Computation of Hermite and Smith Normal Forms of Matrices

by

Arne Storjohann

A thesis
presented to the University of Waterloo
in fulfilment of the
thesis requirement for the degree of
Master of Mathematics
in
Computer Science

Waterloo, Ontario, Canada, 1994

Abstract

We study the problem of computing Hermite and Smith normal forms of matrices over principal ideal domains. The main obstacle to efficient computation of these forms is the potential for excessive growth of intermediate expressions. Most of our work here focuses on the difficult case of polynomial matrices: matrices with entries univariate polynomials having rational number coefficients.

One first result is a fast Las Vegas probabilistic algorithm to compute the Smith normal form of a polynomial matrix for those cases where pre- and post-multipliers are not also required. For computing Hermite normal forms of polynomial matrices, and for computing pre- and post-multipliers for the Smith normal form, we give a new sequential deterministic algorithm. We present our algorithms for the special case of square, nonsingular input matrices. Generalizations to the nonsquare and/or singular case are provided via a fast Las Vegas probabilistic preconditioning algorithm that reduces to the square, nonsingular case.

In keeping with our main impetus, which is practical computation of these normal forms, we show how to apply homomorphic imaging schemes to avoid computation with large integers and polynomials. Bounds for the running times of our algorithms are given that demonstrate significantly improved complexity results over existing methods.

Acknowledgements

My first acknowledgment must go to my supervisor George Labahn. In addition to providing vision, funding, and encouragement, George gave me unlimited freedom to explore new avenues for research, yet was always available to discuss new ideas. My thanks also goes to the other members of the Symbolic Computation Group, who provided a relaxed and friendly working environment.

Second, I would like to thank the members of my examining committee, William Gilbert and Jeffrey Shallit, for their time spent in reviewing this thesis.

Third, a heartfelt thanks to the friends I have gained through rowing: fellow athletes as well as coaches and administrators. My involvement with this sport during the research and writing of this thesis has kept me sane. Special thanks goes to Don McLean for his friendship, advice on academic matters, and implicit encouragement. Finally, my parents deserve much credit for the immeasurable support they have provided for my studies throughout the years.

Contents

Chapter 1

Introduction

A common theme in matrix algebra — especially in a computational setting — is to transform an input matrix into an "equivalent" but simpler canonical form while preserving key properties of the original matrix (e.g. the Gauss-Jordan form of a matrix over a field). For input matrices over principal ideal domains, two fundamental constructions in this regard are the *Hermite normal form*, a triangularization, and the *Smith normal form*, a diagonalization.

Any nonsingular matrix A over a Euclidean domain $\mathbf{R}$ can be transformed via elementary row operations to an upper triangular matrix. Further row operations can be used to reduce the size of entries above the diagonal in each column modulo the diagonal entry and to ensure uniqueness of the diagonal entries; the resulting upper triangular matrix is called the Hermite normal form of A (abbr. by HNF). By applying elementary column as well as row operations, the matrix A can be transformed to a diagonal matrix such that each diagonal entry divides the next and is unique; the resulting diagonal matrix is called the Smith normal form of A (abbr. by SNF). The Hermite and Smith normal forms are canonical forms for matrices over principal ideal domains — they always exist and are unique.

Hermite first proved the existence of the HNF triangularization for integer matrices in a paper of 1851 [19]. Hermite gave a procedure for transforming an integer input matrix into HNF using row operations, and although he did not explicitly prove uniqueness of the form, it is fairly clear he knew of it. Smith gave a construction for the SNF diagonalization in a paper of 1861 [33]. Definitions for the HNF and SNF generalize readily to rectangular matrices of arbitrary rank over any principal idea domain (cf. §2).

The study of matrices over rings is a huge field encompassing almost every area of mathematics — this is demonstrated by the numerous and diverse applications of the Hermite and Smith normal form. Applications involving integer matrices include: solving systems of linear diophantine equations (cf. Blankinship [6] or Iliopoulos [20]); diophantine analysis and integer programming (cf. Newman [29]); computing the canonical structure of finitely represented abelian groups (cf. Cannon and Havas [8] or Havas, Holt and Rees [18]). For matrix polynomials, the HNF corresponds to computing a change of basis for modules over polynomial domains. Examples of the use of HNF in such computations include computing integral basis in Trager's algorithm [34] for symbolic integration of algebraic functions and finding matrix polynomial greatest common divisors. Both the HNF and SNF are very useful in linear systems theory (cf. Kailath [22] or Ramachandran [30]).

The existence and uniqueness of the SNF is regarded as one of the most important result in all of elementary matrix theory. The SNF provides a constructive proof of the basis theorem for finitely generated Abelian groups (cf. [17]), and a fundamental result of matrix theory states that if A and B are square matrices over a field, then A is similar to B if and only if their respective characteristic matrices $xI - A$ and $xI - B$ have the same SNF (cf. Gantmacher [13, §5, Theorem 7]). Other applications of the SNF to matrices over fields include computing the invariant factors, computing geometric multiplicities of the eigenvalues, and finding the Frobenius (or rational canonical) form.

We proceed with an explicit example of the HNF for matrices over two different domains — the ring of integers $\mathbb{Z}$ and the ring $\mathbf{Q}[x]$ of univariate polynomials with rational coefficients. Recall that an *elementary row operation* is one of: (e1) multiplying a row by an invertible element, (e2) adding a multiple of one row to a different row, or (e3) interchanging two rows. To each elementary row operation there corresponds a nonsingular (hence invertible) *elementary matrix* and an elementary row operation can be applied to a matrix by premultiplying by the corresponding elementary matrix. Over the ring of integers only the units ± 1 are invertible (compare this with Gauss-Jordan elimination over the field $\mathbf{Q}$ where every nonzero element is invertible). Consider the matrix

$$A = \begin{bmatrix} 2 & -8 & 14 & 20 \\ 0 & 1 & -6 & -5 \\ 10 & -39 & 70 & 131 \\ 2 & -18 & 50 & -90 \end{bmatrix} \tag{1.1}$$

Matrix A can be reduced to triangular form using a type of Gaussian elimination that that applies only elementary (invertible over $\mathbb{Z}$) row operations. (This algorithm is given in §2.1.) For matrix A of (1.1) we can obtain a transformation matrix U and a reduced matrix H where

$$\begin{array}{c@{}c@{}c@{}c@{}c}
U & & A & & H \\
\begin{bmatrix} -281 & 74 & 54 & 12 \\ -26 & 6 & 5 & 1 \\ -47 & 11 & 9 & 2 \\ 21 & -6 & -4 & -1 \end{bmatrix} &
\begin{bmatrix} 2 & -8 & 14 & 20 \\ 0 & 1 & -6 & -5 \\ 10 & -39 & 70 & 131 \\ 2 & -18 & 50 & -90 \end{bmatrix} & = &
\begin{bmatrix} 2 & 0 & 2 & 4 \\ 0 & 1 & 0 & 15 \\ 0 & 0 & 6 & 4 \\ 0 & 0 & 0 & 16 \end{bmatrix}
\end{array} \tag{1.2}$$

The matrix H is the HNF of A. The determinant of the transformation matrix U is a unit in $\mathbb{Z}$ (in this case, $\det(U) = -1$) and hence by Cramer's rule U has an integer inverse (U is said to be *unimodular* in this case). Matrix U is unimodular precisely because it is the product of unimodular elementary matrices. Equation (1.2) expresses the rows of H as integer linear combinations of the rows of A. Conversely, $U^{-1}H$ expresses the rows of A as integer linear combinations of the rows of H:

$$\begin{array}{c@{}c@{}c@{}c}
U^{-1} & H & & A \\
\begin{bmatrix} 1 & -8 & 2 & 8 \\ 0 & 1 & -1 & -1 \\ 5 & -39 & 10 & 41 \\ 1 & -18 & 8 & 9 \end{bmatrix} &
\begin{bmatrix} 2 & 0 & 2 & 4 \\ 0 & 1 & 0 & 15 \\ 0 & 0 & 6 & 4 \\ 0 & 0 & 0 & 16 \end{bmatrix} & = &
\begin{bmatrix} 2 & -8 & 14 & 20 \\ 0 & 1 & -6 & -5 \\ 10 & -39 & 70 & 131 \\ 2 & -18 & 50 & -90 \end{bmatrix}
\end{array}$$

We conclude that the nonzero rows of H provide a basis for the lattice of A (the set of all integer linear combinations of rows of A). While reduction to Gauss-Jordan form over $\mathbf{Q}$ is useful for solving systems of linear equations, reduction to HNF over $\mathbb{Z}$ can be used to solve systems of linear diophantine equations.

The size of an integer is given by its absolute value. In the Euclidean domain $\mathbf{F}[x]$, $\mathbf{F}$ a field, the size of a polynomial is given by its degree. A nonsingular matrix over $\mathbf{F}[x]$ is in HNF if it is upper triangular with diagonal elements monic and elements above the diagonal in each column having smaller degree than the diagonal entry. Consider the matrix

$$A = \begin{bmatrix} -2x - 2 + x^3 + x^2 & x^2 + 2x + 1 & -x - 1 \\ 173x + 152 - 85x^3 - 84x^2 & -85x^2 - 171x - 82 & 87x + 79 \\ -7x + 767 - 55x^3 - 89x^2 & -56x^2 - 71x - 163 & -21x + 274 + x^2 \end{bmatrix} \tag{1.3}$$

Gaussian elimination using unimodular row operations can be used to reduce a polynomial matrix to HNF. For the matrix A of (1.3) we obtain $UA = H$ for U unimodular and H in HNF where

$$U = \left[\begin{array}{ccc} \frac{5409}{140} + \frac{34}{7}\,x^2 - \frac{7279}{20}\,x & \frac{31}{70} - \frac{589}{140}\,x + \frac{2}{35}\,x^2 & -\frac{4}{35}\,x + \frac{1}{70} \\ -3112\,x - 6315 - 6316\,x^2 + 85\,x^3 & -36\,x - 73 - 73\,x^2 + x^3 & -x - 2 - 2\,x^2 \\ 85\,x^3 - 6061\,x^2 - 22060\,x - 12706 & x^3 - 70\,x^2 - 255\,x - 147 & -2\,x^2 - 7\,x - 4 \end{array} \right]$$

and

$$H = \left[\begin{array}{ccc} 1 & \frac{27}{140}\,x - \frac{1}{140} & -\frac{19}{140}\,x + \frac{37}{140} \\ 0 & x^2 - 2\,x - 3 & 0 \\ 0 & 0 & x^2 - 2\,x - 3 \end{array} \right].$$

The rows of matrix H provide a unique basis for the set of all linear combinations (over $\mathbf{Q}[x]$) of the rows of matrix A. One application of reduction to HNF for polynomial matrices is solving systems of polynomial diophantine equations. Another application, which we expand upon now, is computing matrix greatest common divisors.

For the following example, let $P(z)$ and $Q(z)$ be two nonsingular matrices in $\mathbf{Q}[z]^{n\times n}$. (In general, let $\mathbf{R}^{n\times m}$ denote the set of all $n \times m$ matrices over a ring $\mathbf{R}$.) In some computations found in linear control theory one works with the matrix rational form $P(z)Q(z)^{-1}$. For these cases it is often necessary to have the rational form in its simplest form. In particular, the matrix polynomials should have no common divisors on the right. To effect this simplification, the greatest common right divisor can be determined using a HNF computation and divided out. This computation can be done as follows. Let

$$A(z) = \left[\begin{array}{c} P(z) \\ Q(z) \end{array} \right]$$

belong to $\mathbf{Q}[z]^{2n\times n}$. By computing the HNF of $A(z)$ we can obtain unimodular matrices $U(z), V(z) \in \mathbf{Q}[z]^{2n\times 2n}$ and a matrix $N(z) \in \mathbf{Q}[z]^{n\times n}$ such that

$$U(z)V(z) = V(z)U(z) = I_{2n} \tag{1.4}$$

and

$$U(z)A(z) = \left[\begin{array}{c} N(z) \\ 0 \end{array} \right]. \tag{1.5}$$

Note that the matrix on the right hand side of (1.5) is the HNF of $A(z)$ and that $V(z) = U(z)^{-1}$. Write

$$U(z) = \left[\begin{array}{cc} U_{11}(z) & U_{12}(z) \\ U_{21}(z) & U_{22}(z) \end{array} \right] \quad \text{and} \quad V(z) = \left[\begin{array}{cc} V_{11}(z) & V_{12}(z) \\ V_{21}(z) & V_{22}(z) \end{array} \right],$$

where the individual blocks are all of size $n \times n$. Then equations (1.4) and (1.5) imply that

$$A(z) = V(z)\begin{bmatrix} N(z) \\ 0 \end{bmatrix}$$

so that

$$\begin{bmatrix} P(z) \\ Q(z) \end{bmatrix} = \begin{bmatrix} V_{11}(z)N(z) \\ V_{21}(z)N(z) \end{bmatrix}$$

and hence $N(z)$ is a common divisor of $P(z)$ and $Q(z)$ on the right. In addition equation (1.4) along with the partitioning (1) gives

$$U_{11}(z)V_{11}(z) + U_{12}(z)V_{21}(z) = I_n$$

hence $V_{11}(z)$ and $V_{21}(z)$ cannot have nontrivial (i.e. non-unimodular) divisors on the right. Therefore $N(z)$ is the largest right divisor — as desired.

Consider again the integer matrix A of (1.1). By applying unimodular column operations as well as unimodular row operations, we can reduce the matrix to diagonal form. We obtain unimodular matrices U and V and a diagonal matrix S that verify $UAV = S$. For example,

$$\overset{U}{\begin{bmatrix} -5 & 0 & 1 & 0 \\ -10 & 0 & 2 & -1 \\ -26 & 3 & 5 & 2 \\ 103 & -10 & -20 & -3 \end{bmatrix}} \overset{A}{\begin{bmatrix} 2 & -8 & 14 & 20 \\ 0 & 1 & -6 & -5 \\ 10 & -39 & 70 & 131 \\ 2 & -18 & 50 & -90 \end{bmatrix}} \overset{V}{\begin{bmatrix} 10 & -110 & -109 & -956 \\ 1 & -31 & -31 & -279 \\ 0 & -5 & -5 & -46 \\ 0 & 1 & 1 & 9 \end{bmatrix}} = \overset{S}{\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 2 & 0 & 0 \\ 0 & 0 & 2 & 0 \\ 0 & 0 & 0 & 48 \end{bmatrix}}$$

The matrix S of (1.6) is the SNF of A. In addition to being diagonal, an integer matrix in SNF has the property that each diagonal element is nonnegative and divides the next. Matrices U and V of (1.6) are unimodular since they are product of unimodular matrices. (This follows directly from the algorithm used to find S given in §2.2.) The usefulness of the SNF hinges on it being a canonical form; for any integer matrix A, there exists a unique matrix S in SNF such that $UAV = S$ for some unimodular matrices U and V. The SNF, defined for matrices over a principal ideal domain, is directly related to the existence of canonical decompositions for finitely generated modules over principal ideal domains. Next, we present the key ideas of this relationship by considering considering the important case of matrices over the integers.

There is a natural correspondence between $\mathbb{Z}$-modules and abelian groups. The fundamental theorem of finitely generated abelian groups classifies all such groups up to isomorphism by giving a canonical decomposition. One version of the theorem states that a finitely generated group G has the (unique) direct decomposition $G \cong G_1 \oplus G_2 \oplus \cdots \oplus G_r \oplus G_{r+1} \oplus \cdots G_{r+f}$ where: G_i is a nontrivial finite cyclic group of order s_i for $i = 1, \ldots, r$; G_i is an infinite cyclic group for $i = r+1, \ldots, r+f$; and $s_1|s_2|\cdots|s_r$. Fully knowing the structure and characteristics of a finitely generated abelian group is tantamount to knowing its canonical decomposition.

The results of some computations involving groups produce finitely generated abelian groups — let G be one such group. For example, G may be known to be a normal (hence abelian) subgroup of an unclassified group of large order, say G'. Knowing the structure of G may lend insight into the structure of G'. One possible (and desirable) representation for G would be the unique integers r, f, and $s_1, \ldots, s_r$ corresponding to the canonical decomposition for G. Typically, however, groups that arise during computations are specified via a more general *presentation*. A *presentation* for G is a set of m *generators* together with n *relations* among the generators — usually written as $\langle g_1, g_2, \ldots, g_m | \sum_{j=1}^{m} a_{i,j} g_i = 0, i = 1, \ldots, n \rangle$. These relations can be expressed in matrix form as $A\vec{g} = 0$ where A is an $n \times m$ matrix with $a_{i,j}$ as the entry in the i-th row j-th column and $\vec{g}$ is a vector of m generators. The column dimension of the matrix A implicitly gives the number of generators. Hence, an $n \times m$ integer matrix A represents the unique group G (up to isomorphism) that can generated by m generators, say $\vec{g} = [g_1, \ldots, g_m]$, that satisfy the relations $A\vec{g} = 0$ and such that all other relations among the generators are contained in the lattice of A. For example, consider the presentation given by the SNF matrix

$$S = \begin{bmatrix} 1 & & & \\ & 2 & & \\ & & 6 & \\ & & & 0 \end{bmatrix}. \tag{1.6}$$

Matrix S has column dimension 4 and so represents a group G that can be generated by 4 generators, say $\vec{g} = [g_1, g_2, g_3, g_4]$. The relations $S\vec{g} = 0$ give the presentation $\langle g_1, g_2, g_3, g_4 | g_1, 2g_2, 6g_3 \rangle$. Each relation involves only a single generator so $G \cong G_1 \oplus G_2 \oplus G_3 \oplus G_4$ where, for $1 \le k \le 4$, G_k is the cyclic group generated by g_k. The relation $g_1 = 0$ implies G_1 is the trivial group C_1. Similarly, $G_2 \cong C_2$ and $G_3 \cong C_6$ where C_k is the cyclic group consisting of k elements. Since no relation involves g_4, G_4 is the infinite cyclic group C. We conclude that G has the canonical decomposition $C_2 \oplus C_6 \oplus C$. In general, a SNF matrix with diagonal $[1, 1, \ldots, 1, d_1, d_2, \ldots, d_r, 0, 0, \ldots, 0]$ having f trailing zeroes and $d_1 > 1$, $d_r \neq 0$ will represent the group with canonical decomposition

$C_{d_1} \oplus C_{d_2} \oplus \cdots \oplus C_{d_r} \oplus \overbrace{C \oplus C \oplus \cdots \oplus C}^{f}$. Thus, a presentation in terms of a SNF matrix is desirable since the canonical decomposition of the group in question can be determined immediately from the diagonal entries.

Consider now a general $n \times m$ integer matrix A. Matrix A represents some group H generated by m generators, say $\vec{h} = [h_1, h_2, \ldots, h_m]$, subject to the relations $A\vec{h} = 0$. The group H can be recognized (its canonical decomposition found) by computing the SNF of A. To see how this works, let U and V be some unimodular matrices of dimension n and m respectively and let $\vec{g} = [g_1, g_2, \ldots, g_m]$ be such that $V^{-1}\vec{h} = \vec{g}$. Since V is unimodular, the group G generated by $\vec{g}$ subject to the relations $A(VV^{-1})\vec{h} = (AV)\vec{g} = 0$ is isomorphic to H. Since U is unimodular, matrix UAV has the same lattice as matrix AV. It follows that S, the SNF of A, represents the same group (up to isomorphism) as A, since there exist unimodular matrices U and V such that $UAV = S$.

The purpose of this thesis is to develop efficient algorithms for computing Hermite and Smith normal forms over two fundamental domains: the integers and the ring of univariate polynomials with coefficients from a field. As is often the case in computer algebra problems, our main task will be to ensure good bounds on the size of intermediate expressions. The different domains will give rise to very different computational techniques and in the case of matrices with polynomial entries we will need to draw on many classical results, including Chinese remaindering and interpolation, resultants and Sylvester matrices, and fraction-free Gaussian elimination (cf. Geddes, Czapor and Labahn [14]). The rest of this thesis is organized as follows.

In chapter 2 the Hermite and Smith normal form are defined over a general principal ideal domain and their basic properties are given. Chapter 3 deals with previous methods for computing these forms for matrices over the integers and discusses the problem of intermediate expression swell. A brief history of the development of HNF algorithms is given with special attention being given to a new class of algorithms which use modular arithmetic to control expression swell.

Chapter 4 introduces the problem of computing normal forms over polynomial domains. An algorithm for computing the HNF of a polynomial matrix is presented which converts the original problem over $\mathbf{F}[x]$, $\mathbf{F}$ a field, to that of triangularizing a large linear system over $\mathbf{F}$. From a theoretical point of view, the new method eliminates the problem of expression swell described since Gaussian elimination for linear systems over $\mathbf{F} = \mathbf{Q}$ admits simple polynomial bounds on the size of intermediate entries. Unfortunately, while providing simple worst-case complexity bounds, the linear systems method doesn't admit a practical implementation because the matrix describing the linear system over $\mathbf{F}$ turns out to be very large.

Chapter 5 takes a broader perspective and considers randomized probabilistic methods for computing normal forms of polynomial matrices. These methods will not lead directly to sequential deterministic complexity results — instead, we give results in terms of *expected* number of bit operations. We successfully build on some of the probabilistic results of Kaltofen, Krishnamoorthy and Saunders in [23]. A new Las Vegas probabilistic algorithm for computing the SNF of square nonsingular matrices over $\mathbf{F}[x]$ is presented. The algorithm admits a practical implementation and gives excellent results for SNF problems where unimodular multiplier matrices U and V are not required. In practice, the expected running time of the algorithm is approximately that of finding the determinant of a matrix of same dimension and similar size entries as the input matrix.

In chapter 6 we apply the results of chapter 4 and the theory developed in chapter 5 to the construction of algorithms for determining bases for lattices generated by the rows of polynomial matrices. The highlight of chapter 6 is a fast (practical) preconditioning algorithm for rectangular input matrices: given an $n \times m$ matrix A over $\mathbf{F}[x]$ with rank m and $n > m$, the algorithm returns an $(m+1) \times m$ matrix A^* with similar size entries as A and such that $\mathcal{L}(A^*) = \mathcal{L}(A)$.

Finally, chapter 7 concludes with a summary of our main results and mentions some open problems with respect to computing matrix normal forms. In particular, there are many research topics that follow naturally from the techniques and algorithms developed in chapter 5. These include developing a sequential deterministic version of the probabilistic SNF algorithm of §5 and devising new algorithms for computing similarity transforms of matrices over the rationals.

Chapter 2

Preliminaries

In this chapter the Hermite and Smith normal form are defined for matrices over principal ideal domains. The principal ideal domain (abbr. PID) is the most general type of ring for which the HNF and SNF exist. Any result given for a general PID will apply to any of the possibly very different concrete rings over which we may wish to compute — such as the integers or univariate polynomials over the rationals. Hence, the benefit of this more abstract approach is twofold: the results are of a general nature and the discussion is not encumbered with details about representation of ring elements or computation of ring operations.

Before making the notion of a HNF and SNF precise we recall some facts about PIDs. A PID is a commutative ring with no zero divisors in which every ideal is principal; that is, every ideal contains an elements which generates the ideal. Throughout, we use $\mathbf{R}$ to indicate a PID and $\mathbf{F}$ to indicate a field. Two elements a and b of a PID $\mathbf{R}$ are said to be congruent modulo a third element $c \in \mathbf{R}$ if c divides $a - b$. Congruence with respect to a fixed element c is an equivalence relation and a set consisting of distinct elements of $\mathbf{R}$, one from each congruence class, is called a *complete set of residues*.

For a PID $\mathbf{R}$, a set of elements $\{a_1, a_2, \ldots, a_n\}$, not all zero, have a greatest common divisor (a gcd). Consider the ideal $I = \langle a_1, a_2, \ldots, a_n \rangle$. An element $g \in \mathbf{R}$ is a gcd of the elements $\{a_1, a_2, \ldots, a_n\}$ if and only if $\langle g \rangle = I$. Since $g \in I$ we have $m_1 a_1 + m_2 a_2 + \cdots + m_n a_n = g$ for some elements $m_1, m_2, \ldots, m_n \in \mathbf{R}$. Since gcds are unique up to associates, and associativity is an equivalence relation on $\mathbf{R}$, we can choose a unique representative for a gcd. (Recall that ring elements a and b are associates if $a = eb$ for a ring unit e.) A *complete set of non associates* of $\mathbf{R}$ is a subset of $\mathbf{R}$ consisting of exactly one element from each equivalence class with respect to associativity.

For a PID $\mathbf{R}$ denote by $\mathbf{R}^{n \times m}$ the set of all n by m matrices over $\mathbf{R}$. A nonsingular matrix $U \in \mathbf{R}^{n\times n}$ is said to be unimodular if $\det(U)$ is a unit in $\mathbf{R}$. Note that the unimodular matrices are precisely those that have an inverse in $\mathbf{R}^{n\times n}$.

2.1 The Hermite Normal Form

A fundamental notion for matrices over rings is *left equivalence*. Two matrices A and B in $\mathbf{R}^{n\times m}$ are said to be *left equivalent* (write $A \equiv_L B$) if one is obtainable from the other via a unimodular transformation (i.e. $A = UB$ for some unimodular matrix U over $\mathbf{R}$). The key point here is that $A = UB$ implies $B = U^{-1}A$ where U^{-1}, being unimodular, is also over $\mathbf{R}$. It follows that left equivalence is an equivalence relation for matrices over $\mathbf{R}$. The HNF provides a canonical form for matrices over $\mathbf{R}$ with respect to left equivalence.

Definition 1 *Let $\mathbf{R}$ be a PID. A matrix H over $\mathbf{R}$ with full column rank is said to be in* Hermite normal form *if: (1) H is upper triangular; (2) diagonal elements of H belong to a given complete set of associates of $\mathbf{R}$; (3) in each column of H off-diagonal elements belong to a given complete set of residues of the diagonal element.*

Uniqueness of the HNF will follow from the following lemma.

Lemma 1 *Let $\mathbf{R}$ be a PID. If G and H in $\mathbf{R}^{n\times m}$ are in HNF and $G \equiv_L H$ then $G = H$.*

Proof: $G \equiv_L H$ implies $H = UG$ and $G = VH$ for some unimodular matrices U and V. The last $n-m$ rows of G and H will be zero. Let G' be the submatrix obtained from G be deleting the last $n-m$ rows. Define H' similarly. Let U' and V' be the submatrices obtained from U and V respectively by deleting the last $n-m$ rows and columns. Then we have the equations $H' = U'G'$ and $G' = V'H'$ where each matrix has dimension $m \times m$. We proceed to show that U' (and V') is the identity matrix. Since G' and H' are upper triangular with nonzero determinant we must have U' and V' upper triangular. Furthermore, $H' = U'G' = U'(V'H')$ which implies $U' = V'^{-1}$ whence U' and V' are unimodular. The diagonal entries of U' and V' must be 1 since the the diagonal entries of G' and H' lie in the same associate class of $\mathbf{R}$. So far we have shown that the diagonal elements of G' and H' are identical. Assume, to arrive at a contradiction, that U is not the identity matrix. Then let i be the index of the first row of U with nonzero off-diagonal element u_{ij} with $j > i$. Then entry h_{ij} in the i-th row j-th column of H' can be expressed as $h_{ij} = g_{ij} + u_{ij}g_{jj} = g_{ij} + u_{ij}h_{jj}$. Hence h_{ij} is congruent to g_{ij} modulo the common diagonal element g_{jj}. Since H' and G' are in HNF, h_{ij} and g_{ij} belong to the same complete set of residues of g_{jj} which implies $h_{ij} = g_{ij}$ — a contradiction since we assumed $u_{ij} \neq 0$. We conclude that U is the identity matrix and $G = H$. ∎

Theorem 1 *Let* $\mathbf{R}$ *be a PID and let* $A \in \mathbf{R}^{n\times m}$ *with full column rank. Then there exists a unique matrix* $H \in \mathbf{R}^{n\times m}$ *in HNF such that* $H \equiv_{\mathrm{L}} A$*. That is,* $UA = H$ *for some unimodular matrix* $U \in \mathbf{R}^{n\times n}$*.*

The matrix H of Theorem 1 can be found by applying unimodular row operations to A. The method is similar to Gaussian row reduction over a field but with division replaced by a new unimodular row operation that works to "zero out" elements below the diagonal element in a particular column. Consider a pair of entries a_{jj} and a_{ij} with $i > j$, not both zero, in column j of A. $\mathbf{R}$ a PID implies there exists elements of $\mathbf{R}$, say p and q, such that $pa_{jj} + qa_{ij} = g$ where $g = \gcd(a_{jj}, a_{ij})$. Define $G = B_U(p, q, i, j)$ to be the $n \times n$ matrix, identical to I_n, except with $g_{jj} = p$, $g_{ji} = q$, $g_{ii} = a_{jj}/g$ and $g_{ij} = -a_{ji}/g$. The determinant of G is $pa_{jj}/g - (-qa_{ij}/g) = (pa_{jj} + qa_{ij})g = 1$, whence G is unimodular. Furthermore, $(GA)_{jj} = g$ and $(GA)_{ij} = 0$. For example

$$\begin{array}{ccc} G & A & H \\ \begin{bmatrix} p & q \\ -b/g & a/g \end{bmatrix} \begin{bmatrix} a \\ b \end{bmatrix} & = & \begin{bmatrix} g \\ 0 \end{bmatrix} \end{array}$$

shows the technique applied to a 2×1 matrix. We now have a list of four unimodular row operations: (R1) multiplying a row by a unit, (R2) adding a multiple of one row to another, (R3) interchanging two rows, and (R4) premultiplying by a matrix of the type B_U as described above. Note that row operations R1, R2 and R3 are also applied to a matrix by premultiplying by a unimodular matrix — namely the elementary matrix corresponding to the elementary row operation. We can now give a procedure in terms of unimodular row operations to reduce the matrix A of Theorem 1 to HNF.

Procedure 2.1: We consider each column j of A in turn for $j = 1, 2, \ldots, m$. When j is first equal to k, the first $k-1$ columns of A will be in HNF. For $j = 1, 2, \ldots, m$ perform the following steps: (1) If the diagonal entry in the j-th column is zero, use a row operation of type R3 to move a nonzero entry below the diagonal to the diagonal position. (2) Use row operations of type R4 to zero out entries below the diagonal in column j. (3) Use a row operation of type R1 to ensure the diagonal element in column i belongs to the given complete set of associates of $\mathbf{R}$. (4) Use row operations of type R2 to ensure off-diagonal elements in column i belong to the given complete set of residues modulo the diagonal element.

With the help of Procedure 2.1 and Lemma 1, the proof of Theorem 1 is straightforward.

Proof: (of Theorem 1) We prove by induction on $i = 1, 2, \ldots, m$ that procedure 1 is correct. Since only unimodular operations are applied to A throughout the procedure, A always has full column rank. In particular, at the start of the procedure there exists a nonzero entry in the first column. After steps (1) through (3) have been completed for column $i = 1$, the first column of A will be zero except for a nonzero leading entry belonging to the given complete set of associates of A. Assume that after completing steps (1) through (4) for columns $i = 1, 2, \ldots, k < m$, the first k columns of A are in HNF. Now consider stage $i = k + 1$. Since A has rank m and the first k columns of A are upper triangular, there will exist at least one entry either on or below the diagonal entry in column $k + 1$ that is nonzero. Now note that the row operations in step (2), (3) and (4) do not change any entries in the first k columns of A. After steps (2), (3) and (4) are completed for $i = k + 1$, the first $k + 1$ columns of the reduced matrix A will be in HNF. By induction, procedure 1 is correct. The uniqueness of the HNF matrix found by Procedure 1 follows from Lemma 1, for if $G \equiv_L A$ and $H \equiv_L A$ then $G \equiv_L H$ ■

The existence of a matrix H in HNF left equivalent to a given matrix A was first proved (constructively) by Hermite in 1851 [19] for the case $\mathbf{R} = \mathbb{Z}$. It is worth noting that there is no one consistent definition of HNF that appears in the literature. In particular, some authors restrict their definitions to the case of square nonsingular matrices, or, instead of reducing a matrix via unimodular row operations, all the definitions and procedure may alternatively be given in terms of unimodular column operations with the unimodular multiplier matrix acting as postmultiplier instead of a premultiplier. Also, the HNF matrix may be defined to be lower instead of upper triangular. All these variations have the same central ingredients: (1) triangularity, (2) diagonal elements belonging to a given complete set of associates, and (3) off-diagonal elements in each column (or row, as it were) belonging to a complete set of residues with respect to the diagonal element. Algorithms or theorems using an alternative definition for HNF in terms of column operations and/or lower triangularity are easily cast into the form of Definition 1.

In one sense this section is complete; we have defined the HNF and proved that it is a canonical form for matrices over PIDs. Furthermore, the proof of existence is constructive (Procedure 2.1) and leads directly to a deterministic algorithm for finding the HNF of an input matrix over a concrete domain such as the integers. However, some cautionary remarks are in order. The reader — who should now be convinced of the uniqueness of the HNF and the correctness of Procedure 2.1 — may be tempted to think of the HNF of an input matrix A as "the upper triangular matrix H obtained from A by applying unimodular row operations to zero out entries below and reduce entries above the diagonal". However appealing this "constructive" definition may be, there is a danger of becoming fixated on a particular method of obtaining the HNF H of a given A. In particular, the fast "practical" algorithms we present in §5 and §6 will depend on results obtained by considering H in a more abstract sense as a particular "basis for the lattice of A" rather than as a "unimodular triangularization" of A.

It is instructive to consider the rows of an $n \times n$ nonsingular matrix A over $\mathbf{R}$ as a basis for the lattice $\mathcal{L}(A) = \{\vec{x}A : \vec{x} \in \mathbf{R}^{1\times n}\}$. Recall that $\mathcal{L}(A)$ is the set of all linear combinations of the rows of A. A basis for $\mathcal{L}(A)$ is any linearly independent subset of $\mathcal{L}(A)$ which generates $\mathcal{L}(A)$. Let B be an $n \times n$ matrix with $\mathcal{L}(B) = \mathcal{L}(A)$. Then B generates the rows of A and vice versa whence $K_1 B = A$ and $K_2 A = B$ with K_1, K_2 over $\mathbf{R}$. But then A, B non-singular implies that $K_2 = {K_1}^{-1}$. This shows that any two bases for $\mathcal{L}(A)$ are related by a unimodular transformation. In particular, every basis for $\mathcal{L}(A)$ is related via a unimodular transformation to the unique basis satisfying the requirements of Definition 1 — namely the HNF H of A.

2.2 The Smith Normal Form

The HNF deals only with row operations and the notion of left equivalence. Another equivalence relation for matrices over PIDs is possible if we allow both column and row operations. Two matrices A and B in $\mathbf{R}^{n\times m}$ are said to be *equivalent* (write $A \equiv B$) if one is obtainable from the other via unimodular row and column transformations (i.e. $A = UBV$ for some unimodular matrices U and V over $\mathbf{R}$). The key point here is that $A = UBV$ implies $B = U^{-1}AV^{-1}$ where U^{-1} and V^{-1}, being unimodular, are also over $\mathbf{R}$. It follows that equivalence is an equivalence relation for matrices over $\mathbf{R}$. The SNF provides a canonical form for matrices over $\mathbf{R}$ with respect to equivalence.

Definition 2 *Let* $\mathbf{R}$ *be a PID. A matrix* S *over* $\mathbf{R}$ *is said to be in* Smith normal form *if: (1)* S *is diagonal; (2) each diagonal entry belongs to a given complete set of nonassociates of* $\mathbf{R}$*; (3) each diagonal entry (except for the last) divides the next.*

Note that if $S \in \mathbf{R}^{n\times m}$ is in SNF with rank r, then, as simple consequence of the divisibility conditions on the diagonal entries, we have $\mathrm{diag}(S) = [s_1, s_2, \ldots, s_r, 0, \ldots, 0]$, where $s_i \neq 0$ for $1 \leq i \leq r$.

Proving that a given matrix $A \in \mathbf{R}^{n\times m}$ is equivalent to a *unique* matrix in SNF will require some facts about matrices over PIDs. Let C_i^r denote the set of all i element subsequences of $[1, 2, \ldots, r]$. For a matrix $A \in \mathbf{R}^{n\times m}$ and $I \in C_i^n$, $J \in C_i^m$, let $A_{I,J}$ denote the $i \times i$ matrix formed by the intersection of rows I and columns J of A; we refer to a matrix so defined as an $i \times i$ minor of A. The i-th *determinantal divisor* of A, denoted by $s^*(A, i)$ for $1 \leq i \leq \min(n, m)$, is the gcd of the determinants of all $i \times i$ minors of A. Let $B = UAV$ for any, not necessarily unimodular, square matrices $U \in \mathbf{R}^{n\times n}$ and $V \in \mathbf{R}^{m\times m}$ over $\mathbf{R}$. Then, for any $i \times i$ minor of B, say $B_{I,J}$ where $I \in C_i^n$ and $J \in C_i^m$, the Cauchy-Binet formula [13, §1.2.4] states that

$$\det(B_{I,J}) = \sum_{K\in C_i^n, L\in C_i^m} \det(U_{I,L}) \det(A_{K,L}) \det(V_{K,J})$$

which expresses $\det(B_{I,J})$ as a linear combination of the determinants of $i \times i$ minors of A. It follows that $s^*(A, i)$ divides $s^*(B, i)$. Now, if we have $A = WBX$ as well as $B = UAV$ for square matrices U, V, W, X over $\mathbf{R}$, then we must have $s^*(B, i) \mid s^*(A, i)$ as well as $s^*(A, i) \mid s^*(B, i)$ whence $s^*(A, i) = s^*(B, i)$.

Lemma 2 *Let* $\mathbf{R}$ *be a PID. If* T *and* S *are matrices over* $\mathbf{R}$*, both in SNF, such that* $T \equiv S$*, then* $T = S$*.*

Proof: Assume that S and T are in $\mathbf{R}^{n\times m}$. Let U and V be unimodular matrices such that $UTV = S$ and $T = U^{-1}SV^{-1}$. U and V unimodular implies that T and S have the same rank r. Let $S = \mathrm{diag}[s_1, s_2, \ldots, s_r, 0, 0, \ldots, 0]$ and $T = \mathrm{diag}[t_1, t_2, \ldots, t_r, 0, 0, \ldots, 0]$. Since $S = UTV$ and $T = U^{-1}SV^{-1}$ we have $s^*(S, i) = s^*(T, i)$ for $1 \leq i \leq r$. Note that $s^*(S, i) = \prod_{1\leq i\leq i} s_i$. It follows that $s^*(S, 1) \simeq s_1$ and $s_i \simeq s^*(S, i)/s^*(S, i-1)$ for $2 \leq i \leq r$. Similarly, $s^*(T, 1) \simeq t_1$ and $t_i \simeq s^*(T, i)/s^*(T, i-1)$ for $2 \leq i \leq r$ and we conclude that $s_i \simeq t_i$ for $1 \leq i \leq r$. Since s_i and t_i are chosen to belong to the same complete set of associates of $\mathbf{R}$ we have $s_i = t_i$ for $1 \leq i \leq r$.

Theorem 2 *Let* $\mathbf{R}$ *be a PID and* A *an element of* $\mathbf{R}^{n\times m}$*. Then there exists a unique matrix* $S \in \mathbf{R}^{n\times m}$ *in SNF such that* $A \equiv S$*. That is, there exist unimodular matrices* $U \in \mathbf{R}^{n\times n}$ *and* $V \in \mathbf{R}^{m\times m}$ *such that* $S = UAV$*.*

The existence of a matrix S in SNF equivalent to a given matrix A was was first proved by Smith in [33]. For the HNF reduction procedure, a new row operation was defined in terms of a unimodular matrix B_U that worked to zero out an entry below the diagonal in a given column. For reduction to SNF, we will need a corresponding unimodular column operation to zero out entries to the right of the diagonal in a given row. For a pair of entries a_{ii} and a_{ij} with $j > i$, not both zero, in row i of A, let p and q be such that $g = pa_{ii} + qa_{ij}$ where $g = \gcd(a_{ii}, a_{ij})$. Define $G = B_V(p, q, i, j)$ to be the $m \times m$ matrix, identical to I_m, except with $g_{ii} = p$, $g_{ji} = q$, $g_{ij} = -a_{ij}/g$ and $g_{jj} = a_{ii}/g$. Then G is unimodular and $(AG)_{ii} = g$ and $(AG)_{ij} = 0$. A procedure similar to the HNF reduction method given earlier can be used to reduce a matrix to SNF. We require four unimodular column operations analogous to the four unimodular row operations R1, R2, R3 and R4 used for the HNF reduction — these are: (C1) multiplying a column by a unit, (C2) adding a multiple of one column to another, (C3) interchanging two columns, or (C4) postmultiplying by a matrix of the type B_V as described above. We can now give a procedure to reduce the matrix A of Theorem 2 to SNF.

Procedure 2: Say $A \in R^{n\times m}$ is of rank r. Each diagonal entry s_i of the reduced matrix is found in succession for $i = 1, \ldots, r$. At stage i of the reduction the matrix has the form

$$\left[\begin{array}{ccc|c} s_1 & & & \\ & \ddots & & 0 \\ & & s_i & \\ \hline & 0 & & A^\star \end{array}\right]$$

The reduction proceeds on matrix $A^\star$. The leading entry in the first row and first column of $A^\star$ is called the *pivot.* If $A^\star$ is the zero matrix then the reduction is finished. If not, permute the rows and columns of A so that the pivot is nonzero. Perform the following steps: (1a) Apply column operations of type C2 to zero out all off-diagonal entries in the pivot row that are divisible by the pivot. (1b) Apply column operations of type C4 to zero out any remaining nonzero off-diagonal entries in the pivot row. (2a) Apply row operations of type R2 to zero out all off-diagonal entries in the pivot column that are divisible by the pivot. (2b) Apply row operations of type R4 to zero out any remaining nonzero off-diagonal entries in the pivot column. (3) Step 2b may have introduced new nonzero entries in the pivot row. If so, return to step 1a, otherwise go to step 4. (4) At this point, all entries in the pivot row and column except for the pivot are zero. If there exists an entry in $A^\star$ that is not divisible by the pivot, then add the row containing such an entry to the pivot row and return to step 1a. Otherwise go to step 5. (5) If necessary, use row operation of type R1 to ensure that the pivot entry belongs to the given complete set of associates of $\mathbf{R}$.

Proof: (of Theorem 2) We prove first that algorithm 2 is correct. First we show that steps 1 through 5 successfully zero out all entries in the pivot row and column of $A^\star$ except for the pivot entry. It is clear that steps 1a and 1b will zero out all entries to the right of the pivot in row 1. Similarly, steps 2a and 2b will zero out all entries below the pivot in column 1. The algorithm may jump back to step 1a at step 3 or step 4 so it remains to show that step 5 is reached. Let p_i be the value of the pivot entry of $A^\star$ when step 1a is restarted for the i-th time. We claim that a sequence of three equal pivots such as $p_k = p_{k-1} = p_{k-2}$ for some $k > 2$ cannot occur. If $p_k = p_{k-1}$ then neither column nor row operations of type C2 and R2 were applied on pass $k-1$, and this, together with $p_{k-1} = p_{k-2}$ implies that step 4 in pass $k-2$ must not have introduced a new entry into row 1 not divisible by the pivot and step 5 would have been reached — a contradiction. Next, assume, to arrive at a contradiction, that the algorithm does not terminate. Consider the ascending chain of ideals $\langle p_1 \rangle \subseteq \langle p_2 \rangle \subseteq \cdots$. There cannot exist an infinite chain of ideals, each properly contained in the next, in a PID, so there exists a k such that $\langle p_1 \rangle \subseteq \langle p_2 \rangle \subseteq \cdots \langle p_k \rangle = \langle p_{k+1} \rangle = \langle p_{k+2} \rangle = \cdots$. In particular, there must exist a sequence of three equal pivots which implies step 5 would have been reached. Lastly, consider stage i of the reduction for some $1 \leq i \leq r-1$. Diagonal element s_i, found during stage $i-1$, divides all entries in $A^\star$. Any row or columns operations applies to $A^\star$ will produce new entries that are linear combinations of entries divisible by s_i — hence any new entries arising in $A^\star$ will be divisible by s_i. This shows that each diagonal element found will divide the next.

2.3 Domains of Computation

Although we have defined the HNF and SNF for matrices over $\mathbf{R}$ a general PID, the most interesting domains from an applications point of view are $\mathbf{R} = \mathbb{Z}$ and $\mathbf{R} = \mathbf{F}[x]$ where $\mathbf{F}$ is a field. The field $\mathbf{F}$ should be computable — we need to be able to represent elements of $\mathbf{F}$ and compute field operations. Typically, $\mathbf{F}$ will be the rationals $\mathbf{Q}$ or a finite field $\mathrm{GF}(p)$, p prime. The special relationship between matrices over the integers and finitely presented Abelian groups has already been noted. Polynomial matrices, especially over $\mathbf{Q}[x]$, play an important role in linear systems theory.

2.3.1 The Extended Euclidean Problem

A common characteristic of the PID's $\mathbb{Z}$ and $\mathbf{F}[x]$ is that they are Euclidean. A key step in the procedure we gave for HNF reduction is solving the extended Euclidean problem: given $a, b \in \mathbf{R}$, find elements $p, q, g \in \mathbf{R}$ such that g is the gcd of a and b and

$$pa + qb = g. \tag{2.1}$$

$\mathbf{R}$ being a PID guarantees that equation (2.1) has a solution for p and q. When $\mathbf{R}$ is also a Euclidean domain we can apply the extended Euclidean algorithm to compute a solution to (2.1), whereas for $\mathbf{R}$ a PID no general method exists. The need for solving the extended Euclidean problem is not particular to Procedure 2.1 given in section 2.1 for HNF reduction; rather, the Euclidean problem is fundamental to the HNF reduction problem. Solving the extended Euclidean problem of (2.1) can be cast into the problem of finding the HNF of a 2×2 matrix. Define the matrix

$$A = \begin{bmatrix} a & 0 \\ b & 1 \end{bmatrix} \quad \text{with HNF} \quad H = \begin{bmatrix} h_{11} & h_{12} \\ 0 & h_{22} \end{bmatrix}. \tag{2.2}$$

H the HNF of A implies that $H = UA$ for some unimodular matrix U. A nonsingular implies U is unique. Solving the equation $UA = H$ for U yields

$$U = \begin{bmatrix} (h_{11} - h_{12}b)/a & h_{12} \\ -b/h_{11} & h_{22} \end{bmatrix}$$

where each entry is integral. Set $p = (h_{11} - h_{12}b)/a$ and $q = h_{12}$. Then U unimodular implies $\det(U) = p(a/h_{11}) + q(b/h_{11}) = \pm 1$ whence h_{11} is the gcd of a and b.

Recall the basic feature of a Euclidean domain $\mathbf{R}$: a valuation (or size) function $v : \mathbf{R} \setminus \{0\} \to \mathbb{N}$, where $\mathbb{N}$ is the set of nonnegative integers. For all $a, b \in \mathbf{R}$ with $b \neq 0$, the function v has the property: (1) $v(ab) \geq v(a)$, and (2) there exist elements $q, r \in \mathbf{R}$ such that $a = bq + r$ where either $r = 0$ or $v(r) < v(b)$. The Euclidean algorithm to find the gcd of a and b, $b \neq 0$, works by computing a remainder sequence: first r is found such that $a = bq + r$ and $v(r) < v(b)$, then r_1 is found such that $b = q_1 r + r_1$ and $v(r_1) < v(r)$, then r_2 is found such that $q_1 = q_2 r_1 + r_2$, and so on. The last nonzero remainder will be the gcd of a and b. The idea of the Euclidean algorithm is applied implicitly in the following procedure which reduces an integer matrix to SNF with only elementary row and column operations.

Procedure 3: Assume A is an integer matrix with rank r. The reduction proceeds in stages for $i = 1, \ldots, r$. At the end of stage i, the $i \times i$ principal minor of A is in SNF. The reduction for a general stage can be understood by considering the first stage when $i = 1$. If A is the zero matrix then the reduction is finished. If not, permute the rows of A using operation R3 to obtain a nonzero entry in the first row first column (the pivot) . If there exists an off-diagonal nonzero entry in the first row (column) use an operation of type C2 (R2) to reduce the size of the off-diagonal element to be less than the size of the pivot. If the reduced entry is nonzero then make it the new pivot. By applying elementary row (and symmetrically column) operations in this manner, A is transformed to a matrix with all entries in the pivot row and column zero except for the pivot entry. If the pivot does not divide all other entries in A, add a row which contains an entry not divisible by the pivot to the first row and continue zeroing out elements in the first row and column. Since the size of the pivot is monotonically decreasing this process terminates. Finally, use a row operation of type R1 to ensure the last nonzero pivot belongs the the proscribed complete set of associates.

Chapter 3

Advanced Computational Techniques

Up until now we have used the general expression *computing the HNF* or *computing the SNF* to refer to the actual computational problem that is the focus of this thesis. Before proceeding with a survey of computational methods we need to define more precisely the problems to be solved. In particular, the theory of Smith normal forms leads to two problems that from a computational perspective need to be distinguished (one will be seen to be much more difficult than the other).

Let A be an input matrix over a principal ideal domain $\mathbf{R}$. The first problem is to compute the Smith normal form S of A. Henceforth, let SmithForm over $\mathbf{R}$ be the problem of computing Smith normal forms over $\mathbf{R}$. Recall that S is the Smith normal form *of* A (rather than just a matrix in Smith normal form) precisely because there exist unimodular matrices U and V (of appropriate dimensions) such that $UAV = S$; these are sometimes sometimes referred to as pre- and post-multipliers for the Smith form. Henceforth, let SmithFormWithMultipliers be the problem of computing candidates for these pre- and post-multipliers. In the case of Hermite normal forms, let HermiteForm over $\mathbf{R}$ be the problem of computing Hermite normal forms over $\mathbf{R}$ and let HermiteFormWithMultipliers be the problem of computing a candidate for a pre-multiplier for the HNF.

The problem HermiteFormWithMultipliers reduces to solving HermiteForm, an adjoint computation, and a matrix multiplication. If A is square nonsingular, then $U \leftarrow 1/\det(A)HA^{\mathrm{adj}}$ expresses U in terms of the adjoint of input matrix A and the HNF H of A found by solving HermiteForm. Now consider the case where $A \in \mathbf{R}^{n\times m}$ of rank m with m strictly less than n. Let U_p be an $n \times n$ unimodular matrix such that U_pA consists of a permutation of the rows of A with the first m rows of U_pA linearly independent. Let A_1 be the $m \times m$

matrix consisting of the first m rows of U_pA and let A_2 consist of the last $(n-m)$ rows. Then the $n \times n$ matrix

$$A_s = \begin{bmatrix} A_1 & 0 \\ A_2 & I_{n-m} \end{bmatrix},$$

obtained by permuting the rows of A and augmenting with I_{n-m}, is non-singular. Now solve HermiteFormWithMultipliers to find the $n \times n$ unimodular matrix U such that $UA_s = H_s$ is the HNF of A_s. Let $(UU_p)A = H$. Then H consists of the first m columns of H_s. Take H to be the Hermite normal form of A. Uniqueness of H_s implies uniqueness of H.

Knowing the Smith normal form S of A does not help at all in general to find candidates for pre- and post-multipliers. Also, HermiteForm and SmithForm admit unique solutions — for SmithFormWithMultipliers this is very far from being the case. Consider the following example.

$$A = \begin{bmatrix} -147 & 84 & 532 & -44 \\ 111 & -78 & -470 & 120 \\ -607 & 382 & 2362 & -392 \\ -2436 & 1528 & 9456 & -1544 \end{bmatrix} \quad \text{has SNF} \quad S = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 2 & 0 & 0 \\ 0 & 0 & 4 & 0 \\ 0 & 0 & 0 & 8 \end{bmatrix}$$

and

$$S = \begin{bmatrix} -8 & 11 & 20 & -4 \\ -1 & 4 & 13 & -3 \\ -2 & 3 & -3 & 1 \\ 0 & 0 & -4 & 1 \end{bmatrix} A \begin{bmatrix} -19 & -6 & 16 & 4 \\ 9 & 8 & -15 & -4 \\ -7 & -3 & 7 & 2 \\ -4 & -1 & 3 & 1 \end{bmatrix} \tag{3.1}$$

$$= \begin{bmatrix} 1979 & -2568 & -4288 & 832 \\ -3699 & 4725 & 7700 & -1480 \\ -13474 & 17833 & 30659 & -6014 \\ 42176 & -55108 & -92976 & 18111 \end{bmatrix} A \begin{bmatrix} -149 & -53784 & -51084 & -51452 \\ 152 & 50582 & 48039 & 48382 \\ -70 & -23961 & -22757 & -22920 \\ -31 & -10587 & -10055 & -10127 \end{bmatrix} \tag{3.2}$$

In a computer algebra setting, we will be much happier to have the pre- and post-multipliers of equation (3.1) — especially when subsequent computations will depend on them — rather than those of (3.2).

3.1 The Classical Algorithm and Improvements

The classical algorithm for HermiteForm provides a starting point for many other HermiteForm algorithms. In particular, the modulo determinant algorithms of §3.2 can be described as a modification of the classical algorithm; for this reason, it is useful to present here detailed pseudocode for a version of the classical algorithm.

Assume for this section that A is an $n \times m$ rank m integral matrix with HNF H. The classical algorithm to compute H from A can be described succinctly as Gaussian row reduction with division replaced by the extended Euclidean algorithm (EEA). For each pair of entries a_{jj} and a_{ij} with $i > j$, the algorithm uses EEA to compute integers p and q, such that $g = pa_{jj} + qa_{ij}$ where $g = \gcd(a_{jj}, a_{ij})$. Define $G = B_U(p, q, i, j)$ to be the $n \times n$ matrix, identical to I_n, except with $g_{jj} = p$, $g_{ji} = q$, $g_{ii} = -a_{jj}/g$ and $g_{ij} = a_{ji}/g$. It is easily verified that G is unimodular. Furthermore, $(GA)_{jj} = g$ and $(GA)_{ij} = 0$. As an example of this technique, let

$$A = \begin{bmatrix} 14 & 3 & 6 \\ 18 & 5 & 13 \\ 3 & 2 & 4 \end{bmatrix}$$

and consider the pair of entries $a_{11} = 14$ and $a_{21} = 18$. We want a unimodular matrix G such that $(GA)_{21} = 0$ and $(GA)_{11} = g$ where $g = \gcd(14, 18)$. Using the EEA we obtain $g = 2$ and $2 = 4 \times 14 + (-3) \times 18$ whence $p = 4$ and $q = -3$. Then $G = B_U(p, q, 2, 1)$ is such that

$$G = \begin{bmatrix} 4 & -3 & 0 \\ 9 & -7 & 0 \\ 0 & 0 & 1 \end{bmatrix}, \text{ and } \begin{bmatrix} 4 & -3 & 0 \\ 9 & -7 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 14 & 3 & 6 \\ 18 & 5 & 13 \\ 3 & 2 & 4 \end{bmatrix} = \begin{bmatrix} 2 & -3 & -15 \\ 0 & -8 & -37 \\ 3 & 2 & 4 \end{bmatrix}$$

as desired. Note that $\det(G) = 4 \cdot (-7) - 9 \cdot (-3) = -1$ whence G is unimodular. The first stage of the algorithm, termed `ClassicTriangularize`, computes an upper triangular matrix T obtained from A by a unimodular transformation U.

```
ClassicTriangularize
  # Initialize T and U.
  T := A;
  U := I_n;
  # Set t_jj to be gcd(t_jj, t_{j+1 j}, ..., t_nj).
  for j := 1 to m do
      # Zero out entries below diagonal entry t_jj.
      for i := j + 1 to n do
          find g := gcd(t_jj, t_ij) and p, q such that g = p t_jj + q t_ij;
          G := G_1(p, q, i, j);
          T := GT;
          U := UG;
```

```
8       od;
9 od;
```

Remarks:
(1) As already mentioned, line 4 is accomplished using the extended Euclidean algorithm.
(2) In practice, lines 6 and 7 would be accomplished by performing on T and U the row operations indicated by matrix G rather than performing the matrix multiplication.
(3) Note that lines 1 and 7 are performed only if computation of the unimodular multiplier matrix is desired.

The second stage of the classical algorithm, termed `ClassicReduce`, continues with the upper triangular matrix T and reduces the entries t_{ij} above the diagonal entry modulo the diagonal entry t_{jj}.

```
ClassicReduce
  # Consider diagonal entry $t_{jj}$.
  for $j := 1$ to $m$ do
      # Reduce modulo $t_{jj}$ entries above diagonal.
      for $i := 1$ to $j - 1$ do
          $p := (t_{ij} - \mathrm{mod}(t_{ij}, t_{jj}))/t_{jj}$;
          set $\mathrm{ROW}(i,T) = \mathrm{ROW}(i,T) - p\,\mathrm{ROW}(j,T)$;
      od;
  od;
```

For clarity, the triangularization and reduction phases have been described separately — first `ClassicTriangularize` and then `ClassicReduce` are used in succession. In an actual implementation, however, these operations are usually interleaved. Let `ClassicTriangularize(`j`)` and `ClassicReduce(`j`)` perform similar operations as the original routines but restricted to column j. The complete algorithm can now be written as:

```
Algorithm ClassicHermite
  Input(A) # an $n \times m$ rank $m$ integral matrix
  Output(U,H) # an $n \times n$ unimodular matrix $U$ together with $UA = H$,
                the Hermite normal form of $A$
    # Initialize the work matrices $U$ and $B$.
  $U := I_m$;
  $B := A$;
  for $j := 1$ to $m$ do
```

```
    # Perform row operations on matrices U and B.
    ClassicTriangularize(j);
    ClassicReduce(j);
od;
H := B;
```

ClassicHermite requires $O(n^3)$ arithmetic operations: each off-diagonal entry position in A is modified using two $O(n)$ row operations in ClassicTriangularize (for positions below diagonal) and using one row operation in ClassicReduce (for positions above diagonal). To bound the number of bit operations we need also to bound the size of integers occurring during the computation. Consider the case of an $n \times n$ rank n matrix A with HNF H. Then H triangular implies $|\det(A)| = \text{trace}(H)$, ensuring that the largest diagonal entry in H is at most $|\det(A)|$. Since entries h_{ij} above the diagonal are less than h_{jj}, we conclude that the largest entry in H will be at most $|\det(A)|$. The usual Hadamard bound [28, Chapter II, §4.1.7] gives $|\det(A)| \le n^{n/2}||A||^n$ where $||A||$ is the largest magnitude coefficient of A — integers of this magnitude can be represented using $\lceil n((\log_2 n)/2 + \log(||A||))\rceil + 1$ bits. Unfortunately, this nice bound on the size of entries of the final output matrix H does not ensure good behavior during the course of the algorithm.

The algorithm ClassicHermite is impractical because entries in the matrix under computation can grow exceedingly large. In [16], Hafner and McCurley give an example of a 20×20 matrix A with positive entries chosen from $\{0, 1, \ldots, 10\}$. After completing ClassicTriangularize with input A, the resulting triangular matrix T contains an entry larger than 10^{5011} even though $|\det(A)| < 10^{20}$. In another example, Domich, Kannan and Trotter [11] use a version of ClassicHermite on an 8×8 matrix with entries having magnitude less than 2^{16}; intermediate computations involved numbers larger in magnitude than 2^{432}.

We give here a worked example showing the type of expression swell that can occur. Consider the following 8×5 randomly generated integral matrix.

$$G = \begin{bmatrix} -85 & -55 & -37 & -35 & 97 \\ 49 & 63 & 57 & -59 & 45 \\ 43 & -62 & 77 & 66 & 54 \\ -50 & -12 & -18 & 31 & -26 \\ -91 & -47 & -61 & 41 & -58 \\ 94 & 83 & -86 & 23 & -84 \\ -53 & 85 & 49 & 78 & 17 \\ -86 & 30 & 80 & 72 & 66 \end{bmatrix} \tag{3.3}$$

We compute the HNF of G using a procedure described by `ClassicHermite` and examine the results of intermediate computations. Denote by B_j, $1 \le i \le 6$, the state of matrix B (the work matrix) after the correct form of columns $1, 2, \ldots, j$ has been computed. Then $B_0 = G$ and

$$B_1 = \begin{bmatrix} 1 & -937 & 217 & 941 & 117 \\ & 5747 & -1322 & -5771 & -711 \\ & 79700 & 18408 & 79950 & 10042 \\ & -46862 & 10832 & 47081 & 5824 \\ & -85314 & 19686 & 85672 & 10589 \\ & 88161 & -20484 & -88431 & -11082 \\ & -49576 & 11550 & 49951 & 6218 \\ & -80552 & 18742 & 80998 & 10128 \end{bmatrix}$$

$$B_2 = \begin{bmatrix} 1 & 0 & 29056587 & -32365850 & 71025654 \\ & 1 & 31010 & -34543 & 75801 \\ & & 427376 & -476050 & 1044674 \\ & & 1453201452 & -1618706985 & 3552192286 \\ & & 2645606826 & -2946915830 & 6466897103 \\ & & -2733893094 & 3045256992 & -6682703043 \\ & & 1537363310 & -1712453817 & 3757916594 \\ & & 2497936262 & -2782426738 & 6105932280 \end{bmatrix}$$

$$B_3 = \begin{bmatrix} 1 & 0 & 1 & -3894548241929407974350 & 4839209651505280436435 \\ & 1 & 0 & -4156370641142457043 & 5164539677620032386 \\ & & 2 & 268066471534500 & -333088660278617 \\ & & & 8700810 & -10811278 \\ & & & 536132943474305 & -666177321060854 \\ & & & 366432537630561611628492 & -455314394012718248788542 \\ & & & -206057778989151562051317 & 256039142644703834587729 \\ & & & -334806479936211949446238 & 416017121485484319687107 \end{bmatrix}$$

$$B_4 = \begin{bmatrix} 1 & 0 & 1 & 0 & -18376609178566059176568112182999160283677900228302465 \\ & 1 & 0 & 0 & -19612030492065843359514869894528011638779778516656 \\ & & 2 & 0 & 1264884262629195766961856490960008462774464383 \\ & & & 1 & -4718547065541562489662839409094 \\ & & & & -98375990 \\ & & & & 11796367663853906224157082426498 \\ & & & & -972293328381272933761167250141489411131870330949889069 \\ & & & & -1579800133427312912106981124086461701228582712121601265 \end{bmatrix}$$

$$B_5 = \begin{bmatrix} 1 & 0 & 1 & 0 & 0 \\ & 1 & 0 & 0 & 0 \\ & & 2 & 0 & 0 \\ & & & 1 & 0 \\ & & & & 1 \\ & & & & \\ & & & & \\ & & & & \end{bmatrix}.$$

Remark: Note that B_5 is the HNF of G.

It remains on open question whether `ClassicHermite` (or a variation of it) has an exponential lower bound on the number of digits in intermediate computations. (Such a bound could be expressed in terms of the dimensions of the input matrix and the size in bits of the largest entry.) However, as the examples given above show, experience indicates that `ClassicHermite` is impractical even for moderately sized matrices.

Various modifications and improvements of `ClassicHermite` appear in [5, 7, 4, 12, 26, 10], although this list is not exhaustive. An algorithm for triangulating an integer matrix is given by Blankinship in [5]. This article appeared in 1966 before the use of arbitrary precision integer arithmetic was widespread. Referring to an input matrix $A \in \mathbb{Z}^{n\times m}$ of rank m, Blankinship writes, "Overflow is generally dependent upon the magnitude of the greatest common divisor of all $r \times r$ minors contained in A, as this number, or a large divisor of it will appear in the r-th row of the work matrix". However, the 8×5 integer matrix of (3.3) has magnitude of the greatest common divisor of all 5×5 minors equal to 2; intermediate expressions are guaranteed to be larger than this since the original matrix has entries two digits in length.

Bradley gives an algorithm in [7] that find the complete HNF but he also doesn't give bounds on the size of intermediate integers. Bradley writes, "The possible increase in the magnitude of the elements of the matrix as the algorithm proceeds is a serious computational consideration ...".

In [26], Kannan and Bachem present a modification of `ClassicHermite` that works by computing for $j = 1, 2, \ldots, m$, the HNF of the $j \times j$ principal submatrix of an $m \times m$ rank m input matrix. (The classical algorithm determines the correct form for the columns $j = 1, \ldots, m$ rather than for principal submatrices.) Kannan and Bachem were able to prove a polynomial running time for their algorithm. However, in [11], Domich, Kannan and Trotter present experimental evidence that the Kannan/Bachem algorithm still suffers from intermediate expression swell.

There is one overriding reason that has motivated the search for a better HNF algorithm: The integers in the output matrix H are small compared to the input matrix A. In particular, we have good *a priori* bounds on the size of integers in H. If $A \in \mathbb{Z}^{n \times m}$ has rank m and d is the gcd of a subset of determinants of $m \times m$ minors of A (not all singular), then entries in H will be bounded in magnitude by d. In the next section, an algorithm for HermiteForm is presented that bounds the magnitude of all intermediate integers occurring during the reduction by d.

3.2 Modular Methods

In [11, 21, 16], algorithms for HermiteForm are presented which perform all calculation modulo d where d is a multiple of the gcd of all $m \times m$ minors of an input matrix $A \in \mathbb{Z}^{n \times m}$ with rank m. The class of modulo determinant algorithms are an important improvement over the `ClassicHermite` algorithm from a practical viewpoint. In [21], Iliopoulos presents a modulo d algorithm together with a worst-case complexity bound (for bit operations) which improves by a factor of $O(s^{3-\epsilon})$ (for any $\epsilon > 0$ where s is the size of input matrix in bits) the best known non-modulo algorithm (given by Chou and Collins in [10]).

We require the notion of the determinant of the lattice $\mathcal{L}(A)$ — denoted by $\det(\mathcal{L}(A))$. In general, $\det(\mathcal{L}(A)) = |det(B)|$ where B is a basis matrix for $\mathcal{L}(A)$. Note that B a basis for $\mathcal{L}(A)$ implies B is square and non-singular since A is $n \times m$

with rank m. In particular, if A is square and non-singular, then $\det(\mathcal{L}(A)) = |\det(A)| = \det(H)$. For an $n \times m$ rank m matrix A, we can choose B to be the first m rows of H, the HNF of A. Another useful fact is: $\det(\mathcal{L}(A))$ equals the gcd of all the determinants of $m \times m$ submatrices of A.

We have already noted that for $n \times n$ rank n matrices A, the entries in the HNF H of A are bounded in magnitude by $|\det(A)|$. In general, entries in H are bounded by $\det(\mathcal{L}(A))$. Modular algorithms given in [11, 21, 16] require as input a positive integer d — a multiple of $\det(\mathcal{L}(A))$. In general, modulo determinant algorithms require three steps: (1) computing a suitable modulant d; (2) computing the HNF; (3) computing a unimodular multiplier matrix. Note that step (3) may be omitted if a transformation matrix is not required. We consider first the actual computation of the HNF and leave steps (1) and (3) until later.

Theorem 3 and Corollary 1 will suffice to motivate our first modulo determinant algorithm. Theorem 3 restates for rectangular matrices a result given for square non-singular matrices by Domich, Kannan and Trotter in [11, prop. 2.5 and cor. 2.6].

Theorem 3 *Let A be an $n \times m$ rank m integer matrix with HNF $H = [h_{ij}]$. Let d be a positive integral multiple of $\det(\mathcal{L}(A))$. Define $d_1 = d$ and $d_{j+1} = d_j/h_{jj}$ for $j = 1, 2, \ldots, m-1$. Let e_j denote the j-th unit vector. Then $d_j e_j \in \mathcal{L}(A)$ for $1 \le j \le m$.*

Proof: Let H_1 be the $m \times m$ submatrix of H consisting of the first m rows of H. Then H_1 is a basis matrix for $\mathcal{L}(A)$. By Cramer's rule, d_1 $(= d)$ a multiple of $\det(H_1)$ implies $d_1 H_1^{-1}$ is an integral matrix whence $d_1 H_1^{-1} H_1 = dI_m$ has rows in $\mathcal{L}(A)$. Next, let H_2 be the $(m-1) \times (m-1)$ submatrix of H_1 consisting of the last m rows and columns of H. Then, $\det(H_2) = \det(H_1)/h_{11}$ implies d_2 is a multiple of $\det(H_2)$. Then, by Cramer's rule, $d_2 H_2^{-1}$ is integral whence $d_2 H_2^{-1} H_2 = d_2 I_{m-1}$. It follows that $d_2 e_2 \in \mathcal{L}(A)$. Continuing in this fashion for $j = 3, 4, \ldots, m$ yields the desired result. ∎

Corollary 1 *For the augmented matrix*

$$A' = \begin{bmatrix} A \\ dI_m \end{bmatrix} \tag{3.4}$$

we have $\mathcal{L}(A') = \mathcal{L}(A)$ whence the HNF A' equals

$$\begin{bmatrix} H \\ O \end{bmatrix}$$

Proof: A generates $\mathcal{L}(A)$ and $\mathcal{L}(dI_m) \subset \mathcal{L}(A)$. ∎

Our initial modulo determinant algorithm uses a modification of `ClassicHermite` with the $(n+m) \times m$ input matrix A' of (3.4). `ClassicHermite` initializes a work matrix B to be A' and computes the correct form for columns j of B for $j = 1, 2, \ldots, m$. Since column k of A' has zero entries below the $(n+k)$th row, algorithm `ClassicHermite` will not change rows $(n+k), n+k+1, ..., (n+m)$ of the work matrix B until `ClassicTriangularize(j)` is called with $j = k$. Thus, after `ClassicHermite` has completed for columns $j = 1, 2, \ldots, (k-1)$, the rows $(n+k), n+k+1, \ldots, (n+m)$ of B still correspond to the vectors $de_{k+1}, de_{k+2}, \ldots, de_m$. Recall that a valid elementary row operation is that of adding a multiple of one row to another; this row operation can be used with the appropriate de_j row to reduce the result of all computations, as they occur, to within d in magnitude. The original `ClassicHermite` algorithm augmented to use modulo d arithmetic as suggested constitutes a complete algorithm; for convenience, call this modification of the original algorithm `AugmentHermite`. The algorithm `AugmentHermite` was first given for square non-singular matrices by Domich, Kannan and Trotter in [11]; they mention also that the method of `AugmentHermite` was independently discovered by A. Schrijver in 1985.

`AugmentHermite` runs in polynomial time since we perform $O((n+m)^3)$ arithmetic operations on integers not exceeding d in magnitude. It still remains to demonstrate a method of obtaining d — a multiple of $\det(\mathcal{L}(A))$ — this we defer until later.

We continue with Theorem 4 and 5 which gives us a method of using modulo d reduction without having to resort to an augmented matrix such as A'. Let d and $d_1, d_2, \ldots, d_m$ be as in Theorem 3. Let `ModTriangularize` be identical to `ClassicTriangularize` except that all computations are performed using mod d arithmetic.

Theorem 4 *Let A be an $n \times m$ rank m integral matrix with HNF $H = [h_{ij}]$. Let $T = [t_{ij}]$ be any $n \times m$ upper triangular matrix with rows in $\mathcal{L}(A)$ and diagonal entries satisfying $t_{jj} = h_{jj}, 1 \leq j \leq m$. Then the HNF of T is H.*

Proof: The rows of T in $\mathcal{L}(A)$ implies that any integral linear combination of the rows of T are in $\mathcal{L}(A)$. In particular, H_T, the HNF of T, has rows in $\mathcal{L}(A)$. It follows that $LH = H_T$ for an $n \times n$ integral matrix L. Moreover, since the diagonal entries of H_1 and H are identical, we can choose L unimodular. But then $\mathcal{L}(H) = \mathcal{L}(H_T)$ from which it follows from the uniqueness of the HNF that $H = H_T$. ∎

Theorem 5 *Let A be an $n \times m$ rank m integral matrix with HNF $H = [h_{ij}]$. Let $T = [t_{ij}]$ be the upper triangular matrix obtained by applying* `ModTriangularize` *to A. Then $h_{jj} = \gcd(d_j, t_{jj})$ for $1 \leq j \leq m$ where d_j is as in Theorem 3.*

Proof: We omit the proof of Theorem 5 — although not very long or difficult, it is also not very illuminating. For a proof in the case of square matrices, the reader is referred to [11, prop. 2.7 and cor. 2.8].

We require one more fact which follows as a corollary of Theorem 3: If $r \in \mathcal{L}(A)$ with $r = (0, 0, \ldots, 0, r_k, r_{k+1}, \ldots, r_m)$ and p is any integer, then the vector obtained from r by multiplying by p and reducing all entries mod d_k is in $\mathcal{L}(A)$. To see this note that the mod d_k reductions can be effected by adding to r appropriate integral multiples of the vectors $d_j e_j$ for $j = k, k+1, \ldots, m$. More generally, for $r = (r_1, r_2, \ldots, r_m) \in \mathcal{L}(A)$ we can conclude that $(r_1 \bmod d_1, r_2 \bmod d_2, \ldots, r_m \bmod d_m) \in \mathcal{L}(A)$. The algorithm `ClassicHermite` works by computing the correct form for the successive columns $j = 1, 2, \ldots, m$. When $j = k, k+1, \ldots, m$, columns $1, 2, \ldots, k-1$ are no longer changed. It follows from the above discussion that at stage $j = k$ of the algorithm we can perform all computations using mod d_k reduction and still keep all rows in $\mathcal{L}(A)$.

We can now give a modulo determinant algorithm that doesn't require an augmented matrix. Let `ModTriangularize`(j) and `ModReduce`(j) be identical to the corresponding `CLASSIC` versions except that all computations be performed using modulo d arithmetic.

`Algorithm ModHermite`

  `Input(A,d) # an` $n \times m$ `rank` m `integral matrix and` d,
      `a positive integral multiple of det(`$\mathcal{L}(A)$`)`
  `Output(H) # the Hermite normal form of` A
    `# Initialize work matrix` B.
  $B := A$;
  `for` $j := 1$ `to` m `do`
      `# Perform row operations on` B `using mod` d $(= d_j)$
        `arithmetic.`
      `ModTriangularize(`j`);`
      `# Reconstruct diagonal entry` b_{jj}.
      `find` p, q `such that` $h_{jj} = p b_{jj} + q d_j$;
      `set` $\mathrm{ROW}(j, B) = p\,\mathrm{ROW}(j, B)$;
      `reduce entries in` $\mathrm{ROW}(j, B)$ `mod` d_j;
      `# Perform row operations on` B `using mod` d $(= d_j)$

```
        arithmetic.
    ModReduce(j);
    # Set d equal to d_{j+1} for next pass.
    d := d/h_{jj};
od;
H := B;
```

Algorithm `ModHermite`, coupled with a method of obtaining d, provides a complete solution for HermiteForm over $\mathbb{Z}$. Let $M(t)$ be an upper bound on the number of bit operations for multiplying two numbers with length t bits and let $||A||$ be largest magnitude of any coefficient of A . Currently, the best known running time for a HermiteForm algorithm is given by Hafner and McCurley in [16]. They demonstrate an asymptotic running time of $O(m^2 nM(m \log(m||A||)) \log(m \log(m||A||)))$ bit operations for both `ModHermite` and for the determination of a suitable modulant d [16, cor. 2.2].

Actually, determination of d is not the bottleneck in the Hermite normal form computation. Bareiss's fraction-Free Gaussian elimination can be used to triangularize an $n \times m$ rank m matrix in $O(m^2 nM(m \log ||A||))$ bit operations (see [21, Theorem 1.3]) and can be used to obtain a suitable multiple of $\det(\mathcal{L}(A))$.

The algorithm `ModHermite`, together with a determination procedure for d, was implemented in Maple V. When the input matrix A is square nonsingular we set $d \leftarrow \det(A)$. In general, Bareiss's fraction-free Gaussian elimination is used to obtain a triangularization of the input matrix with the property that the i-th pivot (the i-th diagonal entry of the triangularization) will be a determinant of an $i \times i$ minor of the input matrix. In particular, for a square input matrix, the last nonzero pivot will be the determinant. (For a discussion of fraction-free Gaussian elimination algorithms see [14] or the original articles by Bareiss [2, 3].) When the input matrix is rectangular, say $A \in \mathbb{Z}^{n \times m}$ with rank m, then we can set d to be the determinant of a nonsingular $m \times m$ minor of A. For example, consider the matrix G of (3.3). Applying the fraction-free Gaussian elimination procedure to zero out entries below the diagonal in the first 4 columns of G yields the matrix

$$T = \begin{bmatrix} -85 & -55 & -37 & -35 & 97 \\ 0 & -2660 & -3032 & 6730 & -8578 \\ 0 & 0 & 51696 & -274200 & 395524 \\ 0 & 0 & 0 & -3967680 & 3468824 \\ 0 & 0 & 0 & 0 & 983759900 \\ 0 & 0 & 0 & 0 & -1768625948 \\ 0 & 0 & 0 & 0 & -458519312 \\ 0 & 0 & 0 & 0 & 336004550 \end{bmatrix}$$

The last nonzero pivot, a determinant of a 5×5 minor of A, is $T_{5,5} = 98375900$. Note however that the trailing entries in rows 6 through 8 of T are determinants of 5×5 minors as well. As such, a better choice for d is $\gcd(983759900, -1768625948, -458519312, 336004550)$; this yields $d = 2$. The Bareiss method works very well in practice; for rectangular input matrices, the d found by computing the gcd of the trailing entries in the last reduction row is typically much smaller than the absolute value of a determinant of a single $m \times m$ minor of A.

At the end of §3.1, we gave an example computation using a non modular algorithm involving the 8×5 matrix G of (3.3). For comparison, we repeat computation but using the new algorithm `ModHermite`. Denote by B'_j, $1 \leq i \leq 6$, the state of matrix B' (the work matrix in the routine `ModHermite`) after the correct form of columns $1, 2, \ldots, j$ has been computed. Then

$$B'_1 = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 \\ & 0 & 0 & 0 & 0 \\ & 1 & 0 & 1 & 1 \\ & 0 & 0 & 1 & 0 \\ & 0 & 0 & 0 & 1 \\ & 1 & 0 & 1 & 0 \\ & 0 & 0 & 1 & 0 \\ & 0 & 0 & 0 & 0 \end{bmatrix}, \quad B'_2 = \begin{bmatrix} 1 & 0 & 1 & 0 & 0 \\ & 1 & 0 & 1 & 1 \\ & & 0 & 0 & 0 \\ & & 0 & 1 & 0 \\ & & 0 & 0 & 1 \\ & & 0 & 0 & 1 \\ & & 0 & 1 & 0 \\ & & 0 & 0 & 0 \end{bmatrix}, \quad B'_3 = \begin{bmatrix} 1 & 0 & 1 & 0 & 0 \\ & 1 & 0 & 1 & 1 \\ & & 2 & 0 & 0 \\ & & & 1 & 0 \\ & & & 0 & 1 \\ & & & 0 & 1 \\ & & & 1 & 0 \\ & & & 0 & 0 \end{bmatrix}$$

$$B_4' = \begin{bmatrix} 1 & 0 & 1 & 0 & 0 \\ & 1 & 0 & 0 & 0 \\ & & 2 & 0 & 0 \\ & & & 1 & 0 \\ & & & & 1 \\ & & & & 1 \\ & & & & 0 \\ & & & & 0 \end{bmatrix}, \quad B_5' \begin{bmatrix} 1 & 0 & 1 & 0 & 0 \\ & 1 & 0 & 0 & 0 \\ & & 2 & 0 & 0 \\ & & & 1 & 0 \\ & & & & 1 \\ & & & & \\ & & & & \\ & & & & \end{bmatrix}$$

Remark: The entries in the intermediate work matrices $B_1', B_2', \ldots, B_5'$ are all single digits because computations are being performed modulo d, which was computed to be 2.

3.2.1 HermiteFormWithMultipliers for Rectangular Input

In many applications, a unimodular multiplier matrix U with $UA = H$ is desired. When A is square nonsingular, then U will be the unique matrix $1/det(A)HA^{\mathrm{adj}}$; this can be found using standard techniques. More complicated is the case when the input matrix in strictly rectangular. The matrix equation $UA = H$ will not admit a unique solution for U in this case. Note that we could apply the standard linear systems techniques — which apply to matrices with coefficients from a field — to find a particular solution for U; unfortunately, this method will almost certainly find a U that contains non-integer entries or is non-unimodular.

The only solution that we are aware of that is offered in the literature is the technique mentioned at the beginning of this chapter. If A is $n \times m$ rank m with $m < n$, then Hafner and McCurley [16] suggest reducing to the square nonsingular case by forming the $n \times n$ matrix

$$A_s = \begin{bmatrix} A_1 & 0 \\ A_2 & I_{n-m} \end{bmatrix} \tag{3.5}$$

by permuting the rows of A such that A_1 is non-singular and augmenting with I_{n-m}. Although this method for finding U works, it can computationally expensive when A is nonsquare. For example, consider the case of a $km \times m$ rank m matrix A where k is some positive integer. If only H is required then the reduction proceeds on a $km \times m$ matrix. If a candidate for U is also required, then we must apply the reduction on the square matrix $A_s \in \mathbb{Z}^{km \times km}$ described by (3.5) and set

$U \leftarrow 1/\det(A_s)HA_s^{\mathrm{adj}}$. This seems exceedingly expensive since the matrix A_s is k times as large as A. Although the last $m(k-1)$ columns of A_s have only $m(k-1)$ nonzero entries, the algorithm `ModHermite` tends to fill in entries of these latter columns while reducing columns 1 through m. As an example, consider the case of a generic 9×3 matrix A. The matrix A_s will be 9×9. Let B_i be the state of the work matrix B of algorithm `ModHermite` after the reduction for column i is complete. Then

$$
\begin{aligned}
B_0 &= \begin{bmatrix}
* & * & * & & & & & & \\
* & * & * & & & & & & \\
* & * & * & & & & & & \\
* & * & * & * & & & & & \\
* & * & * & & * & & & & \\
* & * & * & & & * & & & \\
* & * & * & & & & * & & \\
* & * & * & & & & & * & \\
* & * & * & & & & & & *
\end{bmatrix} \\
B_1 &= \begin{bmatrix}
* & * & * & * & * & * & * & * & * \\
& * & * & & & & & & \\
& * & * & & & & & & \\
& * & * & * & & & & & \\
& * & * & * & * & & & & \\
& * & * & * & & * & & & \\
& * & * & * & & & * & & \\
& * & * & * & & & & * & \\
& * & * & * & & & & & *
\end{bmatrix} \\
B_2 &= \begin{bmatrix}
* & * & * & * & * & * & * & * & * \\
& * & * & * & * & * & * & * & * \\
& & * & & & & & & \\
& & * & * & & & & & \\
& & * & * & * & & & & \\
& & * & * & * & * & & & \\
& & * & * & * & & * & & \\
& & * & * & * & & & * & \\
& & * & * & * & & & & *
\end{bmatrix}
\end{aligned}
$$

$$B_3 = \begin{bmatrix} * & * & * & * & * & * & * & * & * \\ & * & * & * & * & * & * & * & * \\ & & * & * & * & * & * & * & * \\ & & & * & & & & & \\ & & & * & * & & & & \\ & & & * & * & * & & & \\ & & & * & * & * & * & & \\ & & & * & * & * & & * & \\ & & & * & * & * & & & * \end{bmatrix}$$

Remark: All entries that may be nonzero are denoted by $*$.

Note how the process of putting the j-th column in correct form tends to fill in the entries in column $m+j$ of the work matrix. In particular, the last 6 columns of B_3 still remain to be triangularized. In general, when the input is of size $n \times m$, after the first m columns have been triangularized, the work matrix will still contain a submatrix of size $(n-m) \times (n-m)$ that remains to be put into triangular form. A slight modification of algorithm `ModHermite` works to preserve the upper triangularity of the last $n-m$ columns and row of A. As presented, state j of algorithm `ModHermite` zeroes out subdiagonal entries in column j of the work matrix. In particular, the entry in i-th row j-th column of the work matrix is transformed to zero for $i = j+1, j+2, \ldots, n$. If, instead, the entry in the i-th row j-th column is zeroed with the row index being chosen in the order $i = j+1, j+2, \ldots, m, n, n-1, \ldots, n-m+1$, then the upper triangularity of the last $n-m$ columns of the work matrix is preserved. Reworking the previous example with the new reduction order yields:

$$B_0 = \begin{bmatrix} * & * & * & & & & & & \\ * & * & * & & & & & & \\ * & * & * & & & & & & \\ * & * & * & * & & & & & \\ * & * & * & & * & & & & \\ * & * & * & & & * & & & \\ * & * & * & & & & * & & \\ * & * & * & & & & & * & \\ * & * & * & & & & & & * \end{bmatrix}$$

$$
B_1 = \begin{bmatrix}
* & * & * & * & * & * & * & * & * \\
 & * & * & & & & & & \\
 & * & * & & & & & & \\
 & * & * & * & * & * & * & * & * \\
 & * & * & & * & * & * & * & * \\
 & * & * & & & * & * & * & * \\
 & * & * & & & & * & * & * \\
 & * & * & & & & & * & * \\
 & * & * & & & & & & *
\end{bmatrix}
$$

$$
B_2 = \begin{bmatrix}
* & * & * & * & * & * & * & * & * \\
 & * & * & * & * & * & * & * & * \\
 & & * & & & & & & \\
 & & * & * & * & * & * & * & * \\
 & & * & & * & * & * & * & * \\
 & & * & & & * & * & * & * \\
 & & * & & & & * & * & * \\
 & & * & & & & & * & * \\
 & & * & & & & & & *
\end{bmatrix}
$$

$$
B_3 = \begin{bmatrix}
* & * & * & * & * & * & * & * & * \\
 & * & * & * & * & * & * & * & * \\
 & & * & * & * & * & * & * & * \\
 & & & * & * & * & * & * & * \\
 & & & & * & * & * & * & * \\
 & & & & & * & * & * & * \\
 & & & & & & * & * & * \\
 & & & & & & & * & * \\
 & & & & & & & & *
\end{bmatrix}
$$

After the HNF H_s of A_s has been found, it remains to find the unimodular multiplier matrix U such that $UA_s = H_s$. Write

$$
H_s = \begin{bmatrix} N & M \\ 0 & P \end{bmatrix} \quad \text{and} \quad U = \begin{bmatrix} U_1 & U_3 \\ U_2 & U_4 \end{bmatrix}
$$

Then, we need to solve the matrix equation

$$
\begin{bmatrix} U_1 & U_3 \\ U_2 & U_4 \end{bmatrix} \begin{bmatrix} A_1 & 0 \\ A_2 & I_{n-m} \end{bmatrix} = \begin{bmatrix} N & M \\ 0 & P \end{bmatrix}
$$

for U_1, U_2, U_3 and U_4. This yields,

$$
U_3 = M,
$$

$$
\begin{aligned}
U_4 &= P, \\
U_2 &= -\frac{1}{d}P(A_2 A_1^{\mathrm{adj}}), \\
U_1 &= \frac{1}{d}(N A_1^{\mathrm{adj}} - M(A_2 A_1^{\mathrm{adj}}))
\end{aligned}
$$

where $d = \det(A_s) \quad (= \det(A_1))$.

Chapter 4

Polynomial Matrices

This chapter introduces the problem of computing normal forms of matrices over polynomial domains $\mathbf{F}[x]$, $\mathbf{F}$ a field. Our focus will be the HNF. As with matrices over the integers, our main task will be to control the growth in the size of the matrix entries during intermediate computations. We face a double challenge when $\mathbf{F}$ is a field of characteristic zero; in this case not only degrees but also size of coefficients of intermediate polynomials will need to be controlled. Despite the importance of the HNF and SNF in linear systems theory — and hence interest in being able to compute these forms — significant results have been forthcoming only very recently. For example, the fact that SmithFormWithMultipliers over $\mathbf{Q}[x]$ is in $\mathcal{P}$ — the class of problems that can be solved in polynomial time by a sequential deterministic algorithm — was first shown by Villard in a paper presented at ISSAC 93 [35].

On a more practical note, a problem with respect to HermiteForm over $\mathbf{Q}[x]$ that remained unanswered until recently was to establish good *a priori* bounds on the size in bits required to represent the rational coefficients of the matrices U and H corresponding to an input matrix A. Such a bound could be expressed in terms the dimension and size of entries of the input matrix A. The bulk of this chapter will be devoted to obtaining such a bound. Our approach is to develop a sequential deterministic solution for HermiteForm over $\mathbf{Q}[x]$ that converts the problem to that of solving a large linear system over $\mathbf{Q}$; this novel solution method will lead to fast method (given in §4.3.2) for obtaining size bounds on the length of coefficients in the premultiplier U and the HNF H for a given input matrix A. The careful analysis of §4.3 will prove very useful in §6.4 where a modular algorithm for HermiteForm over $\mathbf{Q}[x]$ is developed that require as input bounds for the size of coefficients appearing in U and H.

Recall some basic facts about the Euclidean domain $\mathbf{F}[x]$. A matrix $U \in$

$\mathbf{F}[x]^{n\times n}$ is unimodular if $\det(U) \in \mathbf{F} \setminus \{0\}$. The gcd of set of polynomials $\{p_1, p_2, \ldots, p_k\}$, not all zero, is the unique monic polynomial g such that $\langle g\rangle = \langle p_1, p_2, \ldots, p_k\rangle$. The main results of chapter 2 — which gave the basic properties of the HNF over a general PID $\mathbf{R}$ — are summarized here for the special case $\mathbf{R} = \mathbf{F}[x]$.

Hermite normal form. Let H be an n by m matrix over $\mathbf{F}[x]$ with rank m. H is in Hermite normal form if it is upper triangular, has monic diagonal entries, and in each column entries proceeding the diagonal entry are of lower degree.

(I) To every n by m matrix A over $F[x]$ with rank m there exists a unique n by m matrix H in Hermite normal form such that $UA = H$ for some unimodular matrix U. We call H the Hermite normal form of A. If $n = m$ (i.e. A is square nonsingular) then the unimodular matrix U is unique.

The main topic of chapter 3 was a class of algorithms for computing the HNF that perform all computations modulo the determinant of the input matrix (or determinant of the lattice thereof). These modular methods control the problem of coefficient growth when the normal form is computed over the domain of integers. The size of an integer (in the Euclidean sense) is its magnitude; the number of bits required to represent an integer is proportional to the logarithm of its magnitude. The size of a polynomial in $\mathbf{F}[x]$ (in the Euclidean sense) is its degree; the number of bits required to represent a polynomial will depend not only on the degree but also on the size (in bits) required to represent the individual coefficients from $\mathbf{F}$. The idea of working modulo the determinant of the input matrix can be applied also to HermiteForm over $\mathbf{F}[x]$ and serves to bound the degrees of intermediate polynomial entries. In particular, the results of §3.2 depend only on properties of lattices over PIDs and not on special properties of integers. As such, algorithm `ModHermite` of §3.2 provides a solution for HermiteForm over $\mathbf{F}[x]$. Thus, when the coefficient field $\mathbf{F}$ is a finite field such as GF(p), *a priori* bounds exist for the size in bits required to represent the coefficients from $\mathbf{F}$ and the problem of expression swell has been solved. However, when $\mathbf{F}$ is a field of characteristic 0, excessive coefficient growth is still a problem; working mod the determinant only controls the growth of the degrees of the entries, not the growth of the coefficients from $\mathbf{F}$.

The remainder of this chapter focuses on the more difficult case when $\mathbf{F} = \mathbf{Q}$; specifically, we tackle the problem HermiteForm over $\mathbf{Q}[x]$. Section §4.1 points out the problem of intermediate expression swell. Section §4.2 mentions some previous results that have motivated our work in §4.3 where we present a sequential deterministic solution for HermiteForm over $\mathbf{Q}[x]$. Section §4.4 concludes with a criticism of the algorithm given in §4.3 and summarizes the main results of the chapter.

4.1 Intermediate Expression Swell over $\mathbf{Q}[x]$

A key operation in the algorithm `ModHermite` of §3.2 is to solve the extended Euclidean problem: given elements $f_1, f_2 \in \mathbf{R}$, find $s, t \in \mathbf{R}$ such that

$$s f_1 + t f_2 = g$$

where g is the gcd of f_1 and f_2. When $\mathbf{R} = \mathbb{Z}$ we have good bounds on the size in bits of s and t; in particular, we can choose $|s| < |f_2|$ and $|t| < |f_1|$. When $\mathbf{R} = \mathbf{Q}[x]$ we can choose $deg\,(s) < deg\,(f_2)$ and $deg\,(t) < deg\,(f_1)$. For example, for the polynomials

$$f_1 = -85\,x^5 - 55\,x^4 - 37\,x^3 - 35\,x^2 + 97\,x + 50$$

$$f_2 = 79\,x^5 + 56\,x^4 + 49\,x^3 + 63\,x^2 + 57\,x - 59$$

we obtain

$$\begin{aligned} s &= \frac{325290923548154096}{36343806350461423445}\,x^4 + \frac{260765584444025981}{36343806350461423445}\,x^3 \\ &\quad + \frac{44784265674001653}{7268761270092284689}\,x^2 + \frac{53160964862220324}{7268761270092284689}\,x \\ &\quad + \frac{209029679900820952}{36343806350461423445} \\ t &= \frac{69999312662261008}{7268761270092284689}\,x^4 + \frac{51788020069512915}{7268761270092284689}\,x^3 \\ &\quad + \frac{174186164625775568}{36343806350461423445}\,x^2 + \frac{144940739882558651}{36343806350461423445}\,x \\ &\quad - \frac{87770584255662291}{7268761270092284689} \end{aligned}$$

such that

$$s f_1 + t f_2 = 1$$

For an input matrix A that has entries in the first column comparable in size to f_1 and f_2, the above example shows the amount coefficient growth that can be expected in the work matrix of algorithm `ClassicHermite` after the correct form of column 1 has been found. In particular, the work matrix would be premultiplied by a unimodular matrix that contained the entries s and t. A straightforward analysis of coefficient growth during the course of algorithm `ClassicHermite` when applied to an input matrix over $\mathbf{Q}[x]$ leads to an exponential upper bound on the size in bits of intermediate coefficients (cf. Kannan [25]).

While the potential for intermediate expression swell is great, another fact that must not be overlooked is that the end result of a HNF computation can itself typically be very large. For a coefficient $c \in \mathbf{F}$, the term *length* of c is used to indicate the number of bits required to represent c in binary. Note that if c is an integer, then the length of c will be proportional to the number of digits in the usual base 10 representation of c. Given an input matrix $A \in \mathbf{Q}[x]^{n\times n}$ with degrees of entries bounded by d and coefficients in $\mathbf{Q}$ bounded in length by l, a fundamental question is: what is a bound for the length of the rational coefficients appearing in H, the HNF of A, and in U, the unimodular matrix such that $UA = H$? Of course, we could answer the question by actually finding the HNF H, but what is desired here is an *a priori* bound in terms of the input matrix parameters n, d and l.

Example 1 *Let*

$$A = \begin{bmatrix} -85\,x^2 - 55\,x - 37 & -35\,x^2 + 97\,x + 50 & 79\,x^2 + 56\,x + 49 & 63\,x^2 + 57\,x - 59 \\ 45\,x^2 - 8\,x - 93 & 92\,x^2 + 43\,x - 62 & 77\,x^2 + 66\,x + 54 & -5\,x^2 + 99\,x - 61 \\ -50\,x^2 - 12\,x - 18 & 31\,x^2 - 26\,x - 62 & x^2 - 47\,x - 91 & -47\,x^2 - 61\,x + 41 \\ -58\,x^2 - 90\,x + 53 & -x^2 + 94\,x + 83 & -86\,x^2 + 23\,x - 84 & 19\,x^2 - 50\,x + 88 \end{bmatrix}.$$

Then,

$$U = \begin{bmatrix} [5,49] & [5,49] & [5,49] & [5,49] \\ [5,49] & [5,49] & [5,49] & [5,49] \\ [5,49] & [5,49] & [5,49] & [5,49] \\ [6,8] & [6,8] & [6,8] & [6,8] \end{bmatrix}$$

and

$$H = \begin{bmatrix} [0,1] & 0 & 0 & [7,50] \\ 0 & [0,1] & 0 & [7,50] \\ 0 & 0 & [0,1] & [7,50] \\ 0 & 0 & 0 & [8,9] \end{bmatrix}$$

where $[a,b]$ indicates a polynomial of degree a over $\mathbf{Q}[x]$ with numerators and denominators of coefficients over $\mathbf{Q}$ bounded in length by b. (Note: in this example, the length of an integer is the number of digits in the base 10 representation.) For *example,*

$$\begin{aligned}
U_{1,1} \quad = \quad & -\frac{40764229252165015711671466551074381443250929 1672}{44193224341086416228219762943140807637368610 84663} \\
& - \frac{116759998011328345350359970967423103549419058498}{1473107478036213874273992098104693587912287028221}\, x \\
& - \frac{2253356285821910236920575227470640560123494819 15}{4419322434108641622821976294314080763736861084663}\, x^2 \\
& + \frac{9140025612182889634824688476231059864955379 5}{35929450683810094494487612148894965558836268981}\, x^3 \\
& + \frac{2311509346238766335435316907060666853657392527 0}{1473107478036213874273992098104693587912287028221}\, x^4 \\
& - \frac{1013533038752151244180929023098803050011870270 85}{4419322434108641622821976294314080763736861084663}\, x^5
\end{aligned}$$

■

To summarize results it will be useful to define some notation. For a matrix $A \in \mathbf{F}[x]^{n \times m}$ with entries degree at most $d-1$ polynomials having coefficients from $\mathbf{F}$ representable in l bits, we will use the parameter s as a measure of the *size* of A, where s is defined as $s = n + m + d + l$. We write $\text{Size}(A)$ to indicate the number of bits required to represent A in binary. We may now write, for example, $\text{Size}(A) = O(nmdl)$. For brevity, it will be convenient to use the parameter s. For example: $O(n^2md + nm^2d + nmd^2) = O(s^3)$.

4.2 Previous Methods for HermiteForm over $\mathbf{Q}[x]$

In discussing previous results or presenting new ones, our task will be greatly simplified if we restrict our attention to square nonsingular matrices over $\mathbb{Z}[x]$. This corresponds to a preconditioning of the input and does not effect the generality of the results. To see this, let $A \in \mathbf{Q}[x]^{n \times m}$ with rank m be a general input matrix. Let $D \in \mathbb{Z}^{n \times n}$ be a diagonal matrix with i-th diagonal entry equal to the least common multiple of the denominators of all rational coefficients of all polynomial entries in row i of A. Then the matrix $A^* = DA$ is over $\mathbb{Z}[x]$ and has the same HNF as A. Now apply the technique given in §3 (pp. 19) to construct a square nonsingular matrix A^*_s from A^* such that the first m columns of the HNF of A^*_s comprise the HNF of A.

In what follows — and for the remainder of this chapter — let A be an $n \times n$ nonsingular input matrix over $\mathbb{Z}[x]$ with degrees of entries bounded by d, let H be the HNF of A, and let U the unique unimodular matrix such that $UA = H$. We write $||A||$ to denote the largest magnitude of all integer coefficients of A.

4.2.1 Kannan's Algorithm

The first proof that HermiteForm over $\mathbf{Q}[x]$ is in $\mathcal{P}$ was based on a variation of the classical algorithm by Kannan in [25]. Kannan gives an algorithm for HermiteForm over $\mathbf{Q}[x]$ that works by finding the HNF of the i-th principal minor of A for $i = 1, 2, \ldots, n$. (The classical algorithm finds the correct form of columns $i = 1, 2, \ldots, n$.) Kannan bounds the degrees of intermediate polynomials occurring when his algorithm operates on A by $2n^3d$ and the magnitudes of the numerators and denominators of rational coefficients of these polynomials by $(4nd||A||)^{400n^{11}d^4}$. These bounds lead to a polynomial bound on the length of intermediate rational coefficients based on the size in bits of the input matrix but they are astronomical from a practical point of view. Kannan mentions that these astronomical bounds are due in part to liberties taken during the analysis of his algorithm since he was primarily after a theoretical result (i.e. showing inclusion in $\mathcal{P}$).

4.2.2 The KKS Linear System Solution

The idea of converting HermiteForm over $\mathbf{F}[x]$ to that of solving linear systems over $\mathbf{F}$ appears to have first been used by Kaltofen, Krishnamoorthy and Saunders in [23] where they prove HermiteForm is in the parallel complexity class $\mathbf{NC}^2$. Their approach involves $O(n^2d)$ linear systems each of size $O(n^3d) \times O(n^3d)$ with magnitudes of entries bounded by $||A||$. The key to their approach is the following lemma. (Note: For a matrix $A \in \mathbf{F}[x]^{n\times n}$, $a_{i,j,k}$ indicates the coefficient of x^k of entry $a_{i,j}$.)

Lemma 3 *[23, Lemma 2.1] Given the n by n nonsingular matrix A over $\mathbf{F}[x]$ with entry degrees less than d, and the vector $(d_1, \cdots, d_n)$ of nonnegative integers, consider the system $TA = G$, where G is upper triangular, and more specifically,*

- *$t_{i,j}$ are polynomials of degree less than $nd + \max_{1\le i\le n} d_i$ whose coefficients are unknowns;*
- *$g_{i,j}$ are monic of degree d_i with lower order coefficients unknowns, and for $i < j$, $g_{i,j}$ are polynomials of degree less than d_j with unknowns as coefficients.*

This is a system of linear equations over $\mathbf{F}$ *in the unknown* $t_{i,j,k}$ *and* $g_{i,j,k}$ *for which the following statements hold.*

(1) The system has at least one solution, if and only if each d_i *is no less than the degree of the* i*-th diagonal entry of a Hermite normal form of* A.

(2) If each d_i *is exactly the degree of the* i*-th diagonal entry of a Hermite normal form of* A*, then the system has a unique solution, hence* G *is the unique Hermite normal form of* A *and* T *is unimodular.*

Their method can be described briefly as follows. A bound for deg $\det(A)$ is given by nd. In particular, the degrees of the diagonal entries in the HNF of A will be bounded by nd. If the $d_1, d_2, \ldots, d_n$ of Lemma 3 are each bounded by nd then the linear system will consist of $O(n^3 d)$ equations and $O(n^3 d)$ unknowns. Set $d_i = nd$ for $i = 1, 2, \ldots, k-1, k+1, \ldots n$. Then, for each value of d_k between 0 and nd, the linear system of Lemma 3 is consistent if and only if the degree of the k-th diagonal entry of the HNF of A is not greater than d_k. Hence, the correct degree of the k-th diagonal entry can be found by solving at most $O(nd)$ linear systems each of size $O(n^3 d) \times O(n^3 d)$. The degrees of all diagonal entries can be found by solving $O(n \cdot nd)$ linear systems. Finally, solving the linear system of Lemma 3 for the correct diagonal degrees $(d_1, \ldots, d_n)$ yields the matrices U and H.

Example 2 *Let*

$$A = \begin{bmatrix} x-1 & 3x+2 \\ x-1 & 2x+3 \end{bmatrix}$$

Say, for the purposes of this example, that the correct degrees of the diagonal entries in the HNF of A *have already been found and are* $(d_1, d_2) = (1, 1)$*. Then* $n = 2$*,* $d = 1$*, and Lemma 3 bounds the degrees of entries in* T *by* $nd + \max_{1 \le i \le n} = 3$*. The system we want to solve is*

$$\overset{T}{\begin{bmatrix} t_{113}\,x^3 + t_{112}\,x^2 + t_{111}\,x + t_{110} & t_{123}\,x^3 + t_{122}\,x^2 + t_{121}\,x + t_{110} \\ t_{213}\,x^3 + t_{212}\,x^2 + t_{211}\,x + t_{210} & t_{223}\,x^3 + t_{222}\,x^2 + t_{221}\,x + t_{210} \end{bmatrix}} \overset{A}{\begin{bmatrix} x-1 & 3x+2 \\ x-1 & 2x+3 \end{bmatrix}} = \overset{G}{\begin{bmatrix} x + g_{110} & g_{120} \\ 0 & x + g_{220} \end{bmatrix}}$$

for the coefficients $\{t_{i,j,k}, g_{i,j,k}\}$*. We can write the above as a linear system of constraints in the coefficients* $\{t_{i,j,k}, a_{i,j,k}, g_{i,j,k}\}$ *by expressing the polynomial multiplications as convolution products over* $\mathbf{Q}$ *and equating coefficients on the left* *and right hand side.*

$$\left[\begin{array}{cccc|cccc} t_{113} & t_{112} & t_{111} & t_{110} & t_{123} & t_{122} & t_{121} & t_{120} \\ \hline t_{213} & t_{212} & t_{211} & t_{210} & t_{223} & t_{222} & t_{221} & t_{220} \end{array}\right] \begin{bmatrix} 1 & -1 & 0 & 0 & 0 & 3 & 2 & 0 & 0 & 0 \\ 0 & 1 & -1 & 0 & 0 & 0 & 3 & 2 & 0 & 0 \\ 0 & 0 & 1 & -1 & 0 & 0 & 0 & 3 & 2 & 0 \\ 0 & 0 & 0 & 1 & -1 & 0 & 0 & 0 & 3 & 2 \\ 1 & -1 & 0 & 0 & 0 & 2 & 3 & 0 & 0 & 0 \\ 0 & 1 & -1 & 0 & 0 & 2 & 3 & 0 & 0 & 0 \\ 0 & 0 & 1 & -1 & 0 & 0 & 0 & 2 & 3 & 0 \\ 0 & 0 & 0 & 1 & -1 & 0 & 0 & 0 & 2 & 3 \end{bmatrix} =$$

$$\begin{bmatrix} 0 & 0 & 0 & 1 & g_{110} & 0 & 0 & 0 & 0 & g_{120} \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & g_{220} \end{bmatrix}$$

Converting the above system to standard form (i.e. $A\vec{x} = \vec{b}$ where A and $\vec{b}$ are known, $\vec{x}$ is unknown) yields:

$$\begin{bmatrix}
1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
-1 & 1 & 0 & 0 & -1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & -1 & 1 & 0 & 0 & -1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & -1 & 1 & 0 & 0 & -1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & -1 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 \\
3 & 0 & 0 & 0 & 2 & 2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
2 & 3 & 0 & 0 & 3 & 3 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 2 & 3 & 0 & 0 & 0 & 2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 2 & 3 & 0 & 0 & 3 & 2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 2 & 0 & 0 & 0 & 3 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 1 & 0 & 0 & -1 & 1 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 1 & 0 & 0 & -1 & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 1 & 0 & 0 & -1 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & -1 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 3 & 0 & 0 & 0 & 2 & 2 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 2 & 3 & 0 & 0 & 3 & 3 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 2 & 3 & 0 & 0 & 0 & 2 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 2 & 3 & 0 & 0 & 3 & 2 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 2 & 0 & 0 & 0 & 3 & 0 & 0 & -1
\end{bmatrix}
\begin{bmatrix} t_{113} \\ t_{112} \\ t_{111} \\ t_{110} \\ t_{123} \\ t_{122} \\ t_{121} \\ t_{120} \\ t_{213} \\ t_{212} \\ t_{211} \\ t_{210} \\ t_{223} \\ t_{222} \\ t_{221} \\ t_{220} \\ g_{110} \\ g_{120} \\ g_{220} \end{bmatrix}
=
\begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 1 \\ 0 \end{bmatrix}$$

Solving the above system yields the unique solution

$$\begin{bmatrix} t_{113} \\ t_{112} \\ t_{111} \\ t_{110} \\ t_{123} \\ t_{122} \\ t_{121} \\ t_{120} \\ t_{213} \\ t_{212} \\ t_{211} \\ t_{210} \\ t_{223} \\ t_{222} \\ t_{221} \\ t_{220} \\ g_{110} \\ g_{120} \\ g_{220} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ -2 \\ 0 \\ 0 \\ 0 \\ 3 \\ 0 \\ 0 \\ 0 \\ 1 \\ 0 \\ 0 \\ 0 \\ -1 \\ -1 \\ 5 \\ -1 \end{bmatrix}$$

which corresponds to

$$\overset{U}{\begin{bmatrix} -2 & 3 \\ 1 & -1 \end{bmatrix}} \overset{A}{\begin{bmatrix} x-1 & 3x+2 \\ x-1 & 2x+3 \end{bmatrix}} = \overset{H}{\begin{bmatrix} x-1 & 5 \\ 0 & x-1 \end{bmatrix}}.$$

■

The goal of the authors of [23] was to establish a parallel complexity result. From a theoretical perspective, the number of systems and size of each linear system does not effect their main result: namely, that HERMITEFORM over $\mathbf{Q}[x]$ is in $\mathbf{NC}^2$. From a sequential point of view, the number and size of each system makes their approach impractical. In the next section, we reexamine their approach but with the view of obtaining a more efficient sequential solution.

4.3 A Linear Systems Method

We present here a sequential deterministic solution for HERMITEFORM over $\mathbf{F}[x]$ that works by reducing the problem to that of solving a *single* linear system over $\mathbf{F}$ of size $O(n^2d) \times O(n^2d)$. The method is novel in that the linear system constructed does not involve unknown coefficients of H, the HNF of A, but rather only the coefficients of the unimodular multiplier matrix U. H can then found by effecting the multiplication $H \leftarrow UA$.

4.3.1 Preliminaries

We require some notation: for a matrix $A \in \mathbf{F}[x]^{n\times n}$, let $A_{i,j}$ or $a_{i,j}$ denote the entry in the i-th row, j-th column of A, $a_{i,j,k}$ the coefficient of x^k of $a_{i,j}$, and $\text{row}(A,i)$ the i-th row vector of A. We write *deg* A to mean the maximum of the degrees of all entries in the matrix A and *deg* $a_{i,j}$ for the degree of polynomial $a_{i,j}$. The degree of the zero polynomial is $-\infty$.

Let $\mathcal{L}(A)$ denote the lattice of all $\mathbf{F}[x]$-linear combinations of the rows of A. Then $\mathcal{L}(A)$ will have rank n ($=\text{rank}(A)$). The n rows of H — the HNF of A — provide the unique basis for $\mathcal{L}(A)$ which satisfy the definition of HNF. One of the advantages of the triangular HNF basis for $\mathcal{L}(A)$ is that we can determine immediately that certain vectors cannot belong to $\mathcal{L}(A)$. This is made precise by the following lemma.

Lemma 4 *Let $A \in \mathbf{F}[x]^{n\times n}$ be nonsingular with HNF H. Let d_i be the degree of the i-th diagonal entry of H. Let $\vec{v} = (0,\ldots,0,v_k,v_{k+1},\ldots,v_n)$ be a row vector over $\mathbf{F}[x]$ with $v_i = 0$ for $1 <= i < k$ and* $\deg(v_k) < d_i$. *If $\vec{v}$ is in $\mathcal{L}(A)$ then $v_k = 0$. Conversely, if $v_k \neq 0$ then $\vec{v} \notin \mathcal{L}(A)$.*

Proof: $\vec{v} \in \mathcal{L}(A) = \mathcal{L}(H) \Rightarrow \vec{v} = p_1\text{row}(H,1) + \cdots + p_n\text{row}(H,n)$ for some $p_i \in \mathbf{F}[x], 1 \le i \le n$. Since $v_i = 0$ for $1 \le i < k$ implies $p_i = 0$ for $1 \le i < k$, we must have $\vec{v} = p_k\text{row}(H,k) + \cdots + p_n\text{row}(H,n)$. The only vector in $\{\text{row}(H,k),\ldots,\text{row}(H,n)\}$ with k-th entry non zero is $\text{row}(H,k)$, so $v_k \neq 0$ iff $p_k \neq 0$. But *deg* $(v_k) < d_i \Rightarrow p_k = 0$ whence $v_k = 0$. ∎

We continue with a stronger version of the Lemma 3 given in §4.2.2. Lemma 3 constructed a linear system from the polynomial identity $TA = G$ having as unknowns the coefficients of the polynomial entries in T and G. The following lemma makes the key observation that results analogous to those of Lemma 3 still hold when we express the linear system independently of the unknown coefficients of G. The benefit of this approach is that the dimension of linear system is reduced by a factor of n from $O(n^3d)$ to $O(n^2d)$ (we show this carefully later).

Lemma 5 *Let A be a given n by n nonsingular matrix over $\mathbf{F}[x]$ with degrees of entries bounded by d. Let $(d_1,\ldots d_n)$ be a given vector of nonnegative integers. Let T be an n by n matrix having as entries univariate polynomials in x with unknowns as coefficients and degrees bounded by $D_T = (n-1)d - \deg\ \det(A) + \max_{1\le i\le n} d_i$. Consider the linear system of equations over $\mathbf{F}$ in the unknowns $t_{i,j,k}$ with constraints*

$$(TA)_{i,i,d_i} \quad = \quad 1 \text{ for } 1 \le i \le n$$

$$
\begin{aligned}
(TA)_{i,i,k} &= 0 \text{ for } k > d_i \\
(TA)_{i,j,k} &= 0 \text{ for } i \neq j, k \geq d_i
\end{aligned}
$$

Let $(h_1, \ldots, h_n)$ be the degrees of the diagonal entries of the Hermite normal form of A. The following statements about the above system hold:

(1) The system has at least one solution if $d_i \geq h_i, 1 \leq i \leq n$.

(2) If there exists a nonnegative integer $b \leq n$ such that $d_i = h_i$ for $1 \leq i < b$ and $d_b < h_i$ then the system has no solution.

(3) If $d_i = h_i$ for $1 \leq i \leq n$, then the system has a unique solution T with TA equal to the Hermite normal form of A.

Proof: Let H be the HNF form of A and let U be the unique unimodular matrix such that $UA = H$. To show (1), let $D = \mathrm{diag}(x^{d_1-h_1}, \ldots, x^{d_n-h_n})$ and consider the equality $DUA = DH$. Let H^* be the HNF of DH and U^* a unimodular matrix such that $U^*DUA = H^*$. We claim that we can take as our solution $T = U^*DU$. Firstly, the particular choice of D together with the definition of H^* ensure that the constraints for the linear system are satisfied. It remains to show that entries in T have degrees bounded by $(n-1)d - deg\ \det(A) + \max_{1 \leq i \leq n} d_i$. To see this note that $TA = H^* \Rightarrow \det(A)T = H^*A^{\mathrm{adj}} \Rightarrow deg\ T \leq deg\ H^* + deg\ A^{\mathrm{adj}} - deg\ \det(A) \leq \max_{1 \leq i \leq n} d_i + (n-1)d - deg\ \det(A)$.

To prove (2), assume by contradiction that there exists a nonnegative integer $b \leq n$ and a solution for T such that $deg\ ((TA)_{i,i}) = h_i$ for $1 \leq i < b$ and $deg\ ((TA)_{b,b}) < h_i$. In particular, we are assuming that $\mathrm{row}(TA, b) = ((TA)_{b,1}, \ldots, (TA)_{b,n})$ is in $\mathcal{L}(A)$. Since $deg\ ((T_1A)_{b,1}) < h_1$ we can use Lemma 4 to assert that $(TA)_{b,1} = 0$. By induction assume $(TA)_{b,j} = 0$ for $i \leq j < k$. Then by Lemma 4 we must have $(TA)_{b,k} = 0$ since $deg\ ((TA)_{b,k}) < h_k$. In particular we have $(TA)_{b,b} = 0$ which is a contradiction since the constraints specify that $(TA)_{b,b}$ be monic.

If the conditions of (3) hold then by (1) there exists at least one solution for T. We can use an induction proof similar to that used in the proof of (2) to show that elements below the diagonal in TA are zero (i.e. that $(TA)_{i,j} = 0$ for $i > j$). By the uniqueness of HNF we must have $TA = H$. ■

Remark: Note that the constraints of the linear system specify precisely that the matrix TA should be in HNF with degrees of diagonal entries given by $(d_1, d_2, \ldots, d_n)$.

Example 3 *Let*

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

Say, for the purposes of this example, that we know the correct degrees of the diagonal entries in the HNF of A to be $(d_1, d_2, d_3) = (1, 0, 1)$. Then $n = 3, d = 1, \deg\ \det(A) = 2$, and Lemma 5 bounds the degrees of entries in T by $D_T = 1$. The system we want to solve is

$$\overset{T}{\begin{bmatrix} t_{111}x + t_{110} & t_{120}x + t_{121} & t_{130}x + t_{131} \\ t_{211}x + t_{210} & t_{220}x + t_{221} & t_{230}x + t_{231} \\ t_{311}x + t_{310} & t_{320}x + t_{321} & t_{330}x + t_{331} \end{bmatrix}} \overset{A}{\begin{bmatrix} x-1 & 4x+2 & 0 \\ x-1 & 5 & 2x \\ x-1 & 2x+3 & x+2 \end{bmatrix}} = \overset{G}{\begin{bmatrix} x + g_{110} & 0 & g_{130} \\ 0 & 1 & g_{230} \\ 0 & 0 & x + g_{330} \end{bmatrix}}$$

for the coefficients $\{t_{i,j,k}\}$. Writing the above as a linear system of constraints in the coefficients $\{t_{i,j,k}, a_{i,j,k}, g_{i,j,k}\}$ yields

$$\overset{T_{\text{lin}}}{\left[\begin{array}{cc|cc|cc} t_{111} & t_{110} & t_{120} & t_{121} & t_{130} & t_{131} \\ \hline t_{211} & t_{210} & t_{220} & t_{221} & t_{230} & t_{231} \\ \hline t_{311} & t_{310} & t_{320} & t_{321} & t_{330} & t_{331} \end{array}\right]} \overset{A_{\text{lin}}}{\left[\begin{array}{ccc|ccc|ccc} 1 & -1 & 0 & 4 & 2 & 0 & 0 & 0 & 0 \\ 0 & 1 & -1 & 0 & 4 & 2 & 0 & 0 & 0 \\ \hline 1 & -1 & 0 & 0 & 5 & 0 & 2 & 0 & 0 \\ 0 & 1 & -1 & 0 & 0 & 5 & 0 & 2 & 0 \\ \hline 1 & -1 & 0 & 2 & 3 & 0 & 1 & 2 & 0 \\ 0 & 1 & -1 & 0 & 2 & 3 & 0 & 1 & 2 \end{array}\right]} = \overset{G_{\text{lin}}}{\left[\begin{array}{ccc|ccc|ccc} 0 & 1 & g_{110} & 0 & 0 & 0 & 0 & 0 & g_{130} \\ \hline 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & g_{230} \\ \hline 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & g_{330} \end{array}\right]},$$

By Lemma 5, we can neglect those constraints that involve unknown coefficients $\{g_{i,j,k}\}$ to obtain

$$\overset{T_{\text{lin}}}{\left[\begin{array}{cc|cc|cc} t_{111} & t_{110} & t_{120} & t_{121} & t_{130} & t_{131} \\ \hline t_{211} & t_{210} & t_{220} & t_{221} & t_{230} & t_{231} \\ \hline t_{311} & t_{310} & t_{320} & t_{321} & t_{330} & t_{331} \end{array}\right]} \overset{A^*_{\text{lin}}}{\left[\begin{array}{cc|ccc|cc} 1 & -1 & 4 & 2 & 0 & 0 & 0 \\ 0 & 1 & 0 & 4 & 2 & 0 & 0 \\ \hline 1 & -1 & 0 & 5 & 0 & 2 & 0 \\ 0 & 1 & 0 & 0 & 5 & 0 & 2 \\ \hline 1 & -1 & 2 & 3 & 0 & 1 & 2 \\ 0 & 1 & 0 & 2 & 3 & 0 & 1 \end{array}\right]} = \overset{G^*_{\text{lin}}}{\left[\begin{array}{cc|ccc|cc} 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ \hline 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ \hline 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{array}\right]},$$

a linear system with a unique solution for T_{lin}. Solving the system yields

$$T_{\text{lin}} = \left[\begin{array}{cc|cc|cc} \frac{4}{7} & -\frac{39}{7} & \frac{4}{7} & -\frac{30}{7} & -\frac{8}{7} & \frac{76}{7} \\ \hline 0 & 1 & 0 & 1 & 0 & -2 \\ \hline -\frac{2}{7} & \frac{2}{7} & -\frac{2}{7} & \frac{1}{7} & \frac{4}{7} & -\frac{3}{7} \end{array}\right].$$

Finally, note that

$$\overset{T}{\begin{bmatrix} \frac{4}{7}x - \frac{39}{7} & \frac{4}{7}x - \frac{30}{7} & -\frac{8}{7}x + \frac{76}{7} \\ 1 & 1 & -2 \\ -\frac{2}{7}x + \frac{2}{7} & -\frac{2}{7}x + \frac{1}{7} & \frac{4}{7}x - \frac{3}{7} \end{bmatrix}} \overset{A}{\begin{bmatrix} x-1 & 4x+2 & 0 \\ x-1 & 5 & 2x \\ x-1 & 2x+3 & x+2 \end{bmatrix}} = \overset{G}{\begin{bmatrix} x-1 & 0 & \frac{152}{7} \\ 0 & 1 & -4 \\ 0 & 0 & x - \frac{6}{7} \end{bmatrix}},$$

the correct Hermite normal form of A.

We now determine bounds on the dimension of the linear system of Lemma 5. First, consider the general matrix identity $TA = G$ where A is known and T and G are unknown. The following elementary result should be obvious after perusing the worked example above.

Lemma 6 *Let $A \in \mathbf{F}[x]^{n\times n}$ be a given matrix with degrees of entries bounded by d. Let $T \in \mathbf{F}[x]^{n\times n}$ have entries bounded in degree by D_T with unknowns as coefficients. Let $G \in \mathbf{F}[x]^{n\times n}$ have degrees of entries bounded by $\deg TA = D_T + d$. The matrix identity $TA = G$ can be written as a linear system in the matrix coefficients $\{t_{i,j,k}, a_{i,j,k}, g_{i,j,k}\}$ as $T_{\mathrm{lin}}A_{\mathrm{lin}} = G_{\mathrm{lin}}$ over $\mathbf{F}$ where:*

$$\begin{aligned} \mathrm{rowdim}(T_{\mathrm{lin}}) &= n; \\ \mathrm{coldim}(T_{\mathrm{lin}}) &= n(D_T + 1); \\ \mathrm{coldim}(A_{\mathrm{lin}}) &= n(D_T + d + 1) \end{aligned}$$

Now consider the linear system of Lemma 5. Each entry of matrix T is a polynomial with $D_T + 1$ unknown coefficients contributing to a total of $n(D_T + 1)$ unknown coefficients in each of the n rows of T. In particular, all the unknowns of the linear system of Lemma 5 can be written as an $n \times n(D_T + 1)$ matrix T_{lin} (as in Lemma 6). All the constraints of the system deal with specifying the values of coefficients of entries in the polynomial matrix TA, which has degrees of entries bounded by $D_T + d$. Consider the constraints that are limited to a single column of TA, say column j. Note that there is a constraint precisely for all coefficients $(TA)_{*,j,k}$ where $d_j \le k \le D_T + d + 1$. In other words, precisely d_j constraints, namely the constraints for coefficients of x^k with $0 \le k < d_j$, are not included. Over all columns, $\sum_{1\le j\le n} d_j$ constraints will be ignored. This leads to the following result.

Corollary 2 *The linear system of Lemma 5 can be expressed as a matrix equation $T_{\mathrm{lin}}A^*_{\mathrm{lin}} = G^*_{\mathrm{lin}}$ over $\mathbf{F}$ where:*

$$\begin{aligned} \mathrm{rowdim}(T_{\mathrm{lin}}) &= n; \\ \mathrm{coldim}(T_{\mathrm{lin}}) &= n(D_T + 1); \\ \mathrm{coldim}(A^*_{\mathrm{lin}}) &= n(D_T + d + 1) - \sum_{1\le i\le j} d_i; \end{aligned}$$

where $T, A, G, D_T, (d_1, \ldots, d_n)$ are as in Lemma 5 and 6.

Remarks:
(1) The n rows of G^*_{lin} will have all zero entries except for a single entry 1 in each row. This corresponds to the constraint that the diagonal entries of matrix TA be monic.
(2) The matrices A^*_{lin} and G^*_{lin} are submatrices, respectively, of A_{lin} and G_{lin} of Lemma 6.
(3) Note that the dimensions of the system in Example 3 are consistent with that given by Corollary 2.

4.3.2 A Bound for the Size of U

Now that we have size bounds for the dimension of the linear system, we can obtain bounds for the lengths of rational entries in the solution matrix T_{lin}. In particular, we are interested in the case $\mathbf{F} = \mathbf{Q}$ and when T_{lin} corresponds the the unique unimodular matrix U such that $UA = H$. The following result gives an *a priori* bound on the size in bits of the solution U and H.

Theorem 6 *Let A be a given n by n nonsingular matrix over $\mathbb{Z}[x]$ with degrees of entries bounded by d (> 0) and maximum magnitude of integer coefficients $||A||$. Let U be the unique unimodular matrix $U \in \mathbf{Q}[x]^{n\times n}$ such that $UA = H$, the HNF of A. Then, the degrees of entries in U are bounded by $(n-1)d$. Furthermore, there exists a positive number $\beta_U \leq (n\sqrt{d}||A||)^{n^2 d}$ such that*

1 *The numerators and denominators of rational coefficients of the polynomial entries of U are bounded in magnitude by β_U.*

2 *There exits a nonzero integer $\alpha \leq \beta_U$ such α is the least common multiple of all denominators of rational coefficients in U; that is, $\alpha U \in \mathbb{Z}[x]^{n\times n}$.*

3 *The numerators and denominators of rational coefficients of the polynomial entries of H are bounded in magnitude by $\beta = n(d+1)||A||\beta_U = O(\beta_U)$.*

Proof: In Lemma 5, choose $(d_1, \ldots, d_n)$ to be the correct degrees of the diagonal entries in H, the HNF of A. Then we have $deg\ H \leq deg\ \det(A) \leq nd$ and $deg\ U = deg\ T \leq D_T = (n-1)d - deg\ \det(A) + \max_{1\leq i\leq n} d_i \leq (n-1)d$. By Corollary 2, there exists a linear system with $T_{\text{lin}} A^*_{\text{lin}} = G^*_{\text{lin}}$ that admits a unique solution for T_{lin} (corresponding to the rational coefficients of polynomial entries of U). Since the solution is unique, the matrix A^*_{lin} has full row rank. In particular, there must exists a subset of columns of A^*_{lin}, say ϕ, such that the submatrix of

A^*_{lin} restricted to the columns ϕ, call it A^s_{lin}, is square nonsingular. (In fact, any subset ϕ with this property will do.) Let G^s_{lin} be the submatrix of G^*_{lin} restricted to the columns ϕ. Then, the restricted system $T_{\text{lin}}A^s_{\text{lin}} = G^s_{\text{lin}}$ also admits a unique solution for T_{lin}, namely $T_{\text{lin}} = G^s_{\text{lin}}(A^s_{\text{lin}})^{\text{adj}}1/\det(A^s_{\text{lin}})$. The dimension of A^s_{lin} is $n(D_T+1) = n((n-1)d+1) \leq n^2d$ (the last inequality is correct if we assume $d > 0$). The coefficients of A^s are comprised of the coefficients of the polynomial entries of A — these are integers bounded in magnitude by $||A||$. The usual hadamard bound gives $||\det(A^s_{\text{lin}})|| \leq (n\sqrt{d}||A||)^{n^2d}$. This shows that (2) holds. Entries in $(A^s_{\text{lin}})^{\text{adj}}$ are determinants of minors of A^s_{lin} — these will be bounded by $(n\sqrt{d}||A||)^{n^2d}$ as well. To see that (1) holds for U, note that matrix G^s_{lin} has precisely one nonzero entry (the integer 1) in each row, from which it follows that $T_{\text{lin}} = G^s_{\text{lin}}(A^s_{\text{lin}})^{\text{adj}}$ is a submatrix of $(A^s_{\text{lin}})^{\text{adj}}$. To see that (1) holds for H, note that each entry of H will be the dot product of a row vector in U and a column vector in A. Note that if f and g are polynomials over $\mathbb{Z}$, and *deg* $f \leq d$, then $||fg|| \leq (d+1)||f||||g||$. In particular, $||\alpha H|| = ||\alpha UA|| \leq n(d+1)||A||||\alpha U|| \leq \alpha n(d+1)||A||\beta_U = \alpha\beta$. ∎

Computational experience with dense matrices of dense polynomials indicates that the length of the coefficients in U tend to grow quadratically with respect to n and linearly with respect to d, and, although we have not shown it, we hypothesize that the bound given for β_U in Theorem 6 is in fact an asymptotic lower bound. That is, there exists a positive constant c such that for any positive n, d and M, there exists a nonsingular input matrix $A \in \mathbb{Z}[x]^{n\times n}$ with *deg* $(A) \leq d$, $||A|| \leq M$ and such that the corresponding matrix U contains a rational coefficient with either numerator or denominator having magnitude larger than $c(n\sqrt{d}||A||)^{n^2d}$.

In practice, matrices may be far from the worst case. In particular, Theorem 6 does not take into account that the matrix may be sparse with entries sparse polynomials and that *deg* (A) and $||A||$ may not be indicative of the size of typical entries in A. However, the proof of Theorem 6 immediately suggests an algorithm to quickly determine a better candidate for β_U. The number β_U was chosen to be an upper bound for the determinant of any nonsingular submatrix of the matrix A^*_{lin} of Corollary 2; Rather than appeal to an *a priori* bound as we did in the proof of Theorem 6, a better method is to construct such a bound directly from the linear system A^*_{lin}. In particular, the Hadamard bound for the determinant of a square integer matrix is given by the product of the Euclidean norms of all columns. This leads to the following. (Recall that the Euclidean norm of a vector is the square root of the sum of the squares of the elements.)

Fact 1 *Let A be an $n \times n$ matrix over $\mathbb{Z}$. Denote by $|\mathrm{col}(A,j)|$ the Euclidean norm of column j. For any square submatrix A^s of A of dimension n, we have*

$$|\det(A^s)| \leq \max_{1 \leq j_1 < j_2 < \cdots < j_n \leq n} \left\{ \prod_{1 \leq k \leq n} |\mathrm{col}(A, j_i)| \right\}$$

∎

If the degrees of the diagonal entries of H are known, then choose β_U to be the product of the n largest column norms of matrix A^*_{lin}; otherwise take the product of the n largest column norms of matrix A_{lin}. Note that the matrix A_{lin} or A^*_{lin} does not need to be written down explicitly — only the column norms need to be found.

Example 4 *For the input matrix A of Example 1, we have $n = 4$, $d = 2$, $||A|| = 94$, and the largest numerator or denominator in U or H can be written with 50 decimal digits. The candidate for β given by Theorem 6 has length $\lceil \log_{10} n(d+1)||A||(n\sqrt{d}||A||)^{n^2 d} \rceil = 91$ decimal digits. On the other hand, the candidate for β given by Fact 1 requires only* 66 *decimal digits.*

We end this section with a result that summarizes some size bounds for the matrices U and H corresponding to a given input matrix A. For comparison, we give also the size of A^{adj}.

Corollary 3 *Let A, H and U be the matrices of Theorem 6. Then* $\mathrm{Size}(A) = O(n^2 d \log ||A||)$ *and* $\mathrm{Size}(A^{\mathrm{adj}}) = O^{\sim}(n^4 d \log ||A||)$. *Furthermore,*

$$\mathrm{Size}(H) = O^{\sim}(n^4 d^2 \log ||A||)$$

and

$$\mathrm{Size}(U) = O^{\sim}(n^5 d^2 \log ||A||).$$

4.3.3 An Algorithm for HermiteForm over $\mathbf{F}[x]$

At the start of this section, we claimed that HermiteForm over $\mathbf{F}[x]$ reduces to solving a *single* linear system over $\mathbf{F}$ of size $O(n^2 d)$. One way could demonstrate this is by giving a procedure that finds the correct degrees of the diagonal entries in the HNF of H within the cost required to solve the linear system $T^*_{\mathrm{lin}} A^*_{\mathrm{lin}} = G^*_{\mathrm{lin}}$ of Corollary 2.

Finding the correct degrees of the diagonal entries in H is tantamount to determining the correct submatrix A^*_{lin} of the matrix A_{lin} of Lemma 6. In fact, we have developed an approach that builds the system A^*_{lin} *while* solving it. This technique involves considering, in turn, the j-th column constraint of matrix A_{lin} for $j = 1, 2, \ldots$, and neglecting certain column constraints because of linear dependencies that arise during the reduction. For various reasons, though, we have no interest in giving a rigorous exposition of this algorithm. First, we have already seen that HERMITEFORM over $\mathbf{F}[x]$ in in $\mathcal{P}$. Second, in chapter 6 we we give an algorithm for HERMITEFORM over $\mathbf{Q}[x]$ that, compared to the linear systems method, is dramatically superior in terms of space requirements and expected number of bit operations. (Consider alone that a worst case bound for the size in bits required to represent the triangularized system A^*_{lin} is $O^\sim(n^6 d^3 \log ||A||)$ bits.)

Instead, we give here an example of the exceedingly elegant linear systems method for HERMITEFORM presented by Labhalla, Lombardi and Marlin [27]. Note that the typical method to solve the system $T_{\text{lin}} A^*_{\text{lin}} = G^*_{\text{lin}}$ is to reduce to A^*_{lin} via column operations to column echelon form. In our case, we have devised a method that forms the correct system A^*_{lin} during a column echelon reduction of matrix A_{lin}. A better solution is given by Labhalla, Lombardi and Marlin who developed their linear systems method based on the theory of subresultants and Sylvester matrices. The following is a restatement of their result that uses the notation that we have defined previously.

Theorem 7 ([27]) *Let A_{lin} be the rational matrix of Lemma 6 with $D_T = (n-1)d$. The row vectors of the HNF of A are computed by a triangularization of the matrix A_{lin}, using row operations only, followed by the reduction of the off-diagonal entries by row operations.*

Example 5 *Let*

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

Say, for the purposes of this example, that we know that degrees of entries in U are bounded by $D_T = 1$. Then we have

$$\overset{T_{\text{lin}}}{\left[\begin{array}{cc|cc|cc} t_{111} & t_{110} & t_{120} & t_{121} & t_{130} & t_{131} \\ \hline t_{211} & t_{210} & t_{220} & t_{221} & t_{230} & t_{231} \\ \hline t_{311} & t_{310} & t_{320} & t_{321} & t_{330} & t_{331} \end{array}\right]} \overset{A_{\text{lin}}}{\left[\begin{array}{ccc|ccc|ccc} 1 & -1 & 0 & 4 & 2 & 0 & 0 & 0 & 0 \\ 0 & 1 & -1 & 0 & 4 & 2 & 0 & 0 & 0 \\ \hline 1 & -1 & 0 & 0 & 5 & 0 & 2 & 0 & 0 \\ 0 & 1 & -1 & 0 & 0 & 5 & 0 & 2 & 0 \\ \hline 1 & -1 & 0 & 2 & 3 & 0 & 1 & 2 & 0 \\ 0 & 1 & -1 & 0 & 2 & 3 & 0 & 1 & 2 \end{array}\right]} = \overset{G_{\text{lin}}}{\left[\begin{array}{ccc|ccc|ccc} 0 & 1 & g_{110} & 0 & 0 & 0 & 0 & 0 & g_{130} \\ \hline 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & g_{230} \\ \hline 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & g_{330} \end{array}\right]},$$

Reducing the matrix $A_{\rm lin}$ to Gauss-Jordan form using row operations yields

$$H_{\rm lin} = \left[\begin{array}{ccc|ccc|ccc} 1 & 0 & -1 & 0 & 0 & 0 & 2 & 0 & \frac{272}{7} \\ 0 & 1 & -1 & 0 & 0 & 0 & 0 & 0 & \frac{152}{7} \\ 0 & 0 & 0 & 1 & 0 & 0 & -1/2 & 0 & -\frac{18}{7} \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & -\frac{24}{7} \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & -4 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & -6/7 \end{array}\right]$$

The last nonzero row in each column block will correspond to the respective diagonal entry in the HNF of A. We obtain

$$H = \begin{bmatrix} x-1 & 0 & \frac{152}{7} \\ 0 & 1 & -4 \\ 0 & 0 & x-6/7 \end{bmatrix},$$

the correct HNF of A. ■

4.4 Conclusions

The problem of intermediate expression swell is ubiquitous in computer algebra. Much work has been devoted to coming up with computational techniques that abdicate this problem and allow basic ring operations to be computed faster. For polynomials over $\mathbb{Z}[x]$, these include: (1) modular homomorphisms and Chinese remaindering; (2) evaluation homomorphisms and interpolation; (3) efficient gcd computation; (4) fast polynomial multiplication. All these methods exploit the structure that is inherent to polynomial domains.

A serious drawback of the linear systems solution for HermiteForm given in §4.3.3 is that the structure inherent in the polynomial domain is lost. While converting the problem from the domain $\mathbf{F}[x]$ to $\mathbf{F}$ facilitated the derivation of simple size bounds for intermediate coefficients, the cost of solving the linear system is too expensive (both in terms of space and time) to allow a practical implementation.

Nonetheless, this chapter has made considerable progress towards a practical solution for HermiteForm over $\mathbf{Q}[x]$. Our most important result is the size bounds given by Theorem 6 for the coefficients appearing in the unimodular multiplier matrix U and the HNF H. This will be very useful in chapter 6 where a modular algorithm for HERMITEFORM over $\mathbf{Q}[x]$ is presented. Also, we can now make some general comments on the difficulty of HERMITEFORM over $\mathbf{Q}[x]$. In particular, a fundamental lower bound for the bit complexity (number of bit operations) required to solve a problem is given by the size in bits of the result. Corollary 3 gives a useful comparison in this regard between Size(H) and Size(A^{adj}).

Finding the adjoint of A is by no means a trivial computation, but good methods exists to solve this problem. In particular, a worst case complexity bound for computing A^{adj} is $O\tilde{~}(n^5 d \log ||A||)$ (disallowing fast matrix multiplication), which is at most $O\tilde{~}(n)$ larger than an asymptotic lower bound for Size(A^{adj}). Now consider the matrix H. Although the length of the coefficients from $\mathbf{Q}$ appearing in H will be $O\tilde{~}(nd)$ times as large as those appearing in A^{adj}, there are $O(n)$ times as many coefficients appearing in A^{adj}. In practice, the adjoint of a matrix is typically dense with dense polynomials of degree $O(nd)$ (a similar observation holds for U). Thus, for many cases, we can expect Size(H) $\approx d$Size(A^{adj}). Similarly, we can expect Size(U) $\approx n$Size(H).

Chapter 5

A Fast Algorithm for SmithForm over $\mathbf{Q}[x]$

This chapter considers the problem SmithForm over $\mathbf{F}[x]$, $\mathbf{F}$ a field. We give a fast sequential algorithm for computing the SNF of polynomial matrices over finite fields or the field of rationals. The algorithm we give is probabilistic in the Las Vegas sense — an incorrect result will never be returned but with small probability the algorithm may fail and require repetition. Previous probabilistic algorithms for SmithForm have been given by Kaltofen, Krishnamoorthy and Saunders [23, 24]. In particular, in [23] the authors give a parallel algorithm for SmithForm over $\mathbf{F}[x]$ that reduces the problem to matrix multiplication and determinant and gcd computations. Their algorithm leads to an efficient sequential solution for SmithForm but is probabilistic in the Monte Carlo sense — with small probability an incorrect solution may be returned. Our contribution is a method for quickly verifying that the result of the KKS Monte Carlo algorithm is correct. Consequently, we provide a fast Las Vegas SmithForm algorithm in §5.3. The qualification fast is justified on two fronts. First, we demonstrate a complexity for the algorithm in terms of expected number of bit operations that improves dramatically on any previously published results. Secondly, the algorithm is eminently practical, admitting a very simple implementation (in a computer algebra language), and outperforms any previously known method.

Without loss of generality, we assume that an input matrix $A \in \mathbf{Q}[x]^{n\times m}$ has integer coefficients. Complexity results will be given in terms of n, m, d, and $||A||$ where $d-1$ is a bound on the degrees of entries in A and $||A||$ denotes maximum magnitude of all coefficients of all entries of $A \in \mathbb{Z}[x]^{n\times m}$. We will use the parameter $s = n+m+d+\log||A||$ as a measure of the *size* of matrix $A \in \mathbb{Z}[x]^{n\times n}$. (Note that this is distinct from Size(A), which indicates the total size in bits of matrix A. In particular, $\mathrm{Size}(A) = O(s^4)$.) We use $P(d)$ to indicate the number of field operations in $\mathbf{F}$ required to multiply two polynomials of degree $d-1$ in $\mathbf{F}[x]$. Similarly, $M(t)$ shall denote the number of bit operations required to multiply two t bit integers. Using fast integer and polynomial multiplication (the Schönhage-Strassen algorithm) we can take $P(d) = d \log d \log\log d$ and $M(t) = t \log t \log\log t$. However, for SmithForm over $\mathbf{Q}[x]$ we give a solution that uses a homomorphic imaging scheme that avoids almost all computation with large integers and large degree polynomials — for this algorithm we also give a complexity result in terms of standard integer and polynomial multiplication: $P(d) = d^2$ and $M(t) = t^2$.

Before embarking on a discussion of previous results, we make some qualifying remarks about probabilistic algorithms. Let $\mathcal{A}$ be an algorithm that requires $O(p(M))$ bit operations for a single pass where M is the size of the input in bits. In general, a single pass of an algorithm returns either a result (correct or incorrect) or FAIL. Fix a positive constant ϵ such that $0 < \epsilon < 1$. Algorithm $\mathcal{A}$ is said to be Monte Carlo probabilistic if it requires $O(p(M))$ bit operations to produce a result that will be incorrect with probability less than ϵ. Algorithm $\mathcal{A}$ is said to be Las Vegas probabilistic if requires $O(p(M))$ bit operations to return a result, which is guaranteed correct, except when FAIL is returned, and this happens with probability less than ϵ. The key point here is that the probability ϵ is valid for *every* possible input. That is, there does not exist some pathological input that will *a priori* cause bad behavior of the algorithm. Consequently, for a Las Vegas algorithm we can *a priori* give an expected cost in terms of number of bit operations to produce a correct result.

The first proof that SmithForm over $\mathbf{Q}[x]$ is in $\mathcal{P}$ was given by Kaltofen, Krishnamoorthy and Saunders in [23]. Their algorithm uses the fact, a consequence of Kannan (cf. [25]), that SmithForm over $\mathrm{GF}(p)[x]$ is in $\mathcal{P}$. Given a nonsingular matrix $A \in \mathbb{Z}[x]^{n\times n}$, the algorithm computes the SNF of $A \bmod p$ for various primes p and uses Chinese remaindering to reconstruct the SNF of A over $\mathbf{Q}[x]$. A drawback of this method is the large number of primes needed to guarantee correctness; their approach calls for $\Theta(n^3 d \log nd||A||)$ image solutions.

The authors of [23] take an alternative approach and in the same paper present a parallel Monte Carlo probabilistic algorithm for SmithForm over $\mathbf{F}[x]$ for the case of square nonsingular input. This algorithm is appealing in that it admits very fast sequential solution for SmithForm over $\mathbf{Q}[x]$. In practice, the main cost of the algorithm is triangularizing via fraction-free Gaussian elimination a polynomial matrix of same dimension and with similar size entries as the input matrix. However, this SmithForm algorithm has the drawback that it may return an incorrect result which is not easily detected. In a subsequent paper, the same authors give a Las Vegas algorithm for SmithFormWithMultipliers over $\mathbf{F}[x]$ [24]. This KKS Las Vegas algorithm is based on column echelon form computation and has expected cost at least that of HermiteFormWithMultipliers.

We have made a distinction between the problem of computing pre- and post-multipliers (SmithFormWithMultipliers) and computing only the SNF (SmithForm). This is because over the input domain $\mathbf{Q}[x]$, SmithFormWithMultipliers is inherently a more difficult problem than SmithForm. The complexity of SmithFormWithMultipliers compared to SmithForm is analogous to the complexity of the extended Euclidean problem over $\mathbb{Z}[x]$ compared to a gcd computation. Consider the matrix $A \in \mathbb{Z}[x]^{2\times 1}$:

$$\overset{U}{\begin{bmatrix} u_{11} & u_{12} \\ u_{21} & u_{22} \end{bmatrix}} \overset{A}{\begin{bmatrix} a_{11} \\ a_{21} \end{bmatrix}} \overset{V}{\begin{bmatrix} v_{11} \end{bmatrix}} = \overset{S}{\begin{bmatrix} s_{11} \\ 0 \end{bmatrix}}$$

Finding the SmithForm of A is equivalent of finding $s_{11} \leftarrow \gcd(a_{11}, a_{21})$. A host of powerful techniques have developed to find gcd's over $\mathbb{Z}[x]$ that abdicate the problem of expression swell; Hensel lifing, modular homomorphisms follows by Chinese remaindering, and heuristic methods based on single point interpolation (cf. [14]). In particular, the original heuristic gcd algorithm of Char, Geddes and Gonnet [9] can be made Las Vegas probabilistic (cf. Schönhage [31]) and requires $O^\sim(d(d + ||A||))$ bit operations. On the other hand, finding candidates for the entries of U is equivalent to solving the extended Euclidean problem: find polynomials u_{11} and u_{12} such that

$$u_{11}a_{11} + u_{12}a_{21} = s_{11} \tag{5.1}$$

where s_{11} is gcd of a_{11} and a_{21}. This problem is much more difficult (in terms of bit complexity) because of the size of the coefficients of u_{11} and u_{12} (cf. §4.1). In particular, solving the extended Euclidean problem is far too much work if only the gcd is required.

This brings us to the second reason for distinguishing SmithForm and SmithFormWithMultipliers; namely, the entries in pre- and post-multipliers for a given matrix A over $\mathbf{Q}[x]$ are typically large compared to entries in A or S. Consider a nonsingular input matrix $A \in \mathbb{Z}[x]^{n\times n}$. Let U and V be pre- and post-multipliers for A such that $UAV = S$. The diagonal entries of S are factors of $\det(A)$ and hence the sum of the degrees of all entries of S are bounded by nd and the length of the rational coefficients of entries of S are bounded by $O^\sim(n(d+\log ||A||))$ bits — this leads to Size$(S) = O^\sim(n^2 d(d + \log ||A||)$ bits. To get an idea of the size of matrices U and V, we can look at the problem HermiteForm considered in the previous chapter. Theorem 6 bounds the degrees of polynomials in HNF pre-multiplier matrix U by $(n-1)d$ and the length of rational coefficients in U by $O(n^2 d \log nd||A||)$ bits. Corollary 3 gives Size$(U) = O^{\sim}(n^5 d^2 \log ||A||)$. This suggests Size$(U) \approx O^{\sim}(s^4)$Size$(S)$.

The majority of algorithms found in the literature for SmithForm over $\mathbf{F}[x]$ or $\mathbb{Z}$ are based on HermiteFormWithMultipliers and by virtue of this actually solve the more difficult problem SmithFormWithMultipliers (cf. [24, 35, 25, 21]). Specifically, these algorithms produce pre- and post-multipliers U and V such that $UAV = S$ is in SNF within the same asymptotic complexity as required to produce S alone. One reason for producing multipliers is to verify correctness. In particular, the KKS Monte Carlo SmithForm algorithm does not produce pre- and post-multipliers and may return an incorrect result which cannot be detected easily. Our algorithm differs from these methods in that we are able to completely avoid the need for solving computationally expensive polynomial diophantine equations such as (5.1) and hence obtain a better complexity result.

5.1 Preliminaries and Previous Results

In this section we recall some basic facts about the Smith and Hermite normal forms and review the Monte Carlo and Las Vegas probabilistic SNF algorithms of Kaltofen, Krishnamoorthy and Saunders [23, 24]. As well, the algorithms of the next section will depend on some elementary facts about lattices over principal ideal domains.

We require some notation. For an $n \times m$ matrix A, let minor(A, i) denote the i-th principal minor of A for $1 \leq i \leq \min(n, m)$. In general, a *minor* of A is a square submatrix of A. If A is square, then A^{adj} will denote its adjoint.

Let $A \in \mathbf{F}[x]^{n \times m}$ be of rank r with SNF S. Let $A_{i,j}$ or a_{ij} denote the entry in the i-th row j-th column of A. Let $s^*(A, i)$ or s_i^* denote the gcd of the determinants of all i by i minors of A for $1 \leq i \leq r$ and let $s(A, i)$ or s_i be the i-th diagonal entry of the SNF of A. By convention, define $s^*(A, 0) = 1$. The diagonal entry s_i is called the i-th invariant factor of A while each s_i^* is called the i-th determinantal divisor of A. The invariant factors and determinantal divisors are related by $s_i = s_i^*/s_{i-1}^*$ for $1 \leq i \leq r$. We have some similar facts for diagonal entries of the HNF. First assume that $r = m$ so A is of full column rank and let H be the HNF of A. Let $h^*(A, 0) = 1$ and let $h^*(A, i)$ or h_i^* denote the gcd of the determinants of all i by i minors of the first i columns of A. Let $h(A, i)$ or h_i be the i-th diagonal entry of the HNF of A. Then we have $h_i = h_i^*/h_{i-1}^*$ for $1 \leq i \leq m$. These facts hold in general for matrices over principal ideal domains (cf. [29]). Recall that for the invariant factors of A we have $s_i|s_{i_1}$ for $1 \leq i \leq r$. The following fact follows from the divisibility properties of the invariant factors and determinantal divisors.

Fact 2 *For a principal ideal domain* $\mathbf{R}$*, let* $g_0^*, g_1^*, \ldots, g_r^*$ *be elements from* $\mathbf{R}$ *with* $g_0^* = 1$*. Then, there exists a matrix in* $\mathbf{R}^{n\times m}$ *of rank* r *having, for* $1 \leq i \leq r$*, i-th determinantal divisor an associate of* g_i^**, if and only if*

$$g_i^{*2} | g_{i-1}^* g_{i+1}^*, \qquad 1 \leq i \leq r-1.$$

■

It will be convenient sometimes to express the output of the algorithms we present as a list $s_i^*, \ldots, s_r^*$ of determinantal divisors of the input matrix. Clearly, the SNF of a matrix A is easily determined from its determinantal divisors and vice versa. Note that Fact 2 provides a necessary (but not sufficient) condition for correctness on the output of an algorithm that returns a list of candidates for the determinantal divisors of an input matrix.

Fact 3 *Let* A *and* B *in* $\mathbf{F}[x]^{n\times n}$ *be nonsingular with* $T = UA$ *upper triangular. Then, the following statements are equivalent:*

(1) U *is unimodular;*

(2) $\det(T) \simeq \det(A)$*;*

(3) $T_{i,i} \simeq h^*(A, i)$ *for* $1 \leq i \leq n$*;*

■

5.1.1 The KKS Probabilistic Algorithms

A possible algorithm for finding s_i^* is to compute the determinants of all $i \times i$ minors of A and set s_i^* to be their gcd. Unfortunately, the number of minors increases exponentially with respect to the matrix dimension. The Monte Carlo SmithForm algorithm of [23] overcomes this problem by preconditioning the input matrix using certain random unimodular pre- and post-multipliers with entries chosen from a subset of the coefficient field. With high probability, each s_i^* can be determined by taking the gcd of only two minors.

Algorithm: KKS Monte Carlo SmithForm
Input: A nonsingular matrix $A \in \mathbf{F}[x]^{n\times n}$.

Output: $[s_1^*, s_2^*, \ldots, s_n^*]$, the determinantal divisors of A.
(1) Let U_1, U_2, V_1, V_2 be random matrices from $\mathbf{F}^{n\times n}$.
(2) Set $B = U_1 A V_1$, $C = U_2 A V_2$.
(3) Set $s_i^* = \gcd(\det(\mathrm{minor}(B,i)), \det(\mathrm{minor}(C,i)))$ for $1 \le i \le n$.

The entries of the matrices in step (1) can be chosen from a subset of $\mathbf{F}$ (cf. [23]). With high probability, the entries of U_1, U_2, V_1, V_2 do not form a root of a polynomial of large degree and the s_i^* will be correct [23, Lemma 3.2]. The authors of [23] also mention that U_1 may be chosen upper triangular, V_1 may be chosen lower triangular, and that C need not be randomized.

Example 6 (KKS Monte Carlo SmithForm**)** *Consider the matrix*

$$A = \begin{bmatrix} x-1 & 3x+2 \\ x-1 & 2x+3 \end{bmatrix}$$

We choose random unimodular matrices U_1, V_1, U_2 and V_2 and set

$$\begin{array}{ccccc} B & & U_1 & A & V_1 \\ \begin{bmatrix} 4x+1 & 3x+2 \\ 3x+2 & 2x+3 \end{bmatrix} & = & \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} & \begin{bmatrix} x-1 & 3x+2 \\ x-1 & 2x+3 \end{bmatrix} & \begin{bmatrix} 1 & 0 \\ 1 & 1 \end{bmatrix} \end{array}$$

and

$$\begin{array}{ccccc} C & & U_2 & A & V_2 \\ \begin{bmatrix} x-1 & 3x+2 \\ x-1 & 2x+3 \end{bmatrix} & = & \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} & \begin{bmatrix} x-1 & 3x+2 \\ x-1 & 2x+3 \end{bmatrix} & \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \end{array}$$

The determinants of the principal minors of B are $b_1 = 4x+1$, $b_2 = -x^2+2x-1$ and for C are $c_1 = x-1$, $c_2 = -x^2+2x-1$. As candidates for the determinental divisors of A we obtain $s^*(A,1) = \gcd(b_1,c_1) = 1$ and $s^*(A,2) = \gcd(b_2,c_2) = x^2-2x+1$. The candidate for the SNF of A is

$$S = \begin{bmatrix} 1 & 0 \\ 0 & x^2-2x+1 \end{bmatrix}.$$

Consider the computation again but this time with a different choice for the random unimodular multiplier matrices U_1 and V_1. The new choice gives

$$\begin{array}{ccccc} B & & U_1 & A & V_1 \\ \begin{bmatrix} 2x-2 & 5x+5 \\ x-1 & 2x+3 \end{bmatrix} & = & \begin{bmatrix} 1 & 0 \\ 1 & 1 \end{bmatrix} & \begin{bmatrix} x-1 & 3x+2 \\ x-1 & 2x+3 \end{bmatrix} & \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \end{array}.$$

For the new choice of B, candidates for the determinental divisors turn out to be $s^*(A,1) = x-1$ and $s^*(A,2) = x^2-2x+1$. This leads to the following (incorrect) candidate for the SNF of A:

$$S = \begin{bmatrix} x-1 & 0 \\ 0 & x-1 \end{bmatrix}.$$

■

In a later paper a Las Vegas SMITHFORM algorithm is offered [24]. For convenience, we give here a version restricted square nonsingular input.

Algorithm: KKS Las Vegas SMITHFORM
Input: A nonsingular matrix $A \in \mathbf{F}[x]^{n\times n}$.
Output: $[s_1^*, s_2^*, \ldots, s_n^*]$, the determinantal divisors of A.
(1) Let V_1 be a random unit lower triangular matrix in $\mathbf{F}^{n\times n}$.
(2) $A' \leftarrow AV_1$.
(3) $A_1 \leftarrow$ HNF of A'.
(4) $A_2 \leftarrow$ HNF of transpose(A_1).
(5) If A_2 is in SNF then output A_2 otherwise fail.

The typical algorithm for SMITHFORM involves iterating HERMITEFORM along the diagonal of the input matrix just as in steps (3) and (4) of KKS Las Vegas SMITHFORM. While in theory the number of number of iterations is bounded by $O(n^3)$, in practice two iterations almost always suffice; the KKS Las Vegas SMITHFORM ensures that with high probability this will be the case.

Example 7 (KKS Las Vegas SMITHFORM) *Consider again the matrix*

$$A = \begin{bmatrix} x-1 & 3\,x+2 \\ x-1 & 2\,x+3 \end{bmatrix}$$

We choose a random unit lower triangular matrix V_1 and set

$$\overset{A'}{\begin{bmatrix} 4\,x+1 & 3\,x+2 \\ 3\,x+2 & 2\,x+3 \end{bmatrix}} = \overset{A}{\begin{bmatrix} x-1 & 3\,x+2 \\ x-1 & 2\,x+3 \end{bmatrix}} \overset{V_1}{\begin{bmatrix} 1 & 0 \\ 1 & 1 \end{bmatrix}}$$

In step (3) we obtain

$$A_1 = \begin{bmatrix} 1 & -1/5\,x+6/5 \\ 0 & x^2-2\,x+1 \end{bmatrix},$$

the HNF of A'. In step (4) we obtain

$$A_2 = \begin{bmatrix} 1 & 0 \\ 0 & x^2 - 2x + 1 \end{bmatrix},$$

the HNF of A_1. Since A_2 is in SNF we output A_2 as the SNF of A. Consider the computation again but this time neglecting to randomize the matrix A' (i.e. we choose $V_1 = I$ in step (1) whence $A' = A$). Then we obtain

$$A_1 = \begin{bmatrix} x-1 & 5 \\ 0 & x-1 \end{bmatrix} \quad \textit{and} \quad A_2 = \begin{bmatrix} 1 & 1/5\,x - 1/5 \\ 0 & x^2 - 2x + 1 \end{bmatrix}.$$

Since A_2 is not in SNF the algorithm returns FAIL. ■

5.2 Fraction-Free Gaussian Elimination

The number fields we work with in a computational setting are typically quotient fields of principal ideal domains (e.g. the rationals $\mathbf{Q}$ or the field of rational polynomials $\mathbf{F}(x)$). For problems over these domains, it advantageous to be able to do as much computation within the simpler principal ideal domain before moving to the quotient field. For triangularizing matrices over fields, an algorithm that has proven useful in this regard is fraction-free Gaussian elimination. For a thorough discussion of fraction-free Gaussian elimination see [14] or the original articles by Bareiss [2, 3].

For $\mathbf{R}$ a principal ideal domain, let $A \in \mathbf{R}^{n \times m}$ and a positive integer $r \leq \min(m, n)$ be given. A key step in the algorithm we present in the next section is to compute a partial triangularization of A over the quotient field of $\mathbf{R}$ that is expressible entirely within $\mathbf{R}$. This will correspond to finding a lower triangular matrix $F \in \mathbf{R}^{n \times n}$ such that FA has zero entries below the diagonal in columns $1, 2, \ldots, r$. In other words, if A' is the submatrix of A comprised of the first r columns of A, then $FA' \in \mathbf{R}^{n \times r}$ will be in upper echelon form (with respect to the quotient field of $\mathbf{R}$).

Example 8 *The matrix*

$$A = \begin{bmatrix} 11 & 14 & 18 & 7 & 57 & 43 \\ 6 & 6 & 4 & 20 & 78 & 28 \\ 9 & 19 & 3 & 12 & 73 & 18 \\ 0 & 11 & 12 & 0 & 11 & 24 \\ 10 & 20 & 9 & 5 & 55 & 23 \\ 18 & 19 & 7 & 16 & 103 & 30 \\ 12 & 5 & 6 & 14 & 71 & 26 \end{bmatrix}$$

has rank 4. Triangularizing A using fraction-free Gaussian elimination and recording row operations in a bordering identity matrix yields the matrices F and T such that $FA = T$:

$$\overbrace{\begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ -6 & 11 & 0 & 0 & 0 & 0 & 0 \\ 60 & -83 & -18 & 0 & 0 & 0 & 0 \\ -918 & 2415 & -488 & 694 & 0 & 0 & 0 \\ -15822 & 14904 & -30618 & -594 & 36018 & 0 & 0 \\ -33156 & 2829 & -33398 & 36130 & 0 & 36018 & 0 \\ -30564 & -13524 & -1652 & 32758 & 0 & 0 & 36018 \end{bmatrix}}^{F = \mathrm{FFGE}(A,4)} \overbrace{\begin{bmatrix} 11 & 14 & 18 & 7 & 57 & 43 \\ 6 & 6 & 4 & 20 & 78 & 28 \\ 9 & 19 & 3 & 12 & 73 & 18 \\ 0 & 11 & 12 & 0 & 11 & 24 \\ 10 & 20 & 9 & 5 & 55 & 23 \\ 18 & 19 & 7 & 16 & 103 & 30 \\ 12 & 5 & 6 & 14 & 71 & 26 \end{bmatrix}}^{A}$$

$$= \overbrace{\begin{bmatrix} 11 & 14 & 18 & 7 & 57 & 43 \\ 0 & -18 & -64 & 178 & 516 & 50 \\ 0 & 0 & 694 & -1456 & -4368 & -68 \\ 0 & 0 & 0 & 36018 & 108054 & 36018 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}}^{T}.$$

■

Remark: Since F is non unimodular, we must have $\mathcal{L}(T) \neq \mathcal{L}(A)$. However, the point that will concern us is that since F is over $\mathbb{Z}$, we must have $\mathcal{L}(T) \subset \mathcal{L}(A)$.

The matrices F and T found using fraction-free Gaussian elimination have the property that all entries are associates of determinants of particular minors of A. Not only does this fact ensure good *a priori* bounds for the size of intermediate expressions occurring during the triangulation, but the structure and nature of the entries of F and T will prove very useful in lattice basis problems over PIDs (cf. next section and next chapter). For the remainder of this section, we give an explicit characterization of the matrices F and T and consider the problem of their computation.

First we need to recall some basic definitions and facts from linear algebra. Let $A \in \mathbf{R}^{n\times n}$ be nonsingular. The minor $M_{i,j}$ of entry a_{ij} is defined to be the determinant of the submatrix obtained by deleting the the i-th row and j-th column of A. The cofactor C_{ij} is given by $C_{ij} = (-1)^{i+j} M_{ij}$.

Fact 4 *Let A be an $n \times n$ matrix over $\mathbf{R}$ with adjoint A^{adj}. Then, the entries in A^{adj} are given by $A^{\mathrm{adj}}_{ij} = C_{j,i}$ for $1 \le i, j \le n$. Also,* $\det(A)$ *is given by the i-th row expansion,*

$$\det(A) = \sum_{j=1}^{n} a_{i1}C_{i1} + a_{i2}C_{i2} + \cdots + a_{in}C_{in}$$

or the j-th column expansion

$$\det(A) = \sum_{i=1}^{n} a_{1j}C_{1j} + a_{2j}C_{2j} + \cdots + a_{nj}C_{nj}.$$

■

Note that the last row of A^{adj} will consist entirely of associates of $(n-1)\times(n-1)$ minors of the first $n-1$ columns of A. In particular, each entry of of the last row of A^{adj} will be divisible by $h^*(A, i-1)$. Now consider the lower triangular matrix $F \in \mathbf{R}^{n\times n}$ with $[u_{i,1}, u_{i,2}, \ldots, u_{i,i}]$ equal to the last row of the adjoint of $\mathrm{minor}(A, i)$ for $1 \le i \le n$. Then F is lower triangular with: (1) $F_{1,1} = 1$ and $F_{i,i} = \det(\mathrm{minor}(A, i-1))$ for $2 \le i \le n$; (2) each entry to the left of the diagonal in row i of F equal to an associate of an $(i-1)\times(i-1)$ minor of the first $i-1$ columns of A for $2 \le i \le n$. Furthermore, FA will be upper triangular with: (1) $(FA)_{i,i} = \det(\mathrm{minor}(A, i))$ for $1 \le i \le n$; (2) each entry to the right of the diagonal in row i of FA equal to the determinant of an $i \times i$ minor of A. The following lemma generalizes this idea to rectangular matrices.

Lemma 7 *Let $A \in \mathbf{R}^{n\times m}$ and a positive integer $r \le \min(m, n)$ be given. Let $F \in \mathbf{R}^{n\times n}$ be lower triangular with:*

(1) $F_{i,j}$ equal to the cofactor of the element in the j-th row, i-th column of the i-th principal minor of A for $1 \le j \le i \le r$,

(2) $F_{i,j}$ equal to the cofactor of the element in the j-th row, r-th column of the submatrix of A formed from rows $[1, 2, \ldots, r-1, i]$ and columns $[1, 2, \ldots, r]$ of A for $r+1 \le i \le n$, $1 \le j \le r$,

(3) $F_{i,j} = 0$ *for* $r + 2 \leq i \leq n$, $r + 1 \leq j \leq i$,

(4) $F_{i,i} = \det(\mathrm{minor}(A, r))$ *for* $r + 1 \leq i \leq n$.

Then, the matrix $T = FA$ *will have:*

(1) $T_{i,i} = \det(\mathrm{minor}(A, i))$ *for* $1 \leq i \leq r$;

(2) $T_{i,j}$ *equal to the determinant of the* $i \times i$ *submatrix of* A *formed from rows* $[1, 2, \ldots, i]$ *and columns* $[1, 2, \ldots, i - 1, j]$ *for* $1 \leq i \leq n$, $i < j \leq m$;

(3) $T_{i,j} = 0$ *for* $1 \leq j \leq r$, $j + 1 \leq i \leq n$;

(4) $T_{i,j}$ *equal to the determinant of the* $(r + 1) \times (r + 1)$ *submatrix of* A *formed from rows* $[1, 2, \ldots, r, i]$ *and columns* $[1, 2, \ldots, r, j]$

Remark: When $r \geq \mathrm{rank}(A)$, matrix T will be upper triangular.

Proof: Follows from Fact 4 by noting that the entries of FA are the claimed entries for T — which are determinants of submatrices of A — written according according to their cofactor expansion. ∎

In what follows we use FFGE(A, r) to denote the matrix F of Lemma 7. FFGE(A, r) can be found in $O(nmr)$ ring operations by recording row operations in a companion matrix while reducing A to upper echelon form (columns $1, \ldots, r$) using a variation of the Bareiss single-step fraction-free Gaussian elimination scheme that avoids row switching. The usual Gaussian elimination method zeroes out entries below the diagonal element in the k-th column of A for $k = 1, 2, \ldots, r$. When working on column k, the first step is to switch rows (if necessary) to ensure that the pivot entry (the diagonal entry in column k) is nonzero. Switching of rows will not be necessary only if, for $k = 1, \ldots, r$, the initial diagonal entry in the k-th column of the matrix being reduced is nonzero after columns $1, \ldots, k - 1$ have been reduced. This will happen precisely when minor(A, r) is definite (minor(A, i) nonsingular for $i = 1, 2, \ldots, r$). For our purposes, we may assume that minor(A, r) is definite and when computing FFGE(A, r) return FAIL if a zero pivot is encountered. Hence, for nonsingular definite input $A \in \mathbf{F}[x]^{n \times n}$ with degrees of entries bounded by $d - 1$, FFGE(A, n) can be computed using $O(n^3 P(nd))$ field operations with using fraction free Gaussian elimination. Assuming fast polynomial multiplication yields a cost of $O(n^4 d \log nd \log\log nd)$ or $O^{\sim}(n^4 d)$ field operations.

However, it is desirable to be able to compute FFGE(A, r) even when minor(A, r) is not definite. When $\mathbf{R}$ is a polynomial domain, say $\mathbf{Q}[x]$, then an efficient method to compute FFGE(A, r) is based on modular/evaluation homomorphisms and computation of FFGE(A, r) in the image domain (integers modulo a prime) followed by interpolation and Chinese remaindering. It may happen that $\det(\text{minor}(A,i))$ maps to zero in the image domain for some $1 \leq i \leq r$ even when $\text{minor}(A,r)$ is definite over $\mathbf{Q}[x]$. Clearly, though, the matrices F and T of Lemma 7 are well defined even when this situation occurs, and a simple modification of the Bareiss method obviates the problem of zero pivots. To illustrate, we offer here an algorithm that finds $\text{FFGE}(A,n)$ for *any* square matrix of size n. Let $\texttt{BAREISS}(A,i)$ denote a procedure that returns the 3-tuple (F_i, T_i, k_i) where F_i and T_i, matrices such that $F_iA = T_i$, are found by applying ordinary fraction-free Gaussian elimination to A, but with row switches (used to choose nonzero pivots) limited to the first i rows of A, and where k_i is the maximal column index for which the reduction can proceed (i.e. for which a nonzero pivot can be chosen.)

Algorithm: `FFGE`
Input(A) # an $n \times n$ matrix over $\mathbf{R}$
Output(F,T) # the matrix $F = \text{FFGE}(A,n)$ and $T = FA$

```
1 for i = 1 to n do
2    (F_i, T_i, k_i) ← BAREISS(A, i);
3    if k_i ≥ i − 1 then
4       row(F, i) ← row(F_i, i)
5    else
6       row(F, i) ← [0, 0, ..., 0];
7 od;
8 T ← FA
```

Remarks:
(1) In line 2, F_i and T_i can be computed by continuing the Bareiss method on matrices F_{i-1} and T_{i-1}.
(2) `BAREISS` is called n times, but on pass i, only $k_i - k_{i-1}$ columns of A have entries below the diagonal zeroed. In total, $\sum_{2 \leq i \leq n} k_i - k_{i-1} \leq n$ columns will have entries below the diagonal zeroed. This shows the cost of algorithm `FFGE` is same as for $\texttt{BAREISS}(A,n)$.

Consider again the problem of computing $F = \text{FFGE}(A,n)$ for a square nonsingular matrix A over $\mathbf{F}[x]^{n \times n}$, but now assuming standard polynomial multiplication: $P(d) = d^2$. Using algorithm `FFGE` we can apply an evaluation/interpolation scheme to find F and $T = FA$ without concern about "bad" homomorphisms. The procedure can be described as follows: (1) Find the matrices $A|_{x=i}$ for $i = 0, \ldots, nd$ at a cost of $O(n^2 \cdot nd \cdot d)$ field operations; (2) Find $F|_{x=i}$ and $T_{x=i}$ for $i = 0, \ldots, nd$

at a cost of $O(nd{\cdot}n^3)$ field operations; (3) Use Chinese remaindering to reconstruct the $2n^2$ degree nd polynomials in matrices F and T from their images at a cost of $O(n^2(nd)^2)$ field operations. Combining the above yields a total cost of $O(n^4d^2)$ field operations using standard polynomial arithmetic to compute FFGE(A, n).

5.2.1 Computing FFGE(A, r) over $\mathbb{Z}[x]$

Let $A \in \mathbb{Z}[x]^{n\times m}$ have degree bounded by $d-1$. Recall that $||A||$ denotes the largest magnitude of all coefficients of all entries of A. The purpose of this section is demonstrate that F =FFGE(A, r) (and $T = FA$) can be found in $O^{\sim}(s^8)$ bit operations using standard integer and polynomial multiplication. (Recall that $s = n + m + d + \log ||A||$.) To find F =FFGE(A, r), we find $F_p = F \bmod p \in \mathrm{GF}(p)[x]^{n\times n}$ for sufficiently many primes p to allow recovery of F via the Chinese remainder algorithm. Determinants of $r \times r$ minors of A (and hence entries of F and T) will be polynomials bounded in degree by rd and have coefficients bounded in magnitude by $B = (\sqrt{r}d||A||)^r$. The following lemma from Giesbrecht shows that we can choose all our primes to be $l = 6 + \log\log B$ bits in length.

Lemma 8 (Giesbrecht [15]) *Let $x \geq 3$ and $l = 6 + \log\log x$. Then there exist at least $2\lceil\lceil\log_2(2x)\rceil/(l-1)\rceil$ primes p such that $2^{l-1} < p < 2^l$.*

We need to find F_p for $q = \lceil(2B+1)/2^{l-1}\rceil = \Theta(r\log rd||A||/l)$ primes $(p_i)_{1\leq i\leq q}$ that are each l bits in length. F and FA will contain at most $2nm$ nonzero entries bounded in degree by rd. Hence, the Chinese remainder algorithm will need to be used to reconstruct at most $2nm(rd+1))$ integers $\log B$ bits in length from their images modulo a set of q primes bounded in length by l bits. This yields $O(nmrdM(\log B)\log\log B) = O^{\sim}(nmr^3d\log^2||A||)$ bit operations using standard integer arithmetic to recover F and T. The images (F_{p_i}, T_{p_i}) over $\mathrm{GF}(p)[x]$ can be computed using an evaluation/interpolation scheme: (1) Find the images $(A_{p_i})_{1\leq i\leq q}$; (2) For $1 \leq i \leq q$ and $0 \leq j \leq rd$, compute $(F_{p_i}|_{x=j}, T_{p_i}|_{x=j})$ over $\mathrm{GF}(p)[x]$ at a cost of $O(q \cdot nmr((P(d)\log d + rd)M(l))$ bit operations; (3) For $1 \leq i \leq q$, interpolate the $O(nm)$ nonzero entries of (F_{p_i}, T_{p_i}) at a cost of $O(q \cdot nmP(rd)(\log rd)M(l))$ bit operations. The cost of step (1) will be bounded by the cost of Chinese remaindering. Combining steps (2) and (3) together with the cost of Chinese remaindering gives the following result.

Theorem 8 *Let $A \in \mathbb{Z}[x]^{n\times m}$ with degrees bounded by $d-1$ and a positive integer $r \leq \min(m, n)$ be given. The matrices F =FFGE(A, r) and $T = FA$ can be found in $O^{\sim}(nmr^3d(d + \log||A||)\log||A||)$ bit operations using standard integer and polynomial arithmetic.*

Corollary 4 *F =FFGE$(A, \min(m,n))$ and $T = FA$ can be found in $O^{\sim}(s^8)$ bit operations using standard integer and polynomial arithmetic.*

5.3 An Algorithm for SmithForm over $\mathbf{F}[x]$

In this section we give an extension of the KKS Monte Carlo SmithForm algorithm given in §5.1.1 that verifies correctness of the determinantal divisors found. Given a nonsingular input matrix A in $\mathbf{F}[x]^{n\times n}$, the first step of the algorithm is to precondition A with random pre- and post-multipliers to obtain a new matrix A' that has the same SNF as A. Using fraction free Gaussian elimination, a lower triangular matrix U_T in $\mathbf{F}[x]^{n\times n}$ is found such that $T = U_T A'$ is upper triangular with $T_{i,i}$ being the determinant of the i-th principal minor of A'. The algorithm then computes g_i^*, the gcd of $\det(A)$ and the determinant of the i-th minor of A'. With high probability, g_i^* will equal the i-th determinantal divisor of A. The remainder of the algorithm performs $O(n^2)$ divisibility checks which all hold if and only if all the g_i^* are indeed the desired determinantal divisors. In this section we restrict our attentions to square nonsingular input.

To bound the probability of failure by a constant ϵ, where $0 < \epsilon < 1$, we require that $\#\mathbf{F} \geq 6n^3 d/\epsilon$. The SNF of of A over $\mathbb{K}[x]$ (where $\mathbb{K} < 6n^3 d/\epsilon$) can be found by computing over an algebraic extension $\mathbf{F}$ of $\mathbb{K}$ having the required number of elements; the SNF is an entirely rational form and will not change if we compute over an extension field $\mathbf{F} \supset \mathbb{K}$ of the coefficient field. In any case, our main motivation is the case when the coefficient field $\mathbf{F}$ has characteristic zero. As discussed in the introduction, SmithForm over $\mathbf{F}[x]$ can be especially difficult for the case when $\mathbf{F}$ has characteristic zero because of the potential for intermediate expression swell experienced by the coefficients from $\mathbf{F}$. Algorithm `SquareSmithForm` that follows works particularly well for coefficient fields $\mathbf{F}$ that are the quotient fields of a non finite integral domains. In particular, $\mathbf{Q}$ is the quotient field of $\mathbb{Z}$. Without loss of generality, we assume that an input matrix $A \in \mathbf{Q}[x]^{n\times n}$ has all integer coefficients; in this case the algorithm finds associates of the determinantal divisors of A while keeping all computations within the simpler domain $\mathbb{Z}[x]$. More importantly, we can obtain very good bounds on the size of integers occurring as intermediate expressions.

Algorithm: `SquareSmithForm`
Input: A nonsingular matrix $A \in \mathbf{F}[x]^{n\times n}$ and an upper bound $0 < \epsilon < 1$ on the probability of failing.
Output: $[s_1^*, s_2^*, \ldots, s_n^*]$, the determinantal divisors of A.

(1) [Randomize:]
Let $d-1$ bound the degrees of entries of A and let C be a subset of $\mathbf{F}$ with $\#C = \lceil 6n^3 d/\epsilon \rceil$.
$U_R \leftarrow$ a unit upper triangular matrix with off diagonal elements chosen at random from C;
$V_R \leftarrow$ a unit lower triangular matrix with off diagonal elements chosen at random from C;
$A' \leftarrow U_R A V_R$;

(2) [Triangularize:]
$U_T \leftarrow \mathrm{FFGE}(A', n)$;
$T \leftarrow U_T A'$;

(3) [Find probable determinantal divisors of A:]
$d^* \leftarrow \det(A)^2$;
for $i = 1$ to n do
  $g_i^* \leftarrow$ an associate of $\gcd(d^*, T_{i,i})$;

(4) [Check divisibility properties of g_i^*'s:]
$g_0^* \leftarrow 1$;
for $i = 1$ to $n-1$ do
  if g_i^{*2} does not divide $g_{i-1}^* g_{i+1}^*$ then FAIL;

(5) [Assay that $g_i^* = h^*(A', i)$ for $1 \le i \le n$:]
for $i = 2$ to n do
  for $j = 1$ to $i-1$ do
    if g_{i-1}^* does not divide $U_{T\,i,j}$ then FAIL;

(6) [Assay that $g_i^* = s^*(A', i)$ for $1 \le i \le n$:]
for $i = 1$ to $n-1$ do
  for $j = i+1$ to n do
    if g_i^* does not divides $T_{i,j}$ then FAIL;

(7) [Output:]
$[s_1^*, s_2^*, \ldots, s_n^*]$ with s_i^* the monic associate of g_i^* for $1 \le i \le n$;

Remarks:
(1) Note that $\det(A)$ has already been computed when step (3) is reached since $T_{n,n} = \det(A)$.
(2) In step (3), we may choose $d^* \leftarrow \det(A)$ rather than $d^* \leftarrow \det(A)^2$.

Entries of U_T and T found in step (2) are associates of determinants of minors of A'; these have degrees bounded by nd. This leads to a bound of $2nd$ on degrees of all polynomials occurring in the algorithm. The n gcd computations in step (3) will require at most $O(nP(nd)\log nd)$ field operations. Using fraction free Gaussian elimination, the matrices U_T and T can be found in $O(n^3P(nd))$ field operations. The matrix multiplications in step (1) and the remaining n multiplications and n^2-1 trial divisions in steps (4), (5) and (6) can be accomplished in $O(n^2P(nd))$ field operations. Overall we obtain $O(n^2d(n^2+\log d)\log nd\log\log nd)$ or $O^{\sim}(n^4d)$ field operations using fast polynomial arithmetic: $P(d) = d\log d\log\log d$. In practice, we would use the evaluation/interpolation scheme discussed in §5.2 to recover U_T and T — this leads to a bound of $O(n^4d^2)$ field operations for algorithm `SquareSmithForm` using standard polynomial arithmetic: $P(d) = d^2$.

We now consider the special case when $\mathbf{F} = \mathbf{Q}$ and the input matrix $A \in \mathbf{Q}[x]$ has all integer coefficients. For this case, we derive a complexity result in terms of n, d and β where β is a bound for both $||U_T||$ and $||T||$ where U_T and T are the matrices found in step (2). Note that in step (1) we can choose $C = \{0,\ldots,\lceil 6n^3d/\epsilon\rceil\}$ so that $||A'|| \leq n\cdot\lceil 6n^3d/\epsilon\rceil\cdot||A||$ whence $\beta \leq (\sqrt{n}d||A'||)^n \leq (\sqrt{n}d\lceil 6n^3d/\epsilon\rceil||A||)^n$ or, asymptotically, $\log\beta = O(n\log nd||A||)$.

There is a natural duality between the integers and univariate polynomials with integer coefficients. The integer coefficients (represented in binary) of a degree $d-1$ polynomial $f \in \mathbb{Z}[x]$ having coefficients bounded in magnitude by $2^{k-1}-1$ $(k \in \mathbb{Z})$ can be written as a binary lineup to obtain the dk bit integer $f|_{x=2^k}$. This corresponds to the B-adic expansion of an integer; choosing B a power of 2 allows the conversion to and from polynomial representation to be accomplished in linear time. Thus, we can find U_T and T in $O(n^3M(ndk))$ bit operations by applying fraction-free Gaussian elimination to the $n\times n$ integer matrix $A'|_{x=2^k}$ where $k = \lceil 1+\log(\beta+1)\rceil$. By a result of Schönhage [31], the n gcd computations in step 3 require $O^{\sim}(n\cdot nd(nd+n\log nd||A||))$ bit operations.

The remaining $O(n^2)$ trial divisions in steps (4), (5) and (6) and the $O(n)$ polynomial multiplications in step (3) will require at most $O(n^2M(nd\cdot(nd+n\log nd||A||)))$ bit operations. Overall this yields $O^{\sim}(n^3d(d+n^2\log||A||)) = O^{\sim}(s^7)$ bit operations using fast polynomial and integer arithmetic.

In practice, the dominant cost of the algorithm will almost certainly be finding the triangulation T and transition matrix U_T in step (2). (This would not be true, for example, if the input matrix were of dimension 2×1, in which case the computation reduces to a gcd computation over $\mathbb{Z}[x]$.) Since the size of intermediate expressions occurring in the algorithm admit very good bounds, a complexity result in terms of fast polynomial and integer arithmetic will not be interesting for many practical examples. By employing the homomorphic imaging scheme discussed in §5.2.1 we also have the following result as a corollary of Theorem 8.

Corollary 5 *Let $A' \in \mathbb{Z}[x]^{n\times n}$ be nonsingular with degrees bounded by $d-1$. Then, there exists a sequential deterministic algorithm that finds the matrices U_T and $T = U_T A'$ in step 2 of* `SquareSmithForm` *in $O^\sim(n^5 d(d+\log ||A'||)\log ||A'||) = O^\sim(s^8)$ bit operations using standard integer and polynomial arithmetic.*

To show that algorithm `SquareSmithForm` is a correct Las Vegas algorithm for SmithForm over $\mathbf{F}[x]$ will require some lemmas. Some of the following results are more general than we require here but they will be called upon in the next chapter.

Lemma 9 *Let $A \in \mathbf{F}[x]^{n\times n}$ be square nonsingular and let $U^{(1)}$ and $U^{(2)}$ be matrices in $\mathbf{F}[x]^{n\times n}$ and $g_0^*, g_1^*, \ldots, g_n^*$ polynomials in $\mathbf{F}[x]$. Then, if*

(1) $T^{(1)} = U^{(1)}A$ and $T^{(2)} = U^{(2)}A$ are upper triangular matrices in $\mathbf{F}[x]^{n\times n}$;

(2) $g_0^* \simeq 1$ and $g_i^* \simeq \gcd(T_{i,i}^{(1)}, T_{i,i}^{(2)})$ for $1 \le \imath \le m$;

(3) $g_n^* \simeq \det(A)$;

(4) g_i^* divides each entry in row $i+1$ of $U^{(1)}$ and $U^{(2)}$ for $1 \le \imath \le n-1$;

then $g_i^* \simeq h^*(A,i)$ for $1 \le i \le n$.

Proof: Let $A, U^{(1)}, U^{(2)}$ be matrices and $g_0^*, g_1^*, \ldots, g_n^*$ polynomials that satisfy the conditions of the lemma. A nonsingular implies $g_n^* \ne 0$. Condition (4) implies $g_{i-1}^* | \gcd((U^{(1)}A)_{i,i}, (U^{(2)}A)_{i,i}) = \gcd(T_{i,i}^{(1)}, T_{i,i}^{(2)}) \simeq g_i^*$ for $1 \le i \le n$ whence $g_0^* | g_1^* | \cdots | g_n^* \ne 0$ which shows $g_i^* \ne 0$ for $1 \le i \le n$. We show by construction that there exists a matrix $U \in \mathbf{F}[x]^{n\times n}$ such that UA has i-th diagonal entry g_i^*/g_{i-1}^*; the desired result will then follow by Fact 3 and the fact the $\det(UA) = \prod_{i=1}^n g_i^*/g_{i-1}^* = g_n^* \simeq \det(A)$. Condition (2) implies that there exists a solution (a_i, b_i) to the polynomial diophantine equation

$$a_i T_{i,i}^{(1)} + b_i T_{i,i}^{(2)} = g_i^*.$$

Let E and F be diagonal matrices in $\mathbf{F}[x]^{n\times n}$ such that for $1 \le i \le n$, $(E_{i,i}, F_{i,i})$ is such solution for (a_i, b_i). Let $G \in \mathbf{F}[x]^{n\times n}$ be diagonal with $G_{i,i} = g_{i-1}^*$ for $1 \le i \le n$. Now consider the matrix

$$U = EG^{-1}U^{(1)} + FG^{-1}U^{(2)}.$$

Condition (3) implies that U is in $\mathbf{F}[x]^{n\times n}$ (not just $\mathbf{F}(x)^{n\times n}$). Then UA is also over $\mathbf{F}[x]$:

$$\begin{aligned} UA &= (EG^{-1}U^{(1)} + FG^{-1}U^{(2)})A \\ &= G^{-1}(ET^{(1)} + FT^{(2)}) \\ &= G^{-1}\begin{bmatrix} g_1^* & a_1T_{1,2}^{(1)} + b_1T_{1,2}^{(2)} & a_1T_{1,3}^{(1)} + b_1T_{1,3}^{(2)} & \cdots & a_1T_{1,n}^{(1)} + b_1T_{1,n}^{(2)} \\ 0 & g_2^* & a_2T_{2,3}^{(1)} + b_2T_{2,3}^{(2)} & \cdots & a_2T_{2,n}^{(1)} + b_2T_{2,n} \\ 0 & 0 & \ddots & & \vdots \\ \vdots & & & & \\ 0 & 0 & \cdots & 0 & g_n^* \end{bmatrix} \end{aligned}$$

The last equations shows that UA is upper triangular with $(UA)_{i,i} = g_i^*/g_{i-1}^*$. ∎

Lemma 10 *Let $T \in \mathbf{F}[x]^{n\times m}$ be of rank r with* minor(T,r) *upper triangular and with rows $r+1,\ldots,n$ having all zero entries. Let t_i denote the i-th diagonal entry of T. If t_i divides all off-diagonal entries of row i of T for $1 \le i \le r-1$, then there exists a unimodular matrix $V \in \mathbf{F}[x]^{m\times m}$ such that TV is diagonal with $(TV)_{i,i} = t_{i,i}$ for $i = 1,2,\ldots,r-1$ and $(S_TV)_{r,r} \simeq \gcd(T_{r,r+1}, T_{r,r+2},\ldots,T_{r,m})$.*

Proof: Without loss of generality, we assume that T has row dimension r (since the last $n-r$ rows have all zero entries they will not be modified by column operations). Then we can write T in block form as

$$\begin{aligned} T &= \begin{bmatrix} T_1 & T_2 \\ 0 & T_4 \end{bmatrix} \\ &= \left[\begin{array}{cccc|cccc} t_1 & t_{1,2} & \cdots & t_{1,r-1} & t_{1,r} & t_{1,r+1} & \cdots & t_{1,m} \\ 0 & t_2 & \cdots & t_{2,r-1} & t_{2,r} & t_{2,r+1} & \cdots & t_{2,m} \\ \vdots & & \ddots & \vdots & \vdots & & & \vdots \\ 0 & 0 & \cdots & t_{r-1,r-1} & t_{r-1,r} & t_{r-1,r+1} & \cdots & t_{r-1,m} \\ \hline 0 & 0 & \cdots & 0 & t_r & t_{r,r+1} & \cdots & t_{r,m} \end{array}\right] \end{aligned}$$

Define D to be a square diagonal matrix of size $r-1$ with i-th diagonal entry t_i, and let $V_4 \in \mathbf{F}[x]$ be a unimodular matrix of size $m-r+1$ such that $S_4 = T_4V_4$ is in SNF. Then, the matrix $(D^{-1}T_1)$ will be unimodular over $\mathbf{F}[x]$ and S_4 will have a single nonzero entry, namely $(S_4)_{1,1} \simeq \gcd(T_{r,r+1}, T_{r,r+2},\ldots,T_{r,m})$. We set

$$V = \begin{bmatrix} (D^{-1}T_1)^{-1} & (D^{-1}T_1)^{-1}T_2 \\ 0 & V_4 \end{bmatrix}.$$

To complete the proof, note that V is unimodular and that

$$\begin{aligned} TV &= \begin{bmatrix} T_1 & T_2 \\ 0 & T_4 \end{bmatrix} \begin{bmatrix} (D^{-1}T_1)^{-1} & -(D^{-1}T_1)^{-1}T_2 \\ 0 & I \end{bmatrix} \\ &= \begin{bmatrix} T1(D^{-1}T_1)^{-1} & -T_1(D^{-1}T_1)^{-1}T_2 + T_2 \\ 0 & T_4 I \end{bmatrix} \\ &= \begin{bmatrix} D & 0 \\ 0 & S_4 \end{bmatrix} \end{aligned}$$

■

Theorem 9 *Algorithm* `SquareSmithForm` *is correct and fails with probability less than* ϵ. *The expected cost of finding the SNF of a nonsingular input matrix* $A \in \mathbf{F}[x]$ *with degrees of entries bounded by* $d-1$ *is* $O\tilde{~}(n^4 d)$ *field operations. When* $A \in \mathbb{Z}[x]^{n\times n}$, *finding the SNF of* A *over* $\mathbf{Q}[x]$ *requires an expected number of* $O\tilde{~}(s^7)$ *bit operations.*

Proof: First we show correctness. Let g_i^* for $1 \le i \le n$ be as found in step (3). We show that the algorithm does not return FAIL if and only if g_i^* is an associate of $s^*(A,i)$ over $\mathbf{Q}[x]$ ($g_i^* \simeq s^*(A,i)$) for $1 \le i \le n$.

(*If:*) Assume that $g_i^* \simeq s^*(A,i)$ for $1 \le i \le n$. By Fact 2, step (4) will not abort. By construction, A' is equivalent to and hence has the same determinantal divisors as A. By construction of U_T in step (2), $U_{T\,i,j}$ for $i > 1$, $1 \le j \le i$ is an associate of an $(i-1)\times(i-1)$ minor of A'. Similarly, $T_{i,j}$ is an $i \times i$ minor of A' for $1 \le i \le j \le n$. Since, by assumption, g_i^* is the gcd of all $i\times i$ minors of A (and of A'), neither steps (5) nor (6) abort.

(*Only if:*) Assume that the algorithm does not return FAIL. Set $U^{(1)} = d^*\mathrm{adjoint}(A')$ and $U^{(2)} = U_T$ where $d^* = \det(A)^2$ as in step (3). Then, the matrices $U^{(1)}, U^{(2)}, A'$ and polynomials $g_1^*, g_2^*, \ldots, g_n^*$ satisfy conditions of Lemma 9. We conclude the existence of a unimodular matrix U in $\mathbf{F}[x]^{n\times n}$ such that

$$\begin{aligned} UA' &= (EG^{-1}U^{(1)} + FG^{-1}U^{(2)})A' \\ &= G^{-1}(ET^{(1)} + FT^{(2)}) \\ &= G^{-1} \begin{bmatrix} g_1^* & b_1T_{1,2}^{(2)} & b_1T_{1,3}^{(2)} & \cdots & b_1T_{1,n}^{(2)} \\ 0 & g_2^* & b_2T_{2,3}^{(2)} & \cdots & b_2T_{2,n}^{(2)} \\ 0 & 0 & \ddots & & \vdots \\ \vdots & & & & \\ 0 & 0 & \cdots & 0 & g_n^* \end{bmatrix} \end{aligned}$$

where $T^{(1)}, T^{(2)}, G$ and the a_i, b_i are as in Lemma 9. Note that UA' has i-th diagonal entry g_i^*/g_{i-1}^*. The success of step (6) implies that UA satisfies the conditions of Lemma 10 and we conclude that there exists a unimodular matrix V such that UAV is diagonal with i-th diagonal entry g_i^*/g_{i-1}^* for $1 \le i \le n$. Finally, the success of step (4) together with Fact 3 gives the desired result.

It remains to show that the probability of failure is less than ϵ. To do this, we show that $g_i^* \simeq s_i^*$ for $1 \le i \le n$ provided the entries of U_R above the diagonal and V_R below the diagonal do not form the root of a certain polynomial π with degree bounded by $4n^3d$. Then, our choice of entries for U_R and V_R allows us to apply a result of Schwartz [32] to bound the probability of failure by ϵ. By Lemma 12, the matrix AV_R will be such that $h^*(AV_R, i) = s^*(A, i)$ for $1 \le i \le n$ unless the entries of V_R below the diagonal form a root of a polynomial π_S with degree bounded by $2n^3d$. Similarly, $\gcd(\det(\text{minor}(A', i), \det(A)) = h^*(AV_R, i)$ for $1 \le i \le n$ if the entries of U_R above the diagonal do not form a root of a polynomial $\pi_H = \pi_1\pi_2\cdots\pi_{n-1}$ where each π_i, bounded in degree by $4ndi$, is as in Lemma 13. π_H will be bounded in degree by $4n^3d$. Let $\pi = \pi_S\pi_H$. Then π is bounded in degree by $6n^3d$. ■

Lemma 11 ([24, Lemma 3.5]) *Let $f_1, \ldots, f_t$ be polynomials in $\mathbf{F}[\rho, x]$, $\bar{\rho}$ is a list of new variables, with $\det f_i \le e$. Then for some $\bar{e} \le 2e$, there exists an $\bar{e} \times \bar{e}$ determinant Δ in $\mathbf{F}[\bar{\rho}]$, whose entries are coefficients of f_i, such that for any evaluation $\bar{\rho} \to \bar{r}$ a list of corresponding field elements that are not a root of Δ,* $\gcd(f_1(\bar{\rho}), \ldots, f_t(\bar{\rho})) = (\gcd(f_1, \ldots, f_t))(\bar{\rho})$.

Lemma 12 ([24, Lemma 3.7]) *Let A be a matrix in $\mathbf{F}[x]^{n\times m}$ of rank r and with the degrees of the entries bounded by d, and let $i \in \{1, \ldots, r-1\}$. Then there is a polynomial π_i in $m(m-1)/2$ variables such that if*

(1) V_R in $\mathbf{F}[x]^{m\times m}$ is unit lower triangular,

(2) A_s is the submatrix of AR comprised of the first r columns.

then A_s has rank r, and $s^*(A, i) = h^*(A_s, i)$, unless the $m(m-1)/2$ entries below the diagonal in V_R form a root of π_i. The degree of π_i is not more than $2i^2d + i$.

Lemma 13 *Let A be a matrix in $\mathbf{F}[x]^{n\times m}$ of rank m and with the degrees of the entries bounded by d, and let $i \in \{1, \ldots, m-1\}$. Then there is a polynomial π_i in $n(n-1)/2$ variables such that if*

(1) $U_R \in \mathbf{F}[x]^{n\times n}$ is unit upper triangular,

(2) d^* *is a polynomial with degree less than* $2md$ *and such that* $h^*(A,i) \mid d^*$.

then $h^*(A,i) = \gcd(d^*, \det(\text{minor}(U_R A, i))$*, unless the* $n(n-1)/2$ *entries above the diagonal in* U_R *together with the* $2md$ *coefficients of* d^* *form a root of* π_i*. The degree of* π_i *is bounded* $4mdi$*.*

Proof: First consider that case where the matrix U_R contains indeterminants as entries, say $(U_R)_{i,j} = \rho_{i,j}$ for $j > i$ where $\bar{\rho} = (\rho_{i,j})_{1 \leq i \leq n, j > i}$ is a list of indeterminants. By a result of Kaltofen, Krishnamoorthy and Saunders, [24, Lemma 3.6], we have $\det(\text{minor}(U_R A, i)) = h^*(A,i)p$, where p is an irreducible polynomial in $\mathbf{F}[x, \bar{\rho}] \setminus \mathbf{F}[x]$ or is 1. Since d^* is independant of the indeterminants $\bar{\rho}$, we must have $h^*(A,i) = \gcd(d^*, \det(\text{minor}(U_R A, i)))$ as required. An application of Lemma 11 yields the existence of a $4md \times 4md$ determinant Δ, whose entries are coefficients of x of $\det(\text{minor}(U_R A, i))$ and d^* such that for any evaluation $\bar{\rho} \rightarrow \bar{r}$, where $\bar{r}$ is a list of corresponding field elements that are not a root of Δ, $\gcd(d^*, \det(\text{minor}(U_R A, i))) = h^*(A,i)$. It remains to establish a degree bound for Δ. Coefficients of x of $U_R A$ are of degree 1 whence coefficients of x of $\det(\text{minor}(U_R A, i))$ will have total degrees bounded by i. This leads to a bound on the total degree of Δ of $4mdi$. Finally, set $\pi_i = \Delta$ to complete the proof. ■

Chapter 6

Homomorphisms for Lattice Bases over $\mathbf{F}[x]$

This chapter is the culmination of our work with matrix polynomial normal forms. Here, we bring together many of the results from previous chapters and construct algorithms for obtaining bases for the lattices generated by the rows of matrices over $\mathbf{F}[x]$. Each section in this chapter presents a new algorithm. In general, subsequent sections will depend on results from the previous sections but we present the algorithms separately because we feel each to be useful in its own right.

Section §6.1 gives an algorithm for finding a unimodular triangularization (but not necessarily the unique HNF) of a square nonsingular input matrix. Section §6.2 shows how to apply homomorphisms to the problem HermiteForm over $\mathbf{Q}[x]$ in order to avoid computation with large integers and we give an asymptotically fast sequential deterministic algorithm for HermiteForm over $\mathbf{Q}[x]$ that assumes fast polynomial and integer multiplication. On the other hand, the homomorphism method of §6.2 is also the most practical method for HermiteForm that we know of. Section §6.3 shows how to handle rectangular input via a pre-conditioning algorithm that reduces the problem to the square nonsingular case. Finally, §6.4 gives a generalization of the algorithm `SquareSmithForm` of §5.3 that works for rectangular and/or singular input.

6.1 Triangularizing

A common theme in linear algebra problems is that of triangularizing matrices. For matrices over PIDs we are interested especially in unimodular triangularizations: given $A \in \mathbf{F}[x]^{n\times m}$ with rank m, find a unimodular matrix $U \in \mathbf{F}[x]^{n\times n}$ and an upper triangular matrix $T \in \mathbf{F}[x]^{n\times m}$ such that $UA = T$. We require U to be unimodular so that the nonzero rows of T will not just be contained in but will also span the lattice generated by rows of A (denoted by $\mathcal{L}(A)$). In particular, the HNF H of A provides a unique triangular basis for $\mathcal{L}(A)$. Fact 3 tells us that for any two unimodular triangularizations of A, say T_1 and T_2, we must have $(T_1)_{i,i} \simeq (T_2)_{i,i}$ for $1 \leq i \leq m$; that is, the respective diagonal entries are associates. The HNF H gains its uniqueness by enforcing that diagonal entries be monic and that off diagonal entries have degree strictly less than the diagonal entry in the same column.

For some applications, the degree condition on the off-diagonal entries need not hold and it may be cheaper to compute a generic triangularization rather than the unique HNF. For example, algorithm `SquareSmithForm` of §5.3 worked by finding a construction for a unimodular triangularization T of a pre-conditioned input matrix $A' \in \mathbf{F}[x]^{n\times n}$ and verifying that certain divisibility conditions held. Finding explicitly the off-diagonal entries of HNF of A' would be prohibitively expensive in this case. In general, the problem HERMITEFORM seems to be more difficult than triangularization. Hafner and McCurley show in [16] how to apply fast matrix multiplication techniques to the problem of finding a unimodular triangularization T of an input matrix A over a principal ideal domain $\mathbf{R}$. When $\mathbf{R}$ is a Euclidean domain such as $\mathbb{Z}$, the HNF of A can be found easily from T by reducing off diagonal entries in each column but they are unable to accomplish this within a constant factor of the the number of bit operations required to find T to begin with.

Our first result is a fast method for triangularizing matrices over $\mathbf{F}[x]$; this is similar to the Hafner and McCurley result for triangularizing integer matrices in that we do not produce the unique HNF. For their algorithm they used fast matrix multiplication techniques to obtain a good result in terms of the number of ring operations. In our case, we are working over the ring $\mathbf{F}[x]$ and give our results in terms of the number of field operations over $\mathbf{F}$.

Algorithm: `Triangularize`
Input: A square nonsingular matrix $A \in \mathbf{F}[x]^{n\times n}$.
Output: An upper triangular matrix $T \in \mathbf{F}[x]^{n\times n}$ and a unimodular matrix $U \in \mathbf{F}[x]^{n\times n}$ such that $UA = T$.

(1) [Randomize:]
Let $d-1$ bound the degrees of entries of A.
$U_1, U_2 \leftarrow$ unit upper triangular matrices in $\mathbf{F}^{n\times n}$ with off diagonal entries chosen at random from a subset of $\mathbf{F}$ with $\lceil 4m^3 d/\epsilon \rceil$ elements;
$B \leftarrow U_1 A$;

$C \leftarrow U_2 A$;

(2) [Triangularize:]
$U_B \leftarrow \text{FFGE}(B, n)$;
$U_C \leftarrow \text{FFGE}(C, n)$;
$T_B \leftarrow U_B B$;
$T_C \leftarrow U_C C$;

(3) [Find probable value g_i^* for $h^*(A, i)$, $1 \le i \le n$:]
for $i = 1$ to n do
  $g_i^* \leftarrow \gcd((T_B)_{i,i}, (T_C)_{i,i})$;

(4) [Check that $g_i^* = h^*(A, i)$ for $1 \le i \le n$.]
for $i = 2$ to n do
  for $j = 1$ to $i - 1$ do
    if g_{i-1}^* does not divide $(U_B)_{i,j}$ or $(U_C)_{i,j}$ then FAIL;
If U and T not required explicitly then output U_B, U_C, T_B, T_C and $[g_1^*, g_2^*, \ldots, g_n^*]$ and terminate otherwise continue.

(5) [Solve extended Euclidean problems:]
$(b_n, c_n) \leftarrow (1, 0)$;
for $i = 1$ to $n - 1$ do
  $(b_i, c_i) \leftarrow$ a solution to: $b_i(U_B)_{i,i} + c_i(U_C)_{i,i} = g_i^*$;

(6) [Construct U and T:]
$D_B \leftarrow \text{diag}(b_1, b_2, \ldots, b_n) \in \mathbf{F}[x]^{n \times n}$;
$D_C \leftarrow \text{diag}(c_1, c_2, \ldots, c_n) \in \mathbf{F}[x]^{n \times n}$;
$G \leftarrow \text{diag}(g_0^*, g_1^*, \ldots, g_{n-1}^*) \in \mathbf{F}[x]^{n \times n}$;
$U \leftarrow G^{-1}(D_B U_B + D_C U_C)$;
$T \leftarrow G^{-1}(D_B T_B + D_C T_C)$;

(7) [Output:] U and T.

Remark: In step (1), matrix U_2 can be chosen to be a permutation matrix in $\mathbf{F}^{n \times n}$ such that PA has nonsingular principal minors.

Matrices U_1 and P have entries constant polynomials whence matrices B and C will have degrees bounded by $d - 1$. This leads to a bound on the degrees of entries in U_B, U_C, T_B and T_C of nd. Finding U_B and T_B will require $O(n^4 d^2)$ field operations using an evaluation/interpolation scheme. Similarly, the matrix T_C can be found together with P and U_P using ordinary fraction-free Gaussian elimination (with row pivoting) in $O(n^4 d^2)$ field operations. The extended Euclidean algorithm will need to be used n times in step (5) to obtain suitable polynomials $b_1, b_2, \ldots, b_n$

and $c_1, c_2, \ldots, c_n$ that are bounded in degree by nd; the cost of this is $O(n \cdot (nd)^2)$ field operations. Finally, since matrices D_B, D_C and G are diagonal, the matrix multiplications in step (6) will require at most $O(n^4d^2)$ field operations, and yield matrices U and T with entries bounded in degree by $2nd$.

It is important to note that the above complexity analysis assumes standard polynomial arithmetic: $P(d) = d^2$. The main cost of the algorithm is computing the triangularizations T_B and T_C. In our case, these triangularizations are actually over the quotient field $\mathbf{F}(x)$ and can be computed using a homomorphic imaging scheme in $O(n^4d^2)$ field operations. Other algorithms that compute unimodular triangularizations depend upon solving diophantine equations over the polynomial domain (either explicitly or implicitly) *during* the triangularization phase (i.e. algorithm `ModTriangularize` of §3.2). (In particular, information about the degrees of intermediate polynomial entries in the matrix being reduced is required; this is not readily available when the polynomial is represented as list of residues.) As such, these methods are not suscesptable to a evaluation/interpolation homomorphism scheme and — assuming standard polynomial and matrix arithmetic — require $O(n^3 \cdot P(nd)) = O(n^5d^2)$ field operations. Of course, we could use fast polynomial multiplication to achieve a complexity of $O^{\sim}(n^4d)$ field operations — an improvement of about $O(d)$ — but this is an asymptotic bound. In practice, d would have to be inordinately large before a speedup was achieved.

Theorem 10 *Algorithm* `Triangularize` *is correct and fails with probability less than ϵ. Given a nonsingular input matrix $A \in \mathbf{F}[x]^{n\times n}$ with degrees of entries bounded by $d-1$, a unimodular matrix U and an upper triangular matrix T in $\mathbf{F}[x]^{n\times n}$ such that $UA = T$ can be found in an expected number of $O(n^4d^2)$ field operations using standard polyomial multiplication.*

Proof: First we show correctness. Let g_i^* for $1 \le i \le n$ be as found in step (3). We show that the algorithm does not return FAIL if and only if $g_i^* \simeq h^*(A, i)$ for $1 \le i \le n$.

If: Assume that $g_i^* \simeq h^*(A, i)$ for $1 \le i \le n$. U_1 unimodular implies B has the same determinantal divisors as A. Similarly, C will have the same determinantal divisors as A. By construction of U_B and U_C in step (2) entries on and before the diagonal in row i of U_B and U_C are associates of $(i-1) \times (i-1)$ minors of B and C respectively. Since, by assumption, g_i^* is the gcd of all $i \times i$ minors of A (and hence of B and C), step (3) will not return FAIL.

Only if: Assume that the algorithm does not return FAIL. Then, an application of Lemma 9 with $U^{(1)} = U_BU_1$, $U^{(2)} = U_CP$ and $g_0^* = 1$ will give the desired result. Note that the four conditions of Lemma 9 are verified: construction of U_B and U_C

verifies conditions 1 and 2; $(T_B)_{n,n} = (T_C)_{n,n} = \det(A)$ verifies 3; and success of step (4) verifies condition 4. Lastly, observe that the matrix U and T constructed in step (6) verify $UA = T$ and that the definition of U matches the construction of the matrix with the same name in Lemma 9.

It remains to show that the probability of failure is less than ϵ. The proof for this is analogous to that found in the proof of Theorem 9 for algorithm `SquareSmithForm` and follows from Lemmas 12 and 13. ■

6.2 Homomorphisms for HermiteForm over $\mathbf{Q}[x]$

For this section we assume $\mathbf{F} = \mathbf{Q}$ and that we have a nonsingular input matrix $A \in \mathbb{Z}[x]^{n\times n}$. Let U denote the unique unimodular matrix such that $UA = H$, the HNF of A over $\mathbf{Q}[x]$. For a prime p, let U_p denote the unimoduar matrix over $\mathbb{Z}_p$ (the finite field with p elements) such that $U_p(A \bmod p) = H_p$, the HNF of $A \bmod p$ over $\mathbb{Z}_p[x]$. Note that while for many primes p we will have $H_p = H \bmod p$, there may exist primes such that $H_p \neq H \bmod p$. We say two images H_{p_1} and H_{p_2} are "consistent" if the degrees of the corresponding diagonal entries in the HNFs H_{p_1} and H_{p_2} are identical. An algorithm to find U and H is now easily described.

(1) Find a number β such that:

  (a) Numerators and denominators of rational coefficients in U and H are integers bounded in magnitude by β;

  (b) There exists a positive integer $\alpha \leq \beta$ such that $H_p = H \bmod p$ provided that p does not divide α.

(2) Choose a list of primes $\{p_1, p_2, \ldots, p_q\}$ such that $\prod_{1\leq i\leq q} p_i \geq \beta(\beta^2+1)$. Find (U_p, H_p) for $p = p_1, p_1, \ldots, p_q$. Choose a subset of primes $\{p_{i_1}, p_{i_2}, \ldots, p_{i_k}\}$ with consistent images and such that $\prod_{1\leq j\leq k} p_{i_j} = \Pi \geq \beta^2 + 1$.

(3) Chinese remainder these images together to obtain the matrices $U' = U \bmod \Pi$ and $H' = H \bmod \Pi$.

(4) Perform a rational reconstruction on each of the integer coefficients of polynomial entries of U' and H' to obtain the corresponding matrices U and H over $\mathbf{Q}[x]$.

In step (1), the bound β is chosen to be the number with the same name given in Theorem 6, namely $\beta = (n(d+1)||A||)(n\sqrt{d}||A||)^{n^2d}$. That condition (a) is satisfied follows directly from Theorem 6. Theorem 6 also gives the existence of a number $\alpha \le \beta$ such that $\alpha U \in \mathbb{Z}[z]$. Hence, we have the identity $(\alpha U)A = (\alpha H)$, where αU, A and αH are over $\mathbb{Z}[x]$. For any prime p not dividing α — the leading coefficient of all diagonal entries in (αH) — we must have $U_p = \alpha^{-1}(\alpha U \bmod p) \bmod p$ and $H_p = \alpha^{-1}(\alpha H \bmod p) \bmod p$ (this follows from the uniqueness of the HNF). This shows that condition (b) holds. Note that an actual implementation would choose the candidate for β given by the construction in Fact 1.

Now consider step (2). An application of Lemma 8 shows that we choose all our primes to be l bits in length where $l = \lceil 6 + \log\log\sqrt{\beta(\beta^2+1)}\rceil = \Theta(\log n + \log d + \log\log ||A||)$. In particular, there are at least $q = 2\lceil\lceil\log(2\sqrt{\beta(\beta^2+1)})\rceil/(l-1)\rceil = \Theta((\log\beta)/l)$ primes between 2^{l-1} and 2^l; the product of these is greater than $\beta(\beta^2+1)$. Since the product of all primes for which $H_p \ne (H \bmod p)$ is bounded by β, there must exist a subset $\{p_{i_1}, p_{i_2}, \ldots, p_{i_k}\}$ of primes with the desired properties.

The image solutions $(U_{p_i}, H_{p_i})_{i=1,\ldots,q}$ are found as follows: (a) find $(A_{p_i})_{i=1,\ldots q}$ at a cost of $O(n^2 dM(lq)\log q)$ bit operations (cf. [1, Theorem 8.9]); (b) find U_{p_i} and H_{p_i} for $i = 1, \ldots, q$ using the HermiteForm algorithm of Illiopoulos at a cost of $O(q \cdot n^2(n + \log nd)P(nd)M(l))$ bit operations (cf. [21, Algorithm 4.2,Proposition 4.3]). (Note that for some primes p we may have $A_p = A \bmod p$ be a singular matrix. Such a prime is accounted for among the bad primes and can be discarded. The Iliopoulos HermiteForm algorithm can be trivially modified to return FAIL in this case.) In step (3), the Chinese remainder algorithm will need to be applied with q primes of length l bits to construct at most $n^2(nd+1)+n(nd+1) = O(n^3 d)$ integers, each bounded in magnitude by β. The cost of this is $O(n^3 dM(lq)\log q)$ bit operations (cf. [1, Theorem 8.5]). In step (4), rational reconstruction will need to be applied to $O(n^3 d)$ integers bounded in magnitude by $\Pi = O(\beta)$ to obtain a corresponding list of rational numbers with numerators and denominators bounded by β. The cost of this is $O(n^3 dM(\log\beta)\log\log\beta)$ bit operations. Combining these results and assuming fast polynomial and integer arithmetic gives the following.

Theorem 11 *Let A be a nonsingular input in $\mathbb{Z}[x]^{n\times n}$ with degrees of entries bounded by d and largest magnitude of integer coefficients bounded by $||A||$. The sequential deterministic algorithm described above computes the Hermite normal form H of A (over $\mathbf{Q}[x]$) and the unimodular matrix U such that $UA = H$ in $O^{\sim}(n^6 d^2 \log ||A||)$ or $O^{\sim}(s^9)$ bit operations.*

A fundamental characteristic of the HermiteFormWithMultipliers problem for matrices over $\mathbf{Q}[x]$ is that the resulting matrices U and H are disproportionately large compared to the size of the input matrix A. In Corollary 3 we bounded the size of U by $O^{\sim}(n^5 d^2 \log ||A||)$ or $O^{\sim}(s^8)$; the size of A is only $O(n^2 d \log ||A||)$ or $O(s^4)$. Furthermore, we hypothesized that this bound for Size(U) was tight. In light of this, we hypothesize that the complexity result of Theorem 11 — which is only $O^\sim(s)$ larger than our bound for Size(U) — is optimal (up to log terms) unless a method is devised for applying fast matrix multiplication techniques to the problemm of HermiteForm over $\mathbf{F}[x]$.

The best previous method for HermiteForm over $\mathbf{Q}[x]$ is the linear systems method of Labhalla, Lombardi and Marlin (cf. §4.3.3). In particular, a worst case size bound for the total size in bits of the triangular form of matrix A_{lin} — the coefficient matrix of the linear systems method for HermiteForm given in §4.3.3 — is $O^\sim(n^6 d^3 \log ||A||) = O^\sim(s^{10})$ bits. Actually triangularizing the order $O(n^2 d)$ matrix A_{lin} would be prohibitively expensive.

6.3 A Preconditioning for Rectangular Input

Algorithm `Triangularize` of §6.1 and the method for HermiteForm given in §6.2 were presented only for square nonsingular input. In practice, matrices that arises during computations are typically rectangular — sometimes very rectangular. In particular, matrices with dimension $n^2 \times n$ arise during some algebraic integration problems, and in §1 we gave an example of computing matrix polynomial gcd's by solving HermiteForm for a rectangular input matrix. We have already seen how the rectangular case can be reduced to the square nonsingular case by permuting the rows of A and augmenting with an identity matrix (cf. §3 pp. 19); in practice, this method can be extremely wasteful. Consider the 12×3 matrix

$$A = \begin{bmatrix} x^3+x^2-2x-2 & x^2+2x+1 & -x-1 \\ -35x^3-34x^2+73x+52 & -35x^2-71x-32 & 37x+29 \\ x^3+x^2-2x-2 & x^2+2x+1 & -x-1 \\ -85x^3-84x^2+173x+152 & -85x^2-171x-82 & 87x+79 \\ x^3+x^2-2x-2 & x^2+2x+1 & -x-1 \\ 97x^3+150x^2-50x-1103 & 96x^2+146x+250 & x^2+x-400 \\ x^3+x^2-2x-2 & x^2+2x+1 & -x-1 \\ 45x^3+46x^2-87x-108 & 45x^2+89x+48 & -43x-51 \\ -8x^3-98x^2-269x+1681 & -9x^2+79x-284 & x^2-180x+563 \\ 99x^3+41x^2-387x+891 & 98x^2+261x-81 & x^2-223x+264 \\ -55x^3-89x^2-7x+767 & -56x^2-71x-163 & x^2-21x+274 \\ -5x^3-4x^2+13x-8 & -5x^2-11x-2 & 7x-1 \end{bmatrix}$$

Since $\mathrm{rank}(A) = r = 3$, a basis for $\mathcal{L}(A)$ will contain only 3 rows (ie. a matrix with 9 entries). Applying the preconditioning discussed above requires finding the HNF of a 12×12 nonsingular input matrix (a matrix with 144 entries); this is much too expensive in light of the cost of HermiteForm over $\mathbf{Q}[x]$. Instead, algorithm `Reduce` that follows finds a matrix A^* of dimension 4×3 that has similar size entries as A and such that $\mathcal{L}(A^*) = \mathcal{L}(A)$. The HNF of A can then be found by finding the HNF of a 4×4 matrix. For example, a trial run of an algorithm described by `Reduce` produced

$$A^* = \begin{bmatrix} 177x^3-57x^2-1176x+4218 & 169x^2+628x-573 & 8x^2-709x+1347 \\ 335x^3+38x^2-1681x+5036 & 327x^2+1007x-604 & 8x^2-993x+1567 \\ 211x^3-164x^2-1712x+6823 & 200x^2+852x-980 & 11x^2-1049x+2204 \\ 173x^3+29x^2-898x+2606 & 165x^2+530x-307 & 8x^2-525x+811 \end{bmatrix}.$$

We draw attention to the fact that the matrix A^* produced will be of dimension $(m+1) \times m$ as opposed to $m \times m$. Allowing an extra row in the output matrix A^* allows us to keep the size of rational number coefficients small. For example, the size of rational number coefficients appearing in a basis for $\mathcal{L}(A)$ — a matrix left equivalent to A but with only m nonzero rows (e.g. the HNF of A) — will typically be much larger than the rational number coefficients appearing in the input matrix A.

Let A be an input matrix in $\mathbf{Q}[x]^{n\times m}$ with $n > m+1$. An invariant of $\mathcal{L}(A)$ is the quantity $h^*(A, m)$, defined to be the gcd of the determinant of all $m \times m$ minors of A. The algorithm `Reduce` that follows works by preconditioning the input matrix A with a certain random unimodular matrix U_R. With high probability, the gcd of the determinant of the of the two $m \times m$ minors of $U_R A$ comprised of rows $[1, 2, \ldots, m]$ (i.e. the principal m-th minor) and the rows $[1, 2, \ldots, m-1, m+1]$ will be equal to $h^*(A, m)$. This fact is sufficient to gaurantee that Hermite$(A) =$ Hermite(A^*) where A^* is the matrix comprised of the first $m+1$ rows of $U_R A$.

Algorithm: `Reduce`
Input: A matrix $A \in \mathbf{F}[x]^{n \times m}$ with full column rank and $n > m + 1$. An upper bound $0 < \epsilon < 1$ on the probability of failing.
Output: A matrix $A^* \in \mathbf{F}[x]^{n \times m}$ with all zero entries in the last $m-n+1$ rows and such that Hermite$(A^*) =$ Hermite(A). Optionally, a unimodular transformation matrix $U \in \mathbf{F}[x]^{n \times n}$ such that $H^* = UH$.

(1) [Randomize:]
Let d bound the degrees of entries of A and let C be a subset of $\mathbf{F} \setminus \{1\}$ with $\lceil 2m^2 d/\epsilon \rceil$ elements.
$U_1 \leftarrow$ a strictly upper triangular matrix in $\mathbf{F}^{m \times m}$ with entries chosen at random from C;
$U_2 \leftarrow$ a matrix in $\mathbf{F}^{m \times (n-m)}$ with entries chosen at random from C;
$\vec{\alpha} \leftarrow$ a row vector in $\mathbf{F}^{1 \times (n-m)}$ with entries chosen at random from C except for $\vec{\alpha}_1 = 0$;
$\vec{\gamma} \leftarrow$ a row vector in $\mathbf{F}^{1 \times (n-m)}$ with entries chosen at random from C except for $\vec{\gamma}_2 = 0$;

$$U_R \leftarrow \left[\begin{array}{c|c} U_1 & U_2 \\ \hline \vec{0} & \vec{\alpha} \\ \hline \vec{0} & \vec{\gamma} \\ \hline O & O \end{array}\right] + I_n.$$

$B \leftarrow [U_R A | \vec{e}] \in \mathbf{F}^{n \times (m+1)}$ where $\vec{e}$ is an $n \times 1$ column vector with all entries 0 excepts for the m-th entry, which is 1.

(2) [Triangularize:]
$V \leftarrow \mathrm{FFGE}(B, m)$;
$W \leftarrow \mathrm{FFGE}(B, m+1)$;
Note: $V = \begin{bmatrix} U_T & 0 \\ V' & d_1 I \end{bmatrix}$ and $W = \begin{bmatrix} U_T & 0 \\ W' & d_2 I \end{bmatrix}$ where $U_T \in \mathbf{F}[x]^{(m+1)\times(m+1)}$ is upper triangular, $V', W' \in \mathbf{F}[x]^{(n-m+1)\times(m+1)}$ and $d_1, d_2 \in \mathbf{F}[x]$. With probability less than ϵ, both d_1 and d_2 may be zero — in this case the algorithm returns FAIL.

(3) [Find probable value for $h^*(A,m)$:]
$g^*_m \leftarrow \gcd(d_1, d_2)$;

(4) [Check that $g^*_m = h^*(A,m)$.]
if g^*_m does not divide all entries of V' and W' then FAIL;

(5) [Construct A^*:]
$A^* \leftarrow \begin{bmatrix} U_1 & U_2 \\ 0 & 0 \end{bmatrix} A$;
If U not required then output A^* and terminate otherwise continue.

(6) [Solve extended Euclidean problem:]
$(a,b) \leftarrow$ a solution to: $ad_1 + bd_2 = g^*_m$;

(7) [Construct unimodular multiplier:]
$U \leftarrow \begin{bmatrix} I_{m+1} & 0 \\ \frac{a}{g^*_m}V' + \frac{b}{g^*_m}W' & I_{n-m+1} \end{bmatrix} U_R$;

(8) [Output:] U and A^*.

Remarks:
(1) In step (1), choosing C to be a subset of $\mathbf{F} \setminus \{1\}$ instead of $\mathbf{F}$ ensures that the randomizing premultiplier matrix U_R is unimodular (ie. nonsingular).
(2) In step (2), FFGE$(B, m+1)$ can be found at the same time as FFGE(B,m) by continuing fraction free Gaussian elimination for one more column.

The proof of correctness for algorithm `Reduce` is nearly identical to that of algorithm `Triangularize` of section §6.1 and can be sketched briefly as follows. If step (4) does not fail then the algorithm goes on to to construct a matrix U and A^* such that $UA = A^*$. Clearly, U is unimodular by construction whence Hermite(A^*) = Hermite(A). By construction, entries of V' and W' are associates of determinants $m \times m$ minors of A', whence $g^*_m = h^*(A',m)$ $(= h^*(A,m))$ if and only if the divisibility checks in step (4) are satisfied. This shows that an incorrect result will never be returned. The challenge lies in proving that algorithm `Reduce` is a correct Las Vegas algorithm. In particular, we desire that in step (3) that the identity $g^*_m = h^*(A',m)$ holds with high probability so that repetition of the algorithm will almost never be necessary. The following lemma assures us that g^*_m will be correct provided that the entries in U_R do not form a root of a certain polynomial bounded in degree by $2m^2d$; by a result of Schwartz [32], the probability of this happening is $2m^2d/\#C$ (ie. less than ϵ).

Lemma 14 *Let A be a matrix in $\mathbf{F}[x]^{n\times m}$, $n > m+1$, of rank m and with the degrees of entries bounded by d. Then there is a polynomial π in $(2n(m+1) - m(m+3))/2$ variables such that if*

(1) U_R in $\mathbf{F}^{(m+1)\times n}$ has the form

$$U_R = \left[\begin{array}{c|c} U_1 & U_2 \\ \hline \vec{0} & \vec{\alpha} \\ \hline \vec{0} & \vec{\gamma} \end{array}\right]$$

where $U_2 \in \mathbf{F}^{m\times(n-m)}$, U_1 is unit upper triangular in $\mathbf{F}^{m\times m}$, and $\vec{\alpha}$ and $\vec{\gamma}$ are row vectors in $\mathbf{F}^{1\times(n-m)}$ with $\vec{\alpha} = [1, \alpha_2, \alpha_3, \ldots, \alpha_{n-m}]$ and $\vec{\gamma} = [\gamma_1, 1, \gamma_3, \ldots, \gamma_{n-m}]$;

(2) d_1 is the determinant of the principal m-th minor of U_RA;

(3) d_2 is the determinant of the $m\times m$ minor formed by rows $[1, 2, \ldots, m-1, m+1]$ of U_RA.

then $\gcd(d_1, d_2) = h^*(A, m)$, unless the $(2n(m+1) - m(m_3))/2$ entries in U_2, $\vec{\alpha}$, $\vec{\gamma}$ and above the diagonal in U_1 form a root of π. The degree of π is no more than $2m^2d$.

Proof: First consider the case when U_R contains indeterminant entries. In particular, let the entry in the i-th row k-th column of $[U_1|U_2]$ be $\rho_{i,k}$ where $\bar{\rho} = (\rho_{i,k})_{1\le i\le m, i<k\le n}$ is a vector of indeterminants and let $\bar{\alpha} = (\alpha_2, \alpha_2, \ldots, \alpha_{n-m})$ and $\bar{\gamma} = (\gamma_1, \gamma_3, \ldots, \gamma_{n-m})$. By a result of Kaltofen, Krishnamoorthy and Saunders [24, Lemma 3.6] we must have $d_1 = h^*(A, m)p_1$, where $p_1 \in \mathbf{F}[x, \bar{\rho}, \bar{\alpha}]$ either is an irreducible polynomial in $\mathbf{F}[\bar{\rho}, \bar{\alpha}, x] \setminus \mathbf{F}[x]$ or is 1. Similarly, we must have $d_2 = h^*(A, m)p_2$, where $p_2 \in \mathbf{F}[x, \bar{\rho}, \bar{\gamma}]$ either is an irreducible polynomial in $\mathbf{F}[\bar{\rho}, \bar{\gamma}, x] \setminus \mathbf{F}[x]$ or is 1. Hence, we must have $\gcd(d_1, d_2) = h^*(A, m)$ if p_1 is not an associate of p_2; to show this it will be sufficient to demonstrate that either p_1 depends on $\bar{\alpha}$ or p_2 depends on $\bar{\gamma}$. Let A_s be the submatrix comprised of the last $n-m+1$ rows of A and let $C_{i,j}$ denote the cofactor of the entry in the i-th row j-th column of the principal m-th minor of U_RA. Then, we can express d_1 and d_2 according to their m-th row cofactor expansion (cf. Fact 4) as

$$\begin{bmatrix} d_1 \\ d_2 \end{bmatrix} = \begin{bmatrix} 1 & \alpha_2 & \alpha_3 & \cdots & \alpha_{n-m} \\ \gamma_1 & 1 & \gamma_3 & \cdots & \gamma_{n-m} \end{bmatrix} A_s \begin{bmatrix} C_{m,1} \\ C_{m,2} \\ \vdots \\ C_{m,m} \end{bmatrix} \tag{6.1}$$

$$= \begin{bmatrix} 1 & \alpha_2 & \alpha_3 & \cdots & \alpha_{n-m} \\ \gamma_1 & 1 & \gamma_3 & \cdots & \gamma_{n-m} \end{bmatrix} \begin{bmatrix} q_1 \\ q_2 \\ q_3 \\ \vdots \\ q_{n-m} \end{bmatrix} \tag{6.2}$$

Now, the $C_{m,*}$ in (6.1) will be independant of $(\bar{\alpha}, \bar{\gamma})$ since they are associates of determinant of minors of the first $m-1$ rows of $U_R A$. In particular, the polynomials q_* in (6.2) will depend only on $(\bar{\rho}, x)$ and not on $(\bar{\alpha}, \bar{\gamma})$. Since d_1 and d_2 are nonzero (A has rank m), their must exists a smallest integer i, $1 \leq i \leq n-m$ such that q_i is nonzero. If $i = 1$, then d_2 depends on γ_1; if $i = 2$, then d_1 depends on α_2; if $3 \leq i \leq n-m$ then d_1 depends on α_i and d_2 depends on γ_i. This shows that $\gcd(d_1, d_2) = h^*(A, m)$ as required. An application of Lemma 11 yields the existence of a $2md \times 2md$ determinant Δ, whose entries are coefficients of x of d_1 and d_2, such that for any evaluation $(\bar{\rho}, \bar{\alpha}, \bar{\gamma}) \to (\hat{\rho}, \hat{\alpha}, \hat{\gamma})$ where $(\hat{\rho}, \hat{\alpha}, \hat{\gamma})$ is a corresponding list of field elements that are not a root of Δ, $\gcd(d_1, d_2) = h^*(A, i)$. It remains to establish a degree bound for Δ. Coefficients of x of $U_R A$ are of degree 1 whence coefficients of x of d_1 and d_2 will have total degrees bounded by m. This leads to a bound on the total degree of Δ of $2m^2 d$. Finally, set $\pi = \Delta$ to complete the proof. ■

Theorem 12 *Algorithm* `Reduce` *is correct and requires repitition with probability less than* ϵ.

6.4 A General Algorithm for SmithForm over $\mathbf{F}[x]$

The algorithm we give here depends on the result of the previous section. In particular, for an input matrix $A \in \mathbf{Q}[x]^{n \times m}$ with rank $r < n-1$, algorithm `Reduce` should first be used to construct a matrix A^* with at most $r+1$ nonzero rows. Algorithm `SmithForm` can then be used to compute the determinental divisors of A^*, which will be those of A.

Algorithm 3.1: `SmithForm`
Input: A matrix $A \in \mathbf{F}[x]^{n \times m}$ with rank $r = n$ or $r = n-1$ and a constant $0 < \epsilon < 1$ bounding the probability of failure.
Output: $[s_1^*, s_2^*, \ldots, s_r^*]$ such that s_i^* is an associate of the i-th determinantal divisors of A.

(1) [Randomize:]
Let d bound the degrees of entries of A and let C be a subset of size $\lceil 5n^4 d/\epsilon \rceil$ of $\mathbf{F}$.
$U_R \leftarrow$ a unit upper triangular matrix with off diagonal elements chosen at random from C;
$V_R \leftarrow$ a unit lower triangular matrix with off diagonal elements chosen at random from C;
$A' \leftarrow U_R A V_R$;

(2) [Triangularizee:]
$U_T \leftarrow \mathrm{FFGE}(A', n)$;
$T \leftarrow U_T A'$;

(3) [Find probable values for $h^*(A_1', i), \ldots, h^*(A_1', r)$ where $A_1' \in \mathbf{F}[x]^{n\times r}$ is comprised of the first r columns of A'.]
if $r = n-1$ then $d^* \leftarrow \gcd((U_T)_{n,1}, (U_T)_{n,2}, \ldots, (U_T)_{n,n})$
else $d^* \leftarrow T_{r,r}$;
if $d^* = 0$ then return FAIL;
for $i = 1$ to r do
  $g_i^* \leftarrow$ an associate of $\gcd((d^*)^2, T_{i,i})$;

(4) [Assay that g_i^* for $1 \le i \le r$ is an associate of $h^*(A_1', i)$:]
for $i = 2$ to n do
  for $j = 1$ to i do
    if g_{i-1}^* does not divide $(U_T)_{i,j}$ then FAIL;

(5) [Verify condition of Lemma 10:]
for $i = 1$ to $r-1$ do
  for $j = i+1$ to m do
    if g_i^* does not divide $T_{i,j}$ then FAIL;

(6) [Find probable values for $s^*(A,i)$'s and verify divisibility properties:]
$s_0^* \leftarrow 1$;
for $i = 1$ to $r-1$ do
  $s_i^* \leftarrow g_i^*$;
$s_r^* \leftarrow$ an associate of $\gcd(g_r^*, T_{r,r+1}, T_{r,r+2}, \ldots, T_{r,m})$;
for $i = 1$ to $r-1$ do
  if s_i^{*2} does not divide $s_{i-1}^* s_{i+1}^*$ then FAIL;

(7) [Output:] $[s_1^*, s_2^*, \ldots, s_r^*]$;

Proof: First we show correctness. Let g_i^* for $1 \le i \le r$ be as found in step 3. We show that g_i^* is an associate of $s^*(A,i)$ over $\mathbf{Q}[x]$ ($g_i^* \simeq s^*(A,i)$) for $1 \le i \le r-1$ if and only if the algorithm does not return FAIL.

Only if: Assume that $g_i^* \simeq s^*(A, i)$ for $1 \le i \le r-1$. By Fact 2, step (4) will not abort. By construction of U_T in step (2), $U_{Ti,j}$ for $1 \le j \le i \le n$ is an associate of an $(i-1) \times (i-1)$ minor of A', which by construction has the same determinantal divisors as A. Similarly, $T_{i,j}$ for $1 \le i \le j \le n$ is an $i \times i$ minor of A'. By assumption, g_i^* is an associate of the gcd of all $i \times i$ minors of A (and hence of A') and neither steps (5) nor (6) will abort.

If: Assume that the algorithm does not abort. The success of step (4) together with Lemma 9 will provide the existence of a unimodular matrix $U \in \mathbf{F}[x]^{n \times n}$ such that $S_T = UA'$ is upper triangular with $S_{Ti,i} = g_i^*/g_{i-1}^*$ for $1 \le i \le r$. To see this, let $[A_1'|A_2'] = A'$ with $A_1' \in \mathbf{F}[x]^{n \times r}$ and $A_2' \in \mathbf{F}[x]^{n \times m-n}$. If $r = n$ then set $B = A_1'$. If $r = n-1$ there exists a column vector $e \in \mathbf{F}[x]^{n \times 1}$ such that $d^* = (U_T)_{n,1}e_1 + (U_T)_{n,2}e_2 + \cdots + (U_T)_{n,n}e_n$; set $B = [A_1'|e]$ in this case. Then $B \in \mathbf{F}[x]^{n \times n}$ with $\det(B) = d^*$. Recall that $h^*(B', i)$ is the gcd of all $i \times i$ minors of B. In particular d^* as defined in step (3) will be an associate of $h^*(B, r)$ (and also $h^*(B, n)$) for both the case $r = n$ and $r = n-1$. Set $U^{(1)} = d^* B^{\mathrm{adj}}$. Then $T^{(1)} = U^{(1)}B = (d^*)^2 I_n$ is upper triangular. Set $U^{(2)} = U_T$. Then $T^{(2)} = U^{(2)}B$ is upper triangular. An application of Lemma 9 (with $g_n^* = d^*$ if $r = n-1$) shows that $g_i^* = h^*(B, i)$ for $1 \le i \le n$. Furthermore, Lemma 9 provides a construction for a matrix U such that such that UB is upper triangular with i-th diagonal entry g_i^*/g_{i-1}^* for $1 \le i \le r$. Since $\det(UB) = \det(B)$ we have U unimodular. Now consider the matrix $S_T = UA' = U[A_1'|A_2']$. The first r columns of S_T will be those of the upper triangular matrix UB, since $B = [A_1'|e]$, whence if $r = n-1$ then row n of S_T must have all zero entries. We claim that the i-th diagonal entry of S_T, namely g_i^*/g_{i-1}^*, divides each off-diagonal entry in row i of S_T for $1 \le i \le r-1$. By the construction of U in Lemma 9, the entry in the i-th row j-th column of S_T will be $1/g_{i-1}^*(a_i R_{i,j} + b_i T_{i,j})$ where $R = U^{(1)}A'$. By construction, each row or $U^{(1)}$ is divisible by d^* whence $R_{i,j}$ is divisible by g_i^*. Similarly, since step (5) was successful, $T_{i,j}$ is divisible by g_i^*. It follows that g_i^*/g_{i-1}^* divides $1/g_{i-1}^*(a_i R_{i,j} + b_i T_{i,j})$. We may now apply Lemma 10 with matrix S_T to obtain a unimodular matrix V such that $S_T V$ is diagonal with i-th diagonal entry g_i^*/g_{i-1}^* for $1 \le i \le r-1$ and

$$
\begin{aligned}
g_{r-1}^*(S_T V)_{r,r} &\simeq \gcd(g_r^*, (S_T)_{r,r+1}, (S_T)_{r,r+2}, \ldots, (S_T)_{r,m}) \\
&\simeq \gcd(g_r^*, a_r R_{r,r+1} + b_r T_{r,r+1}, \ldots, a_r R_{r,m} + b_r T_{r,m}) \\
&\simeq \gcd(g_r^*, b_r T_{r,r+1}, \ldots, b_r T_{r,m})
\end{aligned} \tag{6.3}
$$

So far, we have shown the existence of a unimodular U and V such that $UA'V = S_T V$ is a diagonal matrix with i-th diagonal entry an associate g_i^*/g_{i-1}^* for $1 \le i \le r-1$ and r-th diagonal entry given by

(6.3). This implies $s^*(A', r) = s^*(S_T V, r) = (S_T V)_{r,r} \prod_{1 \le i \le r-1} g_i^*/g_{i-1}^* = \gcd(g_r^*, b_r T_{r,r+1}, \ldots, b_r T_{r,m}) = \gcd(g_r^*, T_{r,r+1}, \ldots, b_r T_{r,m})$ where the last equality holds because $s^*(S_T V, r)$ divides all $r \times r$ minors of A'. Finally, the success of step 6, together with Fact 2 and the uniqueness of the SNF, ensures that $S_T V$ must be the SNF of A whence $s_i^* = s^*(A, i)$ for $1 \le i \le r$.

It remains to show that the probability of failure is less than ϵ. The proof for this is analogous to that found in the proof of Theorem 9 for algorithm `SquareSmithForm` and follows from Lemmas 12 and 13. ∎

Chapter 7

Conclusions

This thesis has considered the problem of computing Hermite and Smith normal forms of matrices over two fundamental domains: the ring of integers $\mathbb{Z}$ and the ring $\mathbf{F}[x]$ of univariate polynomials with coefficients from a field. The main challenge in devising efficient algorithms for computing these forms is to control the growth of intermediate expressions. We have presented the classical algorithm for Hermite and Smith normal form — a variation of Gaussian elimination with the extended Euclidean algorithm replacing division — and demonstrated the bad behaviour of the classical technique with respect to intermediate expression swell for both the case of matrices over the integers and matrices over polynomial domains. In the case of matrices over the integers we have discussed a new class of normal form algorithms (that have appeared recently in the literature) that perform all intermediate computations modulo the determinant of the input matrix. The bulk of this thesis, however, has been devoted to presenting new algorithms that we have developed for computing Hermite and Smith normal forms of matrices over $\mathbf{F}[x]$, $\mathbf{F}$ a field. Our emphasis has been the special case $\mathbf{F} = \mathbf{Q}$ — this case poses a double challenge since we must control not only the degrees of intermediate polynomials but also the size of the rational number coefficients.

For the following summaries, let A be an $n \times m$ matrix over $\mathbf{F}[x]$, $\mathbf{F}$ a field, with rank r and with degrees of entries bounded by d. We give our results in terms of the input matrix parameters n, m, and d. When $\mathbf{F} = \mathbf{Q}$, we use $||A||$ to denotes the largest integer coefficient appearing in A. To summarize complexity results, we use the parameter $s = n + m + d + \log ||A||$ as a measure of the size of an input matrix over $\mathbb{Z}[x]$.

Hermite Normal Form

Let $\mathbf{F} = \mathbf{Q}$ and let the input matrix A have $n = m = r$ and have all coefficients integers (i.e. $A \in \mathbb{Z}[x]^{n\times n}$ is nonsingular). Let H denote Hermite normal form of A (over $\mathbf{Q}[x]$) and let U denote unique unimodular matrix such that $UA = H$. We have given a sequential deterministic algorithm that finds the matrices U and H (over $\mathbf{Q}[x]$) in $O\tilde{~}(n^6 d^2 \log ||A||)$ or $O\tilde{~}(s^9)$ bit operations. This complexity result is asymptotic and assumes both fast polynomial and integer arithmetic. In practice, the dominant cost of the algorithm will be $O(n^3 d)$ applications of the Chinese remainder algorithm to construct integers representable in $O(n^2 d \log ||A||)$ bits. We have established that the size in bits required to represent the matrices U and H is $O\tilde{~}(n^5 d^2 \log ||A||)$ or $O\tilde{~}(s^8)$ and have hypothesized that this bound is tight in the worst case. In light of this, we hypothesize that the complexity result of $O\tilde{~}(s^9)$ for computing U and H is optimal (up to log terms) unless methods for applying fast matrix multiplication techniques to the problem of computing Hermite normal forms over $\mathbf{F}[x]$, $\mathbf{F}$ a finite field, are discovered.

Smith Normal Form

Let A be an input matrix over $\mathbf{F}[x]$, $\mathbf{F}$ a field, with $n = m = r$. We have presented a Las Vegas probabilistic algorithm that finds the matrix S, the Smith normal form of A, in an expected number of $O\tilde{~}(n^4 d)$ field operations. This complexity result, however, assumes fast polynomial multiplication and will not be interesting in most practical cases. An important feature of the techniques used in this algorithm is that they are susceptable to a simple homomorphic imaging scheme. In particular, we can achieve a complexity of $O\tilde{~}(n^4 d^2)$ using standard polynomial multiplication.

Next, let $\mathbf{F} = \mathbf{Q}$ and let the input matrix A have all coefficients integers (i.e. $A \in \mathbb{Z}[x]^{n\times n}$). We have presented a Las Vegas probabilistic algorithm that finds the matrix S, the Smith normal form of A over $\mathbf{Q}[x]$, in an expected number of $O\tilde{~}(n^3 d(d + n^2 \log ||A||)) = O\tilde{~}(s^7)$ bit operations. This complexity result assumes fast integer multiplication. In practice, the main cost of the algorithm is triangularizing via fraction-free Gaussian elimination a matrix with similar size entries as A. Again, we are able to emply a homomorphic imaging scheme to accomplish this triangulation step in $O\tilde{~}(s^8)$ bit operations using standard integer and polynomial arithmetic.

In practice, matrices for which the Hermite and Smith normal form are desired are often nonsquare and/or singular. In light of this, we have presented generalizations of our Hermite and Smith algorithms that work for rectangular input. The results we have given above for computing Hermite and Smith normal forms depend on properties of square nonsingular matrices and do not generalize readily to the rectangular case. To handle the rectangular case, we have devised a Las Vegas probabilistic algorithm that takes as input a matrix $A \in \mathbf{Q}[x]^{n \times m}$ with $r = m < n + 1$ and with entry degrees bounded by d and returns a matrix $A^* \in \mathbf{Q}[x]^{(m+1) \times m}$ such that Hermite$(A) =$ Hermite(A^*). Furthermore, the matrix A^* will have entries bounded in degree d and have rational number coefficients only slightly larger than those appearing in A. The cost of finding A^* is that of triangularizing A via fraction-free Gaussian elimination. This reduction method for rectangular lattices should prove useful for other problems as well — we mention some possible applications below.

The computation of Hermite and Smith normal forms is a fundamental operation in a computer algebra systems — the fast algorithms we have presented should prove useful in this regard. In particular, the methods we have employed in our algorithms have two important features: (1) they control the growth of intermediate and/or final expressions very well; and (2) they are susceptible to homomorphic imaging schemes and thus have practical implementations. Our focus in this thesis has been on the actual computation of Hermite and Smith normal form over polynomial domains. A natural topic for further study is to consider applications of these constructions within the context of a computer algebra system. For example, there are a number of applications that follow immediately from the algorithms we have presented: (1) A fast Las Vegas algorithm — $O^{\sim}(n^5 \log ||A||)$ expected bit operations — for computing the Frobenius normal form of an integer input matrix $A \in \mathbb{Z}^{n \times n}$; and (2) A fast method for finding the smallest degree polynomial d such that the system of polynomial diophantine equations $A\vec{x} = d\vec{b}$ admits a solution. Less immediately, the algorithm we have given for reducing rectangular lattices over $\mathbf{Q}[x]$ to the nearly square case suggests a fast method for computing matrix polynomial gcds. From a theoretical perspective, we hope to devise a sequential deterministic version of our fast Las Vegas Smith form algorithm for matrices over $\mathbf{Q}[x]$.

Bibliography

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

[2] E. H. Bareiss. Sylvester's identity and multistep integer-preserving Gaussian elimination. *Mathematics of Computation*, 22(103):565–578, 1968.

[3] E. H. Bareiss. Computational solution of matrix problems over an integral domain. *Phil. Trans. Roy. Soc. London*, 10:68–104, 1972.

[4] 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.

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

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

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

[8] J. Cannon and G. Havas. Algorithms for groups. *Australian Computer Journal*, 24(2):51–58, May 1992.

[9] B. W. Char, K. O. Geddes, and G. H. Gonnet. GCDHEU: Heuristic polynomial GCD algorithm based on integer GCD computations. *Journal of Symbolic Computation*, 7(1):31–48, Jan. 1989.

[10] 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.

[11] 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, Feb. 1987.

[12] M. A. Frumkin. An application of modular arithmetic to the constuction of algorithms for solving systems of linear equations. *Soviet Math. Dokl.*, 17:1165–1168, 1976.

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

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

[15] 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, 1994.

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

[17] B. Hartley and T. O. Hawkes. *Rings, Modules, and Linear Algebra.* Chapman and Hall, 1970.

[18] G. Havas, D. F. Holt, and S. Rees. Recognizing badly presented Z-modules. *Linear Algebra and its Applications*, 192:137–163, 1993.

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

[20] 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, Aug. 1989.

[21] 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, Aug. 1989.

[22] T. Kailath. *Linear Systems.* Prentice Hall, Englewood Cliffs, N.J., 1980.

[23] 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.

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

[25] R. Kannan. Polynomial-time algorithms for solving systems of linear equations over polynomials. *Theoretical Computer Science*, 39:69–88, 1985.

[26] 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, Nov. 1979.

[27] S. Labhalla, H. Lombardi, and R. Marlin. Algorithmes de calcul de la réduction d'Hermite d'une matrice à coefficients polynomiaux. In *Comptes-Rendus de MEGA92, Nice, France*. Birkhauser, 1992.

[28] M. Marcus and H. Minc. *A Survey of Matrix Theory and Matrix Inequalities*. Prindle, Weber & Schmidt, Incorporated, 53 State Street, Boston, Massachusetts, 1964.

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

[30] V. Ramachandran. Exact recuction of a polynomial matrix to the Smith normal form. *IEEE Transactions on Automatic Control*, AC-24(4):638–641, Aug. 1979.

[31] A. Schönhage. Probabilistic computation of integer polynomial GCD's. *Journal of Algorithms*, 9:365–371, 1988.

[32] J. T. Schwartz. Fast probabilistic algorithms for verification of polynomial identities. *J. ACM*, 27:701–717, 1980.

[33] H. J. S. Smith. On systems of linear indeterminate equations and congruences. *Phil. Trans. Roy. Soc. London*, 151:293–326, 1861.

[34] B. Trager. *Integration of Algebraic Functions*. PhD thesis, Dept. of EECS, M.I.T., 1984.

[35] 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.