting of T is complete, the work matrix can be written (up to a similarity permuation) as in

$$\left[\begin{array}{cc|c|cc} A_1 & & & & \\ & \bar{A}_1 & & & \\ \hline & & T_p & & \\ \hline & & & A_2 & \\ & & & & A_2 \end{array}\right]$$

where T_p is block diagonal and the submatrices $\bar{A}_*$ contain the other blocks split off from T, cf. (9.21). The degree conditions on the partial factorization $p^e f_1^{e_1} f_2^{e_2} \cdots f_k^{e_k}$ of f ensure that we can allocate these other blocks (those split off from T) between $\bar{A}_1$ and $\bar{A}_2$ in such a way that the principal and trailing submatrices of the work matrix still have dimension $\leq n/2$, see Lemma 9.26.

8. Recursively transform the principal and trailing block to Frobenius form. Now the work matrix is block diagonal.

9. Complete the transformation using Proposition 9.16.

Corollary 9.25 shows that step 7 can be accomplished in $O(n^\theta(\log n)(\log\log n))$ field operations. We get the following result:

Proposition 9.27. *Let $A \in \mathsf{K}^{n\times n}$. A $U \in \mathsf{K}^{n\times n}$ such that $F = U^{-1}AU$ is in Frobenius form can be computed in $O(n^\theta(\log n)(\log\log n))$ field operations.*

Chapter 10

Conclusions

We developed and analysed generic algorithms for the computation of various matrix canonical forms. We also applied the generic algorithms to get algorithms for solving various problems with integer input matrices. The analysis of the generic algorithms was under the algebraic complexity model — we bounded the number of required basic operations. The analysis of the integer matrix algorithms was under the bit complexity model — we took into account the size of operands and bounded the number of required bit operations. This chapter make some comments and poses some questions with respect to each of these areas.

10.1 Algebraic Complexity

Much of our effort was devoted to showing that the problems of computing the Howell and Smith form are essentially no more difficult over a PIR than than over a field. A precise statement and analysis of the algorithms over a PIR was made possible by defining a small collection of *basic operations* from the ring. In particular, introducing the basic operation Stab made it possible to give an algorithm for diagonalization in the first place. We remark that the algorithms we have given are applicable over any commutative ring with identity that supports the required basic operations. For example, the required operations for triangularization are $\{\text{Arith}, \text{Gcdex}\}$ while those for diagonalization are $\{\text{Arith}, \text{Gcdex}, \text{Stab}\}$. A ring which support only these operations might not be a PIR.

The comments we make next are of a complexity theoretic nature.

Notwithstanding our focus on more general rings, here it will be appropriate to consider the well understood case of matrices over a field.

The Question of Optimality

In this subsection, we restrict ourselves to the special case of square input matrix over an infinite field K. Recall that we are working over an arithmetic RAM. All complexity bounds will be in terms of number of operations of type Arith from K. If we don't specify otherwise, all algorithms and problems will involve an $n \times n$ input matrix over K. Let P1 and P2 be such problems. Then:

- P2 is *reducible* to P1 if, whenever we have an algorithm for P1 that has cost bounded by $t(n)$, then we have[1] also an algorithm for problem P2 that has cost bounded by $O(t(n))$.

- P2 is *essentially reducible* to P1 if, whenever we have an algorithm for P1 that has cost bounded by $t(n)$, then we have also an algorithm for P2 that has cost bounded by $O\tilde{~}(t(n))$.

Let MatMult be the problem of matrix multiplication. More precisely, let MatMult(n) be the problem of multiplying together two $n \times n$ matrices. Over a field, the Howell and Smith form resolve to the Gauss Jordan and rank normal form. The problems of computing these forms over a field had already been reduced to MatMult. The result for the Gauss Jordan form is given Keller-Gehrig (1985), and that for the rank normal form follows from LSP-decomposition algorithm of Ibarra *et al.* (1982). A treatment can be found in Bürgisser *et al.* (1996), Chapter 16. We have shown here that the problems of computing the Frobenius form is essentially reducible to MatMult; this has been already established by Giesbrecht (1993) using randomization.

The question we want to consider here is the opposite:

> Is MatMult essentially reducible to the problem of computing each of these canonical forms?

For nonsingular input over a field, the problem of computing a transform for the Howell, Hermite or Gauss Jordan form coincides with Inverse — matrix inversion. Winograd (1970) has shown that MatMult is reducible to Inverse. The proof is shown in Figure 10.1. We may

[1] The distinction between "we have" and "there exists" is important.

$$\begin{bmatrix} I_n & A & \\ & I_n & B \\ & & I_n \end{bmatrix}^{-1} = \begin{bmatrix} I_n & -A & AB \\ & I_n & -B \\ & & I_n \end{bmatrix},$$

Figure 10.1: Reduction of MatMult(n) to Inverse$(3n)$.

conclude that the algorithms we have presented here for these echelon forms are asymptotically optimal.

The problem Det — computing the determinant — is immediately reducible to that of computing the Frobenius form. Now consider the Smith form. Many problems are reducible to that of computing transforms for the Smith form of a square input matrix A. We mention three. The first is Basis — computing a nullspace basis for A. The second is Trace — computing the sum of the diagonal entries in the inverse of a nonsingular A. The third is LinSys — return $A^{-1}b$ for a given vector b. The first reduction is obvious (the last $n-r$ rows of the left transform are the nullspace basis). To see the second two, note that if $UAV = I$, then $VU = A^{-1}$.

Unfortunately, we don't know of a deterministic reduction of MatMult to any of Det, Basis, Trace or LinSys. But we do know of a randomized reduction from MatMult to Det. Giesbrecht (1993) shows — using techniques from Strassen (1973) and Baur and Strassen (1983) — that if we have a Monte Carlo probabilistic algorithm for Det that has cost bounded by $t(n)$, then we have also a Monte Carlo probablistic algorithm MatMult that has cost bounded by $O(t(n))$.

We can also ask a question of a different nature. Is MatMult computationally more difficult than Basis or Det? Here the emphasis has shifted from algorithms for the problems to the problems themselves. It turns out that the arithmetic RAM is not a suitable computational model to ask such questions. For example, maybe no single algorithm achieves the "best" asymptotic cost for a given problem, but rather a sequence of essentially different algorithms are required as $n \to \infty$. A suitable model over which to ask such questions is that of *computation trees* where we have the notion of *exponent* of a problem. We don't define these terms here, but refer the reader to the text by Bürgisser *et al.* (1996). Over the model of computation trees, it has been established that the exponents of the problems Det, Basis and MatMult coincide. In other words, from a computational point of view, these problems are asymptotically equivalent. The result for Basis is due to Bürgisser *et al.*

(1991). The result for both BASIS and DET can be found in Bürgisser *et al.* (1996), Chapter 16.

Logarithmic Factors

Our algorithm for Smith form requires $O(n^\theta)$ basic operations, but the method we propose for producing transforms requires $O(n^\theta(\log n))$ basic operations.

> Is the $\log n$ factor essential? Is it the necessary price of incorporating fast matrix multiplication?

Our deterministic algorithm for producing a transform for the Frobenius form costs $O(n^\theta(\log n)(\log\log n))$ factor. Such doubly (and triply) logarithmic factors often appear in complexity results because of the use of FFT-based methods for polynomial arithmetic, or because the algorithm works over an extension ring. Our algorithm assumes standard polynomial arithmetic and the doubly logarithmic factor appears in a more fundamental way. The algorithm converges in $O(\log\log n)$ applications of Keller-Gehrig's (1985) algorithm for the shifted Hessenberg form. From the work of Eberly (2000) follows a Las Vegas probabilistic algorithm for Frobenius form that requires an expected number of $O(n^\theta(\log n))$ field operations — this matches Keller-Gherig's result for the characteristic polynomial. Except for the use of randomization, Eberly's result is also free of quibbles — no field extensions are required and a transform is also produced.

> Is the additional $\log\log n$ factor in our result essential? Is it the necessary price of avoiding randomization?

10.2 Bit Complexity

In the study of bit complexity of linear algebra problems there are many directions that research can take. For example: space-efficient or space-optimal algorithms; coarse grain parallel algorithms; fast or processor-efficient parallel algorithms; special algorithms for sparse or structured input, ie. "black-box" linear algebra. We make no attempt here to survey these research areas or to undertake the important task of exposing the links between them. We will focus, in line with the rest of this document, on sequential algorithms for the case of a dense, unstructured input matrix.

Let us first make clear the direction we have taken in the previous chapters. Our programme was to demonstrate that our generic algorithms could be applied, in a straightforward way, to get algorithms for canonical forms of integer matrices which are deterministic, asymptotically fast, produce transform matrices, and which handle efficiently the case of an input matrix with arbitrary shape and rank profile. Thus, much emphasis was placed on handling the most general case of the problem. An additional goal was to produce "good" transform matrices, that is, with good explicit bounds on the magnitudes of entries and good bounds on the total size. This additional goal was achieved under the restriction that the asymptotic complexity should be the same (up to log factors) as required by the algorithm to produce only the form itself.

Since the running times we have achieved for our Hermite and Smith form algorithms for integer matrices are currently the fastest known — at least for solving the general problem as described above — there arises the question of whether these running times can be improved. Our answer is that we believe the bit complexities we have achieved allow for substantial improvement. For this reason, it is useful to identify and clearly define some versions of these fundamental problems which deserve attention in this regard. We do this now.

Recovery of Integer Matrix Invariants

The problem of computing matrix canonical forms belongs to a broader area which we can call "recovery of matrix invariants". We can identify seven important problems: DET, RANK, MINPOLY, CHARPOLY, HERMITE, SMITH and FROBENIUS. These problems ask for the determinant, the rank, the minimal polynomial, the characteristic polynomial, and the Smith, Hermite and Frobenius canonical form of an input matrix A.

An interesting question, and one which is largely unanswered, is to demonstrate fast algorithm to compute each of these invariants in the case of an integer matrix. To greatly simplify the comparison of different algorithms, we define the problems DET, RANK, MINPOLY, CHARPOLY, HERMITE, SMITH and FROBENIUS more precisely with the following comments:

- We consider as input a square $n \times n$ integer input matrix A and summarize the soft-Oh complexity by giving the exponent of the parameter n. Complexity results will be given in terms of word operations — the assumption being that, for some $l = O(\log n + \log\log ||A||)$, where l depends on a given algorithm, the computer on which we are working has words of length (at least) l and the list of primes between 2^{l-1} and 2^l are provided for. This single parameter model captures the essential feature of working over $\mathbb{Z}$; the size (bit-length) of integers that we need to compute with grows from the starting size of $\log_2 ||A||$ — typically at least linearly with respect to n.

- For the problems Hermite, Smith, Frobenius, we ask only for the canonical form and not transform matrices. The reason for this is that the transforms are highly non-unique. By asking only for the matrix invariant there can be no quibbles about the "quality" of the output — our attention is focused purely on the running time.

- For the problems Hermite and Smith we allow that the algorithm might require the input matrix to be nonsingular. A nice feature of Hermite is that, when A is nonsingular, the output will have total size bounded by $O\tilde{}(n^2)$ bits — this is about the same as the input matrix itself. All the other problems also have output which have total size bounded by $O\tilde{}(n^2)$ bits.

Suppose we are given a prime p that satisfies $p > ||A||$. Then all the problems mentioned above can be solved deterministically over the field $\mathbb{Z}/(p)$ in $O\tilde{}(n^\theta)$ bit operations. Now consider the bit complexity over $\mathbb{Z}$. On the one hand, we have taken a naive approach. Our algorithms for solving the problems Det, Rank, Smith, Hermite require $O\tilde{}(n^\theta \cdot n)$ bit operations — the cost in bit operations over $\mathbb{Z}$ is obtained by multiplying the cost to solve the problem over $\mathbb{Z}/(p)$ by a factor of n.

Problem	DET	LV	MC
Det	4	3.5	
Rank	4	3.5	3
MinPoly	5	4	3.5
CharPoly	4		3.5
Hermite	4		
Smith	4		3.5
Frobenius	5	4	3.5
LinSys	4	3	

Table 10.1: Single Parameter Complexity: $\theta = 3$

On the other hand, faster algorithms for almost all these problems are available. In Table 10.1 we give some results under the assumption of standard matrix multiplication. Many of these results can be improved by incorporating fast matrix multiplication, and we discuss this below, but first we give some references for and make some remarks about the algorithms supporting the running times in Table 10.1.

The MC result for MinPoly is due to Kaltofen (1992), who studies the division free computation of the determinant. From his work there follows a deterministic $O\tilde{}(n^{3.5})$ bit operations algorithm to compute the minimal polynomial of the linearly recurring scalar sequence $uv, uAv, uA^2v, \ldots$, for given vectors u and v with word-size entries. Many of the other results in Table 10.1 follow as a corollary to this result since they are obtained by reducing to the problem MinPoly.

- From Wiedemann (1986), we know that for randomly chosen[2] vectors u and v the minpoly of $uv, uAv, uA^2v, \ldots$ will be, with high probability, the minimal polynomial of A — this give the MC algorithm for MinPoly.

- Similarly, for a well chosen diagonal matrix D, and for nonsingular A, the minimal polynomial of DA will coincide with the characteristic polynomial of DA — this gives the LV algorithm for Det.

- Now consider when A may be singular. Saunders observes that, for randomly chosen u, v and D, the correct rank of A can be recovered with high probability from the minimal polynomial of DAA^T by adapting the technique in Saunders *et al.* (2000).

- In (Storjohann, 2000) we observe Frobenius can be reduced to MinPoly plus the additional computation of the Frobenius form of A modulo a randomly chosen word-size prime p — this gives the MC results for Frobenius and CharPoly.

The MC algorithm for Rank is obtained simply by choosing a random word-size prime p and returning the rank of A modulo p. The DET result for Rank, CharPoly, Hermite and LinSys are well know; that for Smith follows from our work here.

Now let us discuss the DET and LV results for MinPoly and Frobenius. The reason that the DET results have exponent 5 is because the best bound available for the number of word-size primes p which are

[2]For our purposes here, random word-size entries.

bad[3] with respect to homomorphic imaging is $O\tilde{~}(n^2)$. Thus, even though $O\tilde{~}(n)$ word-size primes are sufficient to construct the result, $O\tilde{~}(n^2)$ are required to guarantee correctness. Giesbrecht (1993) observes that the minimal polynomial can be certified modulo $O\tilde{~}(n)$ (randomly chosen) word-size primes — the LV result of MinPoly now follows as a corollary to his LV algorithm for Frobenius. In (Giesbrecht and Storjohann, 2000) we extend this certification technique to get the LV result for Frobenius.

The MC result for Smith is due to Eberly *et al.* (2000). Their algorithms works by reducing the problem to $O(n^{.5})$ applications of LinSys — compute $A^{-1}b$ for a given vector b. The LV algorithm for LinSys is p-adic lifting as described in Dixon (1982).

Many of the complexities in Table 10.1 can be improved substantially by incorporating fast matrix methods. Work is currently in progress, but see for example Kaltofen (1992), Mulders and Storjohann (2000).

[3] Those for which the structure of the Frobenius form or degree of the minimal polynomial of A as computed over $\mathbb{Z}/(p)$ differs from that computed over $\mathbb{Z}$.

Bibliography

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

D. Augot and P. Camion. Frobenius form and cyclic vectors. *C.-R.-Acad.-Sci.-Paris-Ser.-I-Math.*, 318(2):183–188, 1994.

E. Bach and J. Shallit. *Algorithmic Number Theory*, volume 1 : Efficient Algorithms. MIT Press, 1996.

E. Bach. Linear algebra modulo N. Unpublished manuscript., December 1992.

E. H. Bareiss. Sylvester's identity and multistep integer-preserving Gaussian elimination. *Mathematics of Computation*, 22(103):565–578, 1968.

E. H. Bareiss. Computational solution of matrix problems over an integral domain. *Phil. Trans. Roy. Soc. London*, 10:68–104, 1972.

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

W. Baur and V. Strassen. The complexity of partial derivatives. *Theoretical Computer Science*, 22(3):317—330, 1983.

D. J. Bernstein. Multidigit multiplication for mathematicians. *Advances in Applied Mathematics*, 1998. To appear.

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

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

E. Bodewig. *Matrix Calculus*. North Holland, Amsterdam, 1956.

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

W. C. Brown. *Matrices over Commutative Rings*. Marcel Dekker, Inc., New York, 1993.

J. Buchmann and S. Neis. Algorithms for linear algebra problems over principal ideal rings. Technical report, Technische Hochschule Darmstadt, 1996.

J. Bunch and J. Hopcroft. Triangular factorization and inversion by fast matrix multiplication. *Mathematics of Computation*, pages 231–236, 1974.

P. Bürgisser, M. Karpinski, and T. Lickteig. Some computational problems in linear algebra as hard as matrix multiplication. *Cop. Compl.*, 1:191–155, 1991.

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

S. Cabay. Exact solution of linear systems. In *Proc. Second Symp. on Symbolic and Algebraic Manipulation*, pages 248—253, 1971.

T-W. J. Chou and G. E. Collins. Algorithms for the solutions of systems of linear diophantine equations. *SIAM Journal of Computing*, 11:687–708, 1982.

H. Cohen. *A Course in Computational Algebraic Number Theory*. Springer-Verlag, 1996.

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

D. Coppersmith. Rectangular matrix multiplication revisited. Technical Report Research Report 20498, IBM Thomas J. Watslon Research Center, Yorktown Heights, NY 10598, USA, 1996.

T. H. Cormen, C. E. Leiserson, and Ronald L. Rivest. *Introduction to Algorithms*. MIT Press and McGraw-Hill, 1989.

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

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

P. D. Domich. *Residual Methods for Computing Hermite and Smith Normal Forms*. PhD thesis, School of Operations Research and Industrial Engineering, Cornell University, Ithaca, NY, 1985.

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

W. Eberly, M. Giesbrecht, and G. Villard. Computing the Smith form of a dense integer matrix. In *Proc. 31st Ann. IEEE Symp. Foundations of Computer Science*, 2000.

W. Eberly. Black box frobenius decompositions over small fields. In C. Traverso, editor, *Proc. Int'l. Symp. on Symbolic and Algebraic Computation: ISSAC '00*, pages 106–113. ACM Press, 2000.

J. Edmonds. On systems of distinct linear representative. *J. Res. Nat. Bur. Standards*, 71B:241–245, 1967.

X. G. Fang and G. Havas. On the worst-case complexity of integer gaussian elimination. In W. W. Küchlin, editor, *Proc. Int'l. Symp. on Symbolic and Algebraic Computation: ISSAC '97*, pages 28–31. ACM Press, 1997.

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

M. Giesbrecht and A. Storjohann. Computing rational forms of integer matrices. Submitted., 2000.

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

M. Giesbrecht. Fast algorithms for rational forms of integer matrices. In M. Giesbrecht, editor, *Proc. Int'l. Symp. on Symbolic and Algebraic Computation: ISSAC '94*, pages 305–311. ACM Press, 1994.

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

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

M. Giesbrecht. Probabalistic computation of the Smith normal form of a sparse integer matrix. In H. Cohen, editor, *Algorithmic Number Theory: Second International Symposium*, pages 175–188, 1996. Proceedings to appear in Springer's Lecture Notes in Computer Science.

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

G. Havas and B. S. Majewski. Hermite normal form computation for integer matrices. *Congressus Numerantium*, 105:87–96, 1994.

G. Havas and B. S. Majewski. Integer matrix diagonalization. *Journal of Symbolic Computation*, 24:399–408, 1997.

G. Havas and C. Wagner. Matrix reduction algorithms for Euclidean rings. In *Proc. 1998 Asian Symposium on Computer Mathematics*, pages 65—70. Lanzhou University Press, 1998.

G. Havas, D. F. Holt, and S. Rees. Recognizing badly presented Z-modules. *Linear Algebra and its Applications*, 192:137–163, 1993.

G. Havas, B. S. Majewski, and K. R. Matthews. Extended gcd and Hermite normal form algorithms via lattice basis reduction. *Experimental Mathematics*, 7:125—136, 1998.

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

J. A. Howell. Spans in the module $(\mathbb{Z}_m)^s$. *Linear and Multilinear Algebra*, 19:67—77, 1986.

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

X. Huang and V. Y. Pan. Fast rectangular matrix multilplications and improving parallel matrix computations. In M. Hitz and E. Kaltofen, editors, *Second Int'l Symp. on Parallel Symbolic Computation: PASCO '97*, pages 11–23. ACM Press, 1997.

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

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

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

E. Kaltofen and B. D. Saunders. On Wiedemann's method of solving sparse linear systems. In *Proc. AAECC-9, Lecture Notes in Comput. Sci., vol. 539*, pages 29–38, 1991.

E. Kaltofen, M. S. Krishnamoorthy, and B. D. Saunders. Fast parallel computation of Hermite and Smith forms of polynomial matrices. *SIAM Journal of Algebraic and Discrete Methods*, 8:683–690, 1987.

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

E. Kaltofen. On computing determinants of matrices without divisions. In P. S. Wang, editor, *Proc. Int'l. Symp. on Symbolic and Algebraic Computation: ISSAC '92*, pages 342–349. ACM Press, 1992.

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

R. Kannan. Polynomial-time algorithms for solving systems of linear equations over polynomials. *Theoretical Computer Science*, 39:69–88, 1985.

I. Kaplansky. Elementary divisors and modules. *Trans. of the Amer. Math. Soc.*, 66:464–491, 1949.

W. Keller-Gehrig. Fast algorithms for the characteristic polynomial. *Theoretical Computer Science*, 36:309—317, 1985.

W. Krull. Die verschiedenen Arten der Hauptidealringe. Technical Report 6, Sitzungsberichte der Heidelberg Akademie, 1924.

F. Lübeck. On the computation of elementary divisors of integer matrices. Unpublished manuscript, 1998.

H. Lüneburg. *On Rational Normal Form of Endomorphisms: a Primer to Constructive Algebra.* Wissenschaftsverlag, Mannheim, 1987.

B. S. Majewski and G. Havas. The complexity of greatest common divisor computations. *Algorithmic Number Theory, Lecture Notes in Computer Science*, 877:184—193, 1994.

T. Mulders and A. Storjohann. The modulo N extended gcd problem for polynomials. In O. Gloor, editor, *Proc. Int'l. Symp. on Symbolic and Algebraic Computation: ISSAC '98*, pages 105—112. ACM Press, 1998.

T. Mulders and A. Storjohann. Diophantine linear system solving. In S. Dooley, editor, *Proc. Int'l. Symp. on Symbolic and Algebraic Computation: ISSAC '99*, pages 281–288. ACM Press, 1999.

T. Mulders and A. Storjohann. Certified diophantine dense linear system solving. In progress, 2000.

M. Newman. *Integral Matrices.* Academic Press, 1972.

M. Newman. The Smith normal form. *Linear Algebra and its Applications*, 254:367—381, 1997.

P. Ozello. *Calcul Exact Des Formes De Jordan et de Frobenius d'une Matrice.* PhD thesis, Université Scientifique Technologique et Medicale de Grenoble, 1987.

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

D. Saunders, A. Storjohann, and G. Villard. Matrix rank certification. In progress, 2000.

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

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

A. Schönhage. Unitäre Transformationen großer Matrizen. *Num. Math.*, 20:409–4171, 1973.

A. Schönhage. Probabilistic computation of integer polynomial GCD's. *Journal of Algorithms*, 9:365–371, 1988.

G. Shapiro. Gauss elimination for singular matrices. *Mathematics of Computation*, 17:441–445, 1963.

C. C. Sims. *Abstract Algebra: A Computational Approach.* John Wiley&Sons, New York, 1984.

H. J. S. Smith. On systems of linear indeterminate equations and congruences. *Phil. Trans. Roy. Soc. London*, 151:293–326, 1861.

D. Squirrel. Computing kernels of sparse integer matrices. Master's thesis, Reed College, University of California Berkeley, 1999.

A. Steel. A new algorithm for the computation of canonical forms of matrices over fields. *Journal of Symbolic Computation*, 24:409—432, 1997.

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

A. Storjohann and G. Labahn. A fast Las Vegas algorithm for computing the Smith normal form of a polynomial matrix. *Linear Algebra and its Applications*, 253:155—173, 1997.

A. Storjohann and T. Mulders. Fast algorithms for linear algebra modulo N. In G. Bilardi, G. F. Italiano, A. Pietracaprina, and G. Pucci, editors, *Algorithms — ESA '98*, LNCS 1461, pages 139–150, 1998.

A. Storjohann and G. Villard. Algorithms for similarity transforms. *extended abstract.* In T. Mulders, editor, *Proc. Seventh Rhine Workshop on Computer Algebra: RWCA'00*, pages 109—118, Bregenz, Austria, 2000.

A. Storjohann. Computation of Hermite and Smith normal forms of matrices. Master's thesis, Dept. of Computer Science, University of Waterloo, 1994.

A. Storjohann. A fast, practical and deterministic algorithm for triangularizing integer matrices. Technical Report 255, Departement Informatik, ETH Zürich, December 1996.

A. Storjohann. Faster algorithms for integer lattice basis reduction. Technical Report 249, Departement Informatik, ETH Zürich, July 1996.

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

A. Storjohann. A solution to the extended gcd problem with applications. In W. W. Küchlin, editor, *Proc. Int'l. Symp. on Symbolic and Algebraic Computation: ISSAC '97*, pages 109–116. ACM Press, 1997.

A. Storjohann. Computing Hermite and Smith normal forms of triangular integer matrices. *Linear Algebra and its Applications*, 282:25–45, 1998.

A. Storjohann. An $O(n^3)$ algorithm for Frobenius normal form. In O. Gloor, editor, *Proc. Int'l. Symp. on Symbolic and Algebraic Computation: ISSAC '98*, pages 101–104. ACM Press, 1998.

A. Storjohann. Monte carlo algorithms for the frobenius form of of an integer matrix. In progress, 2000.

V. Strassen. Vermeidung von Divisionen. *J. reine angew. Math.*, 264:182—202, 1973.

G. Villard. Computation of the Smith normal form of polynomial matrices. In M. Bronstein, editor, *Proc. Int'l. Symp. on Symbolic and Algebraic Computation: ISSAC '93*, pages 208–217. ACM Press, 1993.

G. Villard. Generalized subresultants for computing the Smith normal form of polynomial matrices. *Journal of Symbolic Computation*, 20(3):269—286, 1995.

G. Villard. Fast parallel algorithms for matrix reduction to normal forms. *Applicable Algebra in Engineering, Communication and Control*, 8:511—537, 1997.

G. Villard. Frobenius normal form: Transforming matrices. Personal communication, September 23, 1999.

G. Villard. Computing the frobenius form of a sparse matrix. In V. G. Ganzha, E. W. Mayr, and E. V. Vorozhtsov, editors, *Computer Algebra in Scientific Computing – CASC 2000*. Springer Verlag, 2000.

J. von zur Gathen and J. Gerhard. *Modern Computer Algebra*. Cambridge University Press, 1999.

C. Wagner. *Normalformberechnung von Matrizen über euklidischen Ringen*. PhD thesis, Universität Karlsruhe, 1998.

D. Wiedemann. Solving sparse linear equations over finite fields. *IEEE Trans. Inf. Theory*, IT-32:54–62, 1986.

S. Winograd. The algebraic complexity of functions. In *Proc. International Congress of Mathematicians, Vol. 3*, pages 283–288, 1970.

Curriculum Vitae

Name:	Arne STORJOHANN
Date of Birth:	20^{th} of December 1968
Place of Birth:	Braunschweig
Nationality:	German

1982-1987	Brantford Collegiate Institute High School Diploma
1987-1994	University of Waterloo Bachelor of Mathematics Master of Mathematics
1994-1996	Symbolic Computation Group, University of Waterloo Research Assistant
1996-2000	PhD student (ETHZ)