Theoretical Computer Science 39 (1985) 69–88
North-Holland

SOLVING SYSTEMS OF LINEAR EQUATIONS OVER POLYNOMIALS

R. KANNAN*

Department of Computer Science, Carnegie-Mellon University, Pittsburgh, PA 15213, U.S.A.

Abstract. We consider a system of linear equations of the form $A(x)X(x)=b(x)$, where $A(x)$, $b(x)$ are given $m\times n$ and $m\times 1$ matrices with entries from $\mathbb{Q}[x]$, the ring of polynomials in the variable x over the rationals. We provide a polynomial-time algorithm to find the general solution of this system over $\mathbb{Q}[x]$. This is accomplished by devising a polynomial-time algorithm to find the triangular canonical form (Hermite form) of the matrix $A(x)$ using unimodular column operations. As applications we are able to give polynomial-time algorithms for finding the diagonal (Smith canonical) form of a polynomial matrix, testing whether two given matrices of rational entries are similar and for finding the invariant factors of a matrix of rationals.

1. Introduction

Whereas in most cases the main issue in proving time bounds on an algorithm is proving bounds on the number of arithmetic operations (additions, subtractions, multiplications and divisions), here the main issue is to prove that the number of bits needed to represent all numbers is suitably bounded. In view of this, this section introduces both the particular problem we are dealing with as well as the issue of the size of numbers.

The canonical forms we deal with are from the following two theorems of Hermite and Smith.

Theorem 1.1 (Hermite [12]). *If A is an $m\times n$ matrix of integers (polynomials) with independent rows, then there exists an $n\times n$ matrix K of integers (polynomials) with determinant ± 1 (a nonzero rational) such that AK is lower triangular, i.e.*

$$(AK)_{ij}=0 \quad \text{for } j>i.$$

We do not prove the theorem here, but the algorithm provided in this paper is a constructive proof of the theorem.

Definition 1.2. A matrix of integers (polynomials) is said to be *unimodular* if its determinant is ± 1 (a nonzero rational). It is easily seen that a matrix is unimodular if and only if it has an inverse over the ring being considered ($\mathbb{Z}$ or $\mathbb{Q}[x]$).

* Supported by NSF Grants MCS-841690 and MCS-8304770.

0304-3975/85/$3.30 © 1985, Elsevier Science Publishers B.V. (North-Holland)

Let us confine attention now only to polynomials. Postmultiplying a matrix A by a unimodular matrix corresponds to performing a sequence of the following column operations on A (see, for example, Newman [25] for a derivation of this well-known fact):

(1) multiplying a column of A by a nonzero rational,

(2) adding a polynomial times some column into another.

While the multiplier K and thus AK of Hermite's theorem are not unique they can be made so by stipulating an additional condition on AK. Clearly adding any (polynomial) multiple of column i of AK to any earlier column of AK still keeps it lower triangular. Further we may add a suitable multiple of column i to an earlier column j such that the degree of $(AK)_{ij}$ is strictly less than that of $(AK)_{ii}$. This degree constraint imposed on all the entries of AK makes the lower triangular form unique and we call this the *Hermite normal form* of A. More precisely, for any $m \times n$ matrix A of polynomials with rank m, there exists a unique lower triangular $m \times n$ matrix HNF(A) with the following two properties:

(1) there exists a unimodular matrix K such that $\mathrm{HNF}(A) = AK$,

(2) the degree of any off-diagonal entry of HNF(A) is less than that of the diagonal entry in its row.

If both unimodular row and column operations are allowed, then one can diagonalise the matrix.

Theorem 1.3 (Smith [28]). *If $A(x)$ is an $n \times n$ matrix of polynomials, there exist unimodular matrices $U(x)$ and $K(x)$ such that UAK is diagonal such that $(UAK)_{ii} \neq 0$ for $i = 1, 2, \ldots, r$ and $(UAK)_{ii} = 0$ for $i > r$ for some r and $(UAK)_{ii}$ is a factor of $(UAK)_{i+1,i+1}$ for $i < r$. Further, the diagonal form satisfying these conditions is unique.*

We call this unique form the *Smith normal form* (SNF) of A.

Several algorithms are known for finding these canonical forms and thus for solving the system of polynomial equations. (For example, see [1, 24, 2, 3, 13].) These algorithms find a variety of applications—for example linear sequential machines [11], the computation of Jacobians [16], computation of generating functions of flow graphs [23] and linear system theory [18, 13, 17]. None of the algorithms is known to be polynomial time bounded because we do not seem to be able to prove good bounds on the coefficient sizes and degrees of intermediate polynomials obtained. Indeed in practice it has been observed that coefficient sizes do grow a lot. See for example [4, 27, 8] for a discussion of this phenomenon described as 'intermediate expression swell' or 'coefficient growth'. Frumkin [10] and Kannan [19] contain general discussions of how this phenomenon may take place. To make this more rigorous we need some definitions.

Basic definitions and notation

The *size* of an integer is the number of bits needed to express it in binary representation, thus the *size* of an integer N is at most $1 + \lceil \log_2 N \rceil$, the 1 for the

sign of the integer. The *size* of a rational number is the sum of the sizes of its numerator, its denominator and 1, the latter for a punctuation mark separating the two. We have to define some analogous measures for polynomials, vectors and matrices. By a polynomial we always mean one with rational coefficients, i.e. an element of $\mathbb{Q}[x]$, the ring of polynomials with rational coefficients. The *length* of a vector v denoted $|v|$ is the normal Euclidean length of the vector. The *length* of a polynomial $f(x)=\sum_{i=0}^{d} f_i x^i$ over $\mathbb{Q}[x]$ denoted $|f|$ is defined as the length of the vector $(f_d, f_{d-1}, \ldots, f_0)$. If $X(x)$ is a n vector of polynomials of lengths $l_1, l_2, \ldots, l_n$ then the *length of* $X(x)$ is defined as the length of the vector $(l_1, l_2, \ldots, l_n)$. The *degree* of $X(x)$ is the maximum of the degrees of its components. An $m \times n$ matrix of rationals (or polynomials) is viewed as a mn vector of all its entries for the purpose of defining its length. For a matrix A of rationals (polynomials) the *denominator* of A denoted den(A) is the least common multiple of the denominators of all entries (all coefficients of all entries) of A. Clearly the denominator and the length of a matrix together bound the number of bits needed to represent the matrix, namely if matrix A has length L and denominator D, then each entry of A can be expressed as the ratio of two integers the numerator being at most L and the denominator at most D and thus can be represented in at most $\log_2 L + \log_2 D$ bits. The *size* of a matrix is the total number of bits needed to represent it and thus the size of a matrix is at most the total number of entries times $(\log(L)+\log(D))$. The scenario we deal with is the following: our algorithms will take as input any matrix and transform it through a sequence of operations to arrive finally at a canonical form. We show that there is a fixed polynomial $p(\cdot)$ such that if the number of bits needed to express the input matrix is s, then at any stage of the algorithm, the current matrix can be expressed with at most $p(s)$ bits. (This then leads to a proof that the algorithm is polynomial time bounded.) To establish this it clearly suffices to show that

> *there is a fixed polynomial $q(\cdot)$ such that on any input matrix A expressed in s bits, the length and denominator of any intermediate matrix produced is at most $2^{q(s)}$.*

Thus all our bounds are on the length and denominator. Some other pieces of notation:

For two polynomials $f(x)$ and $g(x)$, $\lfloor f(x)/g(x) \rfloor$ denotes the *remainder* when $f(x)$ is divided by $g(x)$, i.e. the unique polynomial of degree less than that of $g(x)$ which is congruent to $f(x)$ modulo $g(x)$.

For any square matrix S, det(S) denotes the determinant of S.

For any matrix S, S_j denotes the vector consisting of the jth column of S; S_{ij} or $S_{i,j}$ denotes the i,jth entry of the matrix S. The ith *principal minor* of S is the submatrix of S consisting of the first i rows and the first i columns.

We denote polynomials as $a(x)$, $b(x)$, etc., but when the meaning is clear, we just write a, b omitting the argument x. The ring of polynomials in x with integer coefficients is denoted $\mathbb{Z}[x]$ and the ring of polynomials with rational coefficients

is denoted $\mathbb{Q}[x]$. The *content* of a polynomial $a(x)$ in $\mathbb{Z}[x]$ is the greatest common divisor (abbreviated gcd) of its coefficients. It is said to be *primitive* if its content is 1.

With these definitions in place we return to a discussion of 'coefficient growth'. The simplest example where this occurs is in the Euclidean gcd (greatest common divisor) algorithm. Suppose $a(x)$ and $b(x)$ are two polynomials whose gcd is to be found. Assume without loss of generality that $\deg(a(x))$ is at least $\deg(b(x))$ (else swap the two). If $b(x)$ is 0, clearly $a(x)$ is the greatest common divisor of the two polynomials. Otherwise we find

$$b'(x) = a(x) + \lfloor a(x)/b(x) \rfloor \cdot b(x) \quad \text{and} \quad a'(x) = b(x)$$

and reiterate with the two new polynomials a' and b' as a and b. (See [22] for a detailed discussion of this topic.) To prove that this algorithm is polynomial time bounded, we need first to bound the sizes of all polynomials we obtain during the algorithm. Unfortunately, it is easily seen that naive arguments can at best hope to prove that

$$\operatorname{den}(b') \leq \operatorname{den}(b)\operatorname{den}(a).$$

Since the number of iterations can be as many as n the minimum degree of the input polynomials, these *sequential* bounds (i.e. bounds on the parameters of one iteration based on those of the previous one) can only hope to prove that the length and denominator of any intermediate polynomial is at most those of the input polynomials *raised to* 2^n since these quantities could square each iteration. (*Note*: I have been purposely vague in the above discussion because it is only meant to show the infeasibility of an approach.) While these arguments only indicate that we cannot rule out the possibility of enormous coefficient growth by theoretical means it is also the case that such growth was observed in practice in this and several related problems (see [22, pp. 414] for a discussion of the exponential growth in the number of bits of intermediate polynomials).

Collins [6] made a pioneering contribution towards solving this problem by showing that with slight care, one may make the algorithm provably polynomial time bounded. His elegant method of proof was to avoid sequential bounds but argue directly a bound on the intermediate polynomials in terms of the original input polynomials. This was achieved by showing that each coefficient of each intermediate polynomial was a subdeterminant of the so called resultant matrix of the initial two polynomials and then invoking a lemma similar to Proposition 2.2 to follow.

In a different context, Kannan and Bachem [20] gave polynomial time algorithms for finding the Smith and Hermite normal forms of an integer matrix and hence solving linear diophantine equations. Again for these algorithms, instead of sequential bounds, the paper showed that at any stage, the entries of the current matrix are related to subdeterminants of the original matrix and hence satisfy the required bounds. The present paper is an extension of this result to matrices of polynomials.

Chou and Collins [8] improved the analysis and running time of the algorithms for finding the Smith and Hermite normal forms of a matrix of integers.

While my work with Bachem was carried out in late 1977 and 1978, Von zur Gathen and Sieveking [30] had *earlier* obtained polynomial time algorithms for solving systems of linear diophantine equations without getting the Hermite or Smith reduced forms. Unfortunately, their work only appeared in a Lecture Notes and does not seem to have been widely known until much later.

2. Technical results

Proposition 2.1. *Suppose E and e are $m \times n$ and $m \times 1$ matrices of integers respectively with $\max(|E|, |e|) = a$. If the system of m equations $Ey = e$ has a solution y over the rationals then it has a solution y with the length and denominator of y each at most $(na)^n$.*

Proof. Without loss of generality we may assume that the rows of the matrix E are independent (if not, either the system is inconsistent whence the proposition is trivially true or we may discard all but a spanning set of rows). If there is a solution to the system then there is one of the form $y = B^{-1}e$ where B is a $m \times m$ submatrix of E. $\det(B)B^{-1}$ is a matrix of integers and hence $\text{den}(y) \leq \det(B)$. Now $\det(B)$ is the sum of m factorial terms each the product of m entries of B. Thus each term is of absolute value at most a^m and $\det(B)$ is of absolute value at most $(a^m)m!$ which is at most $(am)^m$ at most $(an)^n$. Similarly each entry of B^{-1} is a cofactor of B divided by determinant of B, the latter an integer and thus is at most $(na)^{n-1}$ in magnitude. Thus each component of $B^{-1}e$ is at most $(na)^{n-1}|e|$ in magnitude. Hence $|B^{-1}e| \leq (na)^n$. □

Proposition 2.2. *If A is an $n \times n$ matrix of polynomials each of degree at most d, with $|A|$ equal to a, then each subdeterminant of A is a polynomial of degree at most nd and length at most $(2nad)^n$.*

Proof. The first part follows since each subdeterminant of A is the sum of at most $n!$ terms each of degree at most nd. To derive the bound on the length, the length of each of these at most $n!$ terms is bounded by the length of the polynomial $(a + ax + ax^2 + \cdots + ax^d)^n$ because each coefficient of each $A_{ij}(x)$ is at most a in absolute value. Denote the latter polynomial by $z(x)$. Clearly, $z(x)$ has all positive coefficients and thus $|z(x)|$ is at most the sum of its coefficients which equals $z(1)$. Now, $z(1)$ is $a^n(d+1)^n$. Thus the length of the subdeterminant polynomial is at most $(n!)a^n(d+1)^n$ and the proposition is proved. □

Lemma 2.3 (Gauss). *Suppose polynomials $f(x)$, $g(x)$ and $h(x)$ in $\mathbb{Z}[x]$ satisfy $f(x) = g(x)h(x)$. Then* $\text{content}(f) = \text{content}(g)\text{content}(h)$.

Proof. See for example, [29]. □

Lemma 2.4 (Collins quoted in [22, pp. 391]). *If $f(x)$ and $g(x)$ are two polynomials with integer coefficients such that g divides f, then $|g(x)| \leq |f(x)|d^{2d}$ where d is the degree of $f(x)$.*

Proof. Suppose $f(x) = f_d x^d + f_{d-1}x^{d-1} + \cdots + f_1 x + f_0$ and $g(x) = g_d x^d + g_{d-1}x^{d-1} + \cdots + g_0$. (Some of the leading coefficients of $g(x)$ may be zero.) Let f be the column vector $(f_d, f_{d-1}, \ldots, f_0)^t$, f' be the column vector $(f(d), f(d-1), \ldots, f(0))^t$, g be the column vector $(g_d, g_{d-1}, \ldots, g_0)^t$ and g' the column vector $(g(d), g(d-1), \ldots, g(0))^t$. Finally, let D be the $(d+1)\times(d+1)$ van der Monde matrix defined by $D_{ij} = (d-i+1)^{d-j+1}$. Then clearly, $g' = Dg$ hence $g = D^{-1}g'$ hence $|g| \leq |g'|d^{d/2}$. The last inequality follows from the fact that each entry of D^{-1} is at most $d^{(d/2)-2}$ in absolute value (see, for example, [21, 1.2.3. Exercise 40]). Since $g(x)$ divides $f(x)$, $g(s)$ must divide $f(s)$ for each integer s, thus each component of g' divides the corresponding component of f'. Hence, $|g'| \leq |f'|$. Further each component of f' is at most $|f|d^d$ and hence $|f'| \leq |f|d^{d+2}$ proving the lemma.[1] □

Lemma 2.5. *Suppose A is an $m \times n$ matrix and b an $m \times 1$ matrix both of polynomials with integer coefficients with* $\max(\deg A, \deg b) = d$ *and* $(\max |A|, |b|) = a$. *If the system of polynomial equations $A(x)X(x) = b(x)$ has a solution $X(x)$ belonging to $\mathbb{Q}[x]$, then it has one with* $\deg(X(x))$ *at most* nd; $|X(x)|$ *and* $\operatorname{den}(X(x))$ *at most* $(2nda)^{(4n^2d)}$ *each.*

Proof. As in Proposition 2.1 we may assume without loss of generality that the rows of A are independent. (We mean over $\mathbb{Q}[x]$. Independence thus means that no nontrivial polynomial combination of the rows of A equals zero.) Rearrange columns of A if necessary so that the first m columns are independent and call this nonsingular $m \times m$ matrix B, let N denote the rest $(n-m)$ columns of A. We denote by $\beta(x)$ the determinant of B. Partitioning the vector $X(x)$ of unknowns accordingly into $X_1(x)$ and $X_2(x)$, we may write the given equations as

$$B(x)X_1(x) + N(x)X_2(x) = b(x).$$

Suppose $(X_1(x), X_2(x))$ is a solution. Divide each component of $X_2(x)$ by $\beta(x)$ and call the vector of remainders $X_2'(x)$, i.e. let $X_2'(x)$ be the unique vector with each component a polynomial of degree less than that of $\beta(x)$ such that there exists a vector $v(x)$ of polynomials with $X_2(x) = X_2'(x) + \beta(x)v(x)$. Then,

$$\begin{aligned} b(x) &= BX_1(x) + NX_2(x) \\ &= B(X_1(x) + B^{-1}N\beta v) + NX_2'(x) \\ &= BX_1'(x) + NX_2'(x) \quad \text{(say)}. \end{aligned}$$

[1] More precisely, $|f'| \leq |f|d^d(d+1)$. But $d+1$ is at most d^2 for $1 \leq d$. Thus we get the required bound. We will constantly use such generous bounds so as to simplify them.

Since $\beta(x)B^{-1}$ is a matrix of polynomials (not rational functions), (X_1', X_2') is another polynomial solution to our system.

$$\deg(X_2'(x)) \leqslant \deg(\beta(x)) \leqslant nd$$

by Proposition 2.2 (noting that m is at most n). Now

$$X_1' = B^{-1}(b(x) - NX_2'(x)).$$

Thus,

$$\beta(x)X_1'(x) = \beta(x)B^{-1}(b(x) - NX_2'(x)).$$

Then

$$\begin{aligned}\deg(\beta(x)) + \deg(X_1'(x)) &= \deg(\beta(x)X_1'(x)) \\ &\leqslant \deg(\beta(x)B^{-1}(x)) + \max(d, d + \deg(\beta(x)))\end{aligned}$$

since $\deg(X_2'(x)) \leqslant \deg(\beta(x))$

$$\leqslant (n-1)d + d + \deg(\beta(x))$$

since each entry of $\beta(x)B^{-1}$ is a cofactor of B.

Hence, $\deg(X_1'(x)) \leqslant nd$.

To prove a bound on the coefficient sizes in a solution, we consider each component of $X(x)$ as being a ndth or lower degree polynomial and represent it by $nd+1 = u$ (say) variables which are allowed to assume rational values. Then it is clear that by equating coefficients of corresponding powers of x of both sides of the polynomial system $A(x)X(x) = b(x)$ we can convert it to an equivalent system of equations over the rationals. An important note: the degree bounds derived earlier are crucial for the above statement. Instead of introducing cumbersome notation to prove this rather obvious statement formally, we content ourselves with giving Example 2.6.

Example 2.6. Suppose

$$A = \begin{pmatrix} x+1 & x^2+x-1 \\ x-1 & x^2+1 \end{pmatrix},$$

$$b(x) = \begin{pmatrix} 1 \\ x-1 \end{pmatrix}.$$

Then d is 2; a is $\max\{\sqrt{9}, \sqrt{3}\} = 3$; $n = 2$. So if there is a solution to the system defined by A and b, then there is one with $\deg(X(x))$ at most 4. So let

$$X(x) = \left(\sum_{i=0}^{4} g_i x^i, \sum_{i=0}^{4} h_i x^i \right)^t.$$

The highest power of x obtaining in the product $A(x)X(x)$ is 6, thus we get a total of $7 \cdot 2 = 14$ equations over the rationals. We illustrate by listing some of them:

Equating coefficients of x^6 on both sides of $A(x)X(x) = b(x)$ yields the following two equations:

$$h_4 \cdot 1 = 0, \qquad h_4 \cdot 1 = 0.$$

Equating coefficients of x^5 on both sides gives us

$$g_5 + g_4 + h_3 + h_4 - h_5 = 0, \qquad -g_5 + g_4 + h_3 + h_5 = 0.$$

Going back to the general case it is clear that the system over the rationals has $n(nd+1)$ (which is at most $2n^2d$) variables and that its coefficients are all integers of absolute value at most a and further that the length of the right hand side vector is exactly $|b(x)|$ which is at most a. Thus applying Proposition 2.1, this system has a solution of length and denominator at most $(2n^2da)^{(2n^2d)}$ which is at most $(2nda)^{(4n^2d)}$. This completes the proof of the lemma. □

Lemma 2.7. *Suppose $a(x)$, $b(x)$ are two elements of $\mathbb{Z}[x]$ with $\max\{|a|, |b|\}$ equal to l and maximum of the degree of a and b equal to d. Then there exist three polynomials p, q, r in $\mathbb{Z}[x]$ such that r is the greatest common divisor of a and b and $pa + qb = r$. Further $|r|, |p|, |q| \leq (ld)^{2d}$.*

Proof. The proof follows by Collins's [6] subresultant algorithm. □

Lemma 2.8. *Suppose $A(x)$ is an $n \times n$ matrix of degree d with entries from $\mathbb{Q}[x]$. The determinant of $A(x)$ can be found in polynomial time by evaluating* $\det(A(0))$, $\det(A(1))$, $\det(A(2)), \ldots, \det(A(nd))$ *and interpolating.*

Proof. By Proposition 2.2, $\det(A(x))$ is a polynomial of degree at most $n \cdot d$. Thus it can be found by interpolation from values at the $nd+1$ arguments. We have to argue the time bound for the computation. Evaluation of determinants of matrices of rationals can be done in polynomial time [9]. This is accomplished by proving that Gaussian elimination can be done in polynomial time. Let us consider interpolation. Suppose $D(x)$ is an Nth or lower degree polynomial whose coefficients are to be found given $D(0), D(1), \ldots, D(N)$. Arguing as in Lemma 2.4, if D is a vector of the $N+1$ coefficients of $D(x)$ and D' the vector of the $N+1$ values and M the suitable Van der Monde matrix, we have

$$D' = MD,$$

and D is the unique solution to this system. Since the paper of Edmonds [9] shows that Gaussian elimination can be done in polynomial time, a slight modification can be used to show that systems of equations can be solved in polynomial time proving the current lemma. □

Lemma 2.9. *There is a polynomial time algorithm that for any square matrix $A(x)$*

determines whether A is nonsingular and if so finds a permutation of the columns of A such that each principal minor is nonsingular.

Proof. The proof is identical to the proof of a similar lemma for matrices of rationals found in [20]. The first row must have a nonzero element. Thus after permuting columns if necessary, we can assume that A_{11} is nonzero. Assume for induction that the columns of A have been rearranged so that the first i principal minors of A are nonsingular. Let A' be the matrix consisting of the first $i+1$ rows of A and all columns of A. Since the $i \times i$ principal minor of A is nonsingular, the first i columns of A' are independent. Either one of the $n-i$ subdeterminants of A' formed by taking the first i columns of it plus one of the remaining columns is nonzero whence we can rearrange columns to make the first $i+1$ principal minors of A nonsingular or all these subdeterminants are zero whence A is singular. Clearly using the algorithm of the previous lemma one can determine in polynomial time which of these possibilities obtains. □

3. The algorithm for the Hermite normal form of polynomial matrices

As stated earlier, for matrices of integers, Kannan and Bachem [20] give polynomial time algorithms for finding the Hermite and Smith normal forms and the multipliers K and U; K. A suitable modification of this algorithm when applied to polynomial matrices, keeps the degrees of all intermediate polynomials obtained bounded by a polynomial in the length of the input. Intuitively, the degree of a polynomial is analogous to the size of an integer and thus this is not surprising. However the analogy does not help solve the basic problem of growth of coefficients. This is the main concern in this paper. First the algorithm for square nonsingular matrices is presented to be later extended to all matrices. In this section, we assume that *the input matrix A with entries in $\mathbb{Z}[x]$ is $n \times n$ and has rank n* and assume (cf. Lemma 2.9) that the principal minors of A are all nonsingular. In the next section we extend this to work on any rectangular matrix by a simple construction. We indicate there that the time consuming algorithm of Lemma 2.9 can be dispensed with. The reason for introducing it is simplicity of description of the main ideas of the algorithm HNF to follow. This algorithm works in $n-1$ stages—stages 1 through $n-1$ by performing column operations on the matrix A. So, the variable A denotes the *current* matrix A. In the proofs we need to refer to the matrix at different stages, so we introduce the notation $A^{(i)}$ to denote the matrix A at the completion of stage $i-1$ of the algorithm. Thus the initial input matrix is $A^{(1)}$ and the final (output) matrix is $A^{(n)}$. At the completion of the $(i-1)$st stage the matrix A denoted $A^{(i)}$ satisfies the following two conditions:

(1a) The $i \times i$ principal minor of $A^{(i)}$ is a lower triangular nonsingular matrix.

(1b) $A^{(i)}$ has been obtained from $A^{(1)}$ by unimodular column operations on the first i columns alone.

Thus there is a unimodular matrix $K^{(i)}$ such that

(2a) $A^{(i)} = A^{(1)}K^{(i)}$.

(2b) $K^{(i)}_{jl} = 0$ for $\max(j, l) \geqslant (i+1)$ and $j \neq l$.

(2c) $K^{(i)}_{jj} = 1$ for $j \geqslant i+1$.

Stage i consists of two substages. In the first, we do column operations on the first $(i+1)$ columns of A to get the $(i+1)\times(i+1)$ principal minor of A to be lower triangular. In the second substage, we solve a suitable system of equations to find a small multiplier matrix $K^{(i+1)}$ and finally we determine $A^{(i+1)}$ which is the product $AK^{(i+1)}$ and are ready to go to stage $i+1$. The two substages are described as two different subroutines.

Substage 1 of stage (i)

Comment. Takes as input an $m \times n$ polynomial matrix A whose $i \times i$ principal minor is a nonsingular lower triangular matrix and puts the $(i+1)\times(i+1)$ principal minor in lower triangular form using unimodular column operations. Reminder on notation: for any matrix R, R_j denotes the jth column of R.

for $j = 1$ **to** i **do**

(1) Calculate $r(x) = \gcd(A_{jj}(x), A_{j,(i+1)}(x))$ and polynomials $p(x)$ and $q(x)$ such that p, q, r belong to $\mathbb{Z}[x]$ and $r(x) = pA_{jj} + qA_{j,(i+1)}$ using Collins's subresultant algorithm (Lemma 2.7).

(2a) Perform unimodular column operations on A so that $A_{j,(i+1)}$ becomes zero:

$$D = \begin{pmatrix} p & -A_{j,(i+1)}/r \\ q & A_{jj}/r \end{pmatrix}.$$

Replace column j and column $i+1$ of A by the two columns of the product: $(A_jA_{i+1})D$.

(2b) Make $A_{jj}(x)$ a primitive polynomial with integer coefficients by multiplying column j of A by a suitable rational.

(2c) Make $A_{j+1,i+1}(x)$ a primitive polynomial by multiplying column $i+1$ of A by a suitable rational.

end;

end substage 1.

Before we describe the other substage, we need some facts. We use the notation set up immediately before the program.

Lemma 3.1. *Substage* 1 *of stage i on an input matrix whose* $i \times i$ *principal minor is lower triangular produces a matrix whose* $(i+1)\times(i+1)$ *principal minor is lower triangular using only unimodular column operations.*

Proof. Clearly for each j step (2) makes $A_{j,i+1}$ zero. Further, since this is done in

increasing order of j's, the above diagonal entries in column j remain at zero, thus proving the lemma. □

Lemma 3.2. *Suppose an $m \times m$ matrix S of polynomials with integer coefficients and maximum degree d has been transformed into a lower triangular matrix T by unimodular column operations and the diagonal entries of T are $d_1(x), d_2(x), \ldots, d_m(x)$ which are all primitive polynomials (with integer coefficients). Then $|d_l(x)| \leq (2m|S|d)^{3^{md}}$ for all l.*

Proof. Since T is lower triangular, $\det(T) = \prod_{l=1}^{m} d_l(x)$. Thus, by Gauss' lemma, $\det(T)$ is a primitive polynomial. Further, since T has been obtained from S by unimodular operations, $\det(T) = (p/q)\det(S)$ for some relatively prime integers p and q. Since $\det(T)$ is primitive, we must have $p = 1$ and hence $|\det(T)| \leq |\det(S)| \leq (2m|S|d)^m$, the last inequality from Proposition 2.2. Finally, the current lemma follows from Collins's lemma (Lemma 2.4) since each of the $d_l(x)$ divides $\det(T)$. □

We have so far shown that the diagonal entries of the matrix A are polynomials that can be expressed using a small number of bits at the end of each stage. We need to deal with the off-diagonal entries. Substage 2 is executed to ensure that these are small.

Lemma 3.3. *Suppose T is an $m \times m$ lower triangular matrix that is obtained from a matrix S by unimodular column operations and the diagonal entries of T are $d_1(x)$, $d_2(x), \ldots, d_m(x)$. Suppose K is any $m \times m$ matrix of polynomials such that SK is a lower triangular matrix with diagonal entries $d_1(x), d_2(x), \ldots, d_m(x)$. Then, K must be unimodular. Further there exists such a matrix K with $\deg(K)$ at most $m^2 d$, $|K|$ and $\operatorname{den}(K)$ each at most $(4md|S|)^{16m^5d^2}$.*

Proof. The first assertion follows because $\det(SK) = \det(T)$ and by the first sentence in the statement of the lemma, $\det(T)$ and $\det(S)$ are associates. Suppose that the unimodular matrix that transforms S into T is K_0. Then K_0 is a candidate for K except that it may not satisfy the size conditions. Let K_{ij}, $1 \leq i, j \leq m$, be a set of m^2 unknowns and consider the polynomial equations

$$\sum_{j=1}^{m} S_{ij}(x) K_{jl}(x) = d_l(x) \quad \text{for } l = 1, \ldots, m,$$

$$\sum_{j=1}^{m} S_{ij}(x) K_{jl}(x) = 0 \quad \text{for } 1 \leq i, l \leq m;\ l \geq i+1.$$

These equations stipulate that the product $S \cdot K$ be lower triangular with $d_1(x)$, $d_2(x), \ldots, d_m(x)$ as the diagonal entries. We wish to apply Lemma 2.5 to derive the current lemma. First note that the right-hand side vector of polynomials in our system has length at most $m \cdot \max(|d_l(x)|)$ which by the previous lemma is at most

$m(2md|S|)^{3md}$ and clearly the coefficient matrix of our system is bounded in length by this quantity too and thus this can be taken to be a of Lemma 2.5. We have m^2 unknown polynomials, thus n of Lemma 2.5 is here m^2. d remains d. Plugging these quantities into the result of Lemma 2.5 yields Lemma 3.3. (Of course, after noting that the hypothesis of Lemma 3.3 guarantees that the system of equations does have a solution.) □

The series of the last two lemmas directly suggests what substage 2 of stage i ought to be.

Substage 2 of stage (i)

Input. Original input matrix $A^{(1)}$ and the current A whose $(i+1)\times(i+1)$ principal minor is lower triangular and has been obtained from $A^{(1)}$ by unimodular column operations on the first $i+1$ columns alone.

(1) $m=(i+1)$; $S=$ the $(i+1)\times(i+1)$ principal minor of $A^{(1)}$.

(2) $d_l(x)=A_{ll}$ for $l=1, 2, \ldots, m$.

(3) Write and solve the system of equations of the last lemma to find an $(i+1)\times(i+1)$ matrix K.

(4) Enlarge K to an $n\times n$ matrix as follows: $K_{jj}=1$ for $j=i+2, i+3, \ldots, n$ and $K_{lj}=0$ for $l\neq j$ and $\max(l,j)\geq(i+2)$.

(5) $A=A^{(1)}K$ (this is now our $A^{(i+1)}$).

end substage 2.

Program HNF(A)

Input. $n\times n$ square matrix $A(x)$.

Using the algorithm outlined in Lemma 2.9, determine whether A is nonsingular; if it is not, terminate.

if it is, rearrange the columns so that all the principal minors are nonsingular;

for $i=1$ **to** $n-1$ **do**:

Call substage 1 of stage i

Call substage 2 of stage i

end;

Return A.

Lemma 3.4. *At the end of substage 2 of stage i of the program, the current A denoted* $A^{(i+1)}$ *satisfies*

$$|A^{(i+1)}|\leq(4nad)^{17n^5d^2},\quad \text{den}(A^{(i+1)})\leq(4nad)^{16n^5d^2}\quad \textit{and}\quad \deg(A^{(i+1)})\leq(n^2+2)d,$$

where d and a denote the degree and length of $A^{(1)}$ *respectively.*

Proof. As in substage 2 take m to be $i+1$ and S to be the $(i+1)\times(i+1)$ principal

minor of $A^{(1)}$. By the last lemma, the K found in step (3) of substage 2 satisfies:

$$|K|, \operatorname{den}(K) \leq (4mda)^{16n^5d^2}.$$

Now $A^{(i+1)} = A^{(1)}K$. Thus by the above inequality and the fact that $A^{(1)}$ is made up of polynomials from $\mathbb{Z}[x]$, we have the required inequality on $\operatorname{den}(A^{(i+1)})$. For the bound on the length, we make use of the following simple claim.

Claim 3.5. *If $a(x)$ and $b(x)$ are two polynomials in $\mathbb{Q}[x]$ of length α and β respectively and the degree of a is r then the length of $a(x)b(x)$ is at most $(r+1)\alpha\beta$.*

Proof. Clearly each coefficient of $a(x)$ is at most α in absolute value. Thus $|a(x)b(x)| \leq \sum_{t=0}^{r} |\alpha x^t b(x)|$. Hence, the claim follows. □

Now going back to the proof of the lemma, each entry of $A^{(1)}$ has degree at most d and length at most a and each entry of K has length at most $(4nad)^{16n^5d^2}$ thus each entry of the product has length at most $(4nad)^{17n^5d^2}$ by the last claim. Thus the length of the whole matrix A satisfies the bounds claimed. □

So far we have shown that numbers are 'small' (i.e. their number of bits is bounded by a polynomial in the length of the input) at the conclusion of each stage. What about during the execution of a stage? Clearly substage 2 does not increase the size of numbers by more than a polynomial. We now consider substage 1.

Lemma 3.6. *At all times during the execution of the algorithm* HNF *we have*

$$|A|, \operatorname{den}(A) \leq (4nad)^{400n^{11}d^4} \quad \text{and} \quad \deg(A) \leq 2n^3d,$$

where n, a, and d are the dimension, length, and degree respectively of the original input square matrix whose entries are all polynomials with integer coefficients.

Proof. As before, let $A^{(i)}$ be the matrix at the end of stage $i-1$ of the algorithm and denote the matrix at the end of execution of step 2 of substage 1 with i, j as parameters $A^{(i,j)}$. Let L, D and N denote the length, degree and denominator of $A^{(i)}$ respectively. The $(j+1)\times(j+1)$ subdeterminants D_1 and D_2 of $A^{(i)}$ and $A^{(i,j)}$ consisting of the first $j+1$ rows and first j columns and column $i+1$ must be associates since the latter matrix has been obtained from the former by unimodular operations on its columns. Call the two determinants $d_1(x)$ and $d_2(x)$. Since D_2 is lower triangular, $A^{(i,j)}_{j+1,i+1}$ divides $d_2(x)$ and since it is primitive, it divides the primitive part of $d_2(x)$ and hence divides $d_1(x)$. Noting that $d_1(x)$ belongs to $\mathbb{Z}[x]$ we have, from Lemma 2.4,

$$|A^{(i,j)}_{j+1,i+1}| \leq |d_1(x)|(Dn)^{2nD} = (2nLD)^n(nD)^{2nD} \leq (2nLD)^{3nD},$$

by Proposition 2.2, and $\deg(A^{(i,j)}_{j+1,i+1}) \leq D$.

By Lemma 2.7, $|r(x)|$, $|p(x)|$ and $|q(x)|$ in step (2) of substage 1 of stage i are bounded by

$$((2nLD)^{3nD}D)^{2D} \leq (2nLD)^{6nD^2+1},$$

and $\deg(p), \deg(q) \leq D$.

Now consider the execution of step (2) with $j+1$.

$$|A^{(i,j)}_{j+1,j+1}/r(x)| \leq LD^{2D} \quad \text{by Lemma 2.4,}$$

$$|A^{(i,j)}_{j+1,i+1}/r(x)| \leq (2nLD)^{3nD}D^{2D}.$$

Thus the execution of step (2) with i and $j+1$ as parameters multiplies the length of A_{i+1} by at most a factor of $2(2nLD)^{6nD^2+1}(D+1)$ (see Claim 3.5). Thus the execution of substage 1 of stage i (for a fixed i) multiplies the length of A by at most

$$(2nLD)^{7n^2D^2}.$$

Clearly, for every other column too, the length goes up by at most this factor. Finally, the denominator of A is increased by a factor of at most the length of $r(x)$ for each execution of step 2 and therefore reasoning as above it too goes up by at most this factor. Substituting the inequalities

$$L \leq (4nad)^{17n^5d^2}, \qquad N \leq (4nad)^{16n^5d^2}, \qquad D \leq (n^2+2)d,$$

from Lemma 3.4 we get the conclusions of the current lemma. □

A note of caution: These bounds are polynomial but astronomically large. Fortunately, experience indicates that the actual running times are usually much better for such algorithms. However it is hoped that further theoretical improvements would be made as was done for the case of integer matrices by Chou and Collins.

Theorem 3.7. *Algorithm* HNF *finds the Hermite normal form of a square nonsingular matrix over* $\mathbb{Q}[x]$ *in polynomial time.*

Proof. Lemma 3.6 shows that the number of bits is bounded by a polynomial. We note that each time substage 1 is executed, we perform at most $O(n)$ arithmetic operations (additions, subtractions and multiplications) and substage 2 is polynomial time bounded as argued in the proof of Lemma 2.8. □

3.1. Avoiding substage 2

As we hinted earlier, given any lower triangular matrix $H(x)$ with nonzero diagonal entries, we can easily perform unimodular column operations on the matrix H so that each off-diagonal entry of $H(x)$ has degree strictly less than that of the diagonal entry in its row. To see this, say H is $m \times n$. Since $H_{2,2}$ is nonzero, we may first add a suitable polynomial multiple of column 2 to the first column so that the degree condition is satisfied for the $(2, 1)$ entry of the matrix. Next we add a suitable

multiple of column 3 into each of columns 1 and 2 so that the condition is met for the $(3, 1)$ and the $(3, 2)$ entries, noting that this does not change the $(2, 1)$ entry. We also note that given H these operations are uniquely determined. We repeat this process until a suitable multiple of column m is added to each of the previous columns. Clearly this procedure is polynomial time bounded, we call it the *reduce-off-diagonal* procedure for obvious reasons. We will replace substage 2 with this procedure in the HNF algorithm. The argument that this new algorithm is polynomial time bounded is based on three facts.

Fact 3.1.1. For any input matrix $A^{(1)}$ whose principal minors are all nonsingular, there is a unique matrix $A^{(i)}$ satisfying conditions (1a) and (1b) of Section 3 and the additional condition that each off diagonal entry of the $(i \times i)$ principal minor of $A^{(i)}$ has degree less than that of the diagonal entry in its row (henceforth we refer to such a matrix as off diagonal reduced). This is a well known fact, see for example [25] for a proof.

Fact 3.1.2. Suppose A is the input matrix and $A^{(i)}$ is the unique matrix satisfying the conditions of the last fact. Then there is a 'small' matrix (i.e. matrix whose size is bounded by a polynomial in the length of the input) $K^{(i)}$ such that $AK^{(i)} = A^{(i)}$.

Proof. The proof of this fact is almost identical to that of Lemma 3.3. All we need to do is modify the system of linear equations there to stipulate that the off diagonal entries have low enough degree. The crucial observation is that this stipulation can be made by linear equations since it is equivalent to saying that certain terms are zero. Although this fact is important to conclude that the time consuming substage 2 can be dispensed with, we do not elaborate more on the proof because it is quite close to that of Lemma 3.3. □

Fact 3.1.3. Suppose we have executed substage 1 of stage i of the HNF algorithm and then *reduce-off-diagonal* to arrive at $A^{(i)}$, the unique off diagonal reduced matrix. Then by the previous two facts, the $A^{(i)}$ must be the product of $A^{(1)}$ and a 'small' $K^{(i)}$ and hence must be 'small' itself.

So we may now replace substage 2 by the *reduce-off-diagonal* procedure to get the new stage *i*. We have proved that there is a fixed polynomial say $r(\cdot)$ such that the entries at the conclusion of the stage are bounded by r (length of the original input). Now a lemma analogous to Lemma 3.6 shows that they are bounded at all times by some polynomial.

4. Extension to rectangular matrices

Suppose A is a $m \times n$ matrix of polynomials in $\mathbb{Q}[x]$. Let us continue to assume that rank of A is m. After permuting columns if necessary, we may assume that the

first m columns of A are independent. Define a $n \times n$ matrix A' as follows:

$$A'_{ij} = A_{ij} \quad \text{for } i \leqslant m,$$

$$A_{ii} = 1 \quad \text{for } i \geqslant m+1, \qquad A_{ij} = 0 \quad \text{otherwise.}$$

Thus A' is of the form

$$A' = \begin{pmatrix} A_1 & A_2 \\ 0 & I \end{pmatrix}$$

where I is the $(n-m) \times (n-m)$ identity matrix and A_1 is an $m \times m$ nonsingular matrix. Now, clearly A' is nonsingular and Algorithm HNF operating on A' yields HNF(A'). Clearly, the first m rows of this matrix form a lower triangular matrix obtained by column operations on A and is the required matrix. Thus we have shown the following lemma.

Lemma 4.1. *Given an $m \times n$ matrix $A(x)$ with rank m we can find in polynomial time a lower triangular matrix obtained from it by unimodular column operations alone.*

However before we apply this to solve systems of equations we have to deal with the problem that in general, the rows of A may not be independent. To deal with this, we devise a modified HNF algorithm. The input now is any $m \times n$ matrix $A(x)$. We again run the algorithm in stages. At the completion of stage $i-1$ we again have the matrix satisfying conditions (1a) and (1b) of Section 2. Now we execute substage 1 of stage i. If at the end of this $A_{i+1,i+1}$ is nonzero we proceed with substage 2, etc. If it is zero, but some entry say $A_{k,i+1}$ of column $i+1$ is nonzero we swap rows k and $i+1$ (note that in this case k must be greater than $i+1$), remember the swap and proceed. If the entire column A_{i+1} is zero it is dependent on the first i columns. In this case we swap this column and the first nonzero column of A that comes after it and redo stage i. We repeat this process either until $A_{i+1,i+1}$ is nonzero or until all but the first i columns of A are zero. In the first case we proceed with stage $i+1$; in the second the algorithm terminates. We present the algorithm and prove that it is polynomial time bounded. The proof is indirect and short; it shows that the algorithm as described above produces essentially the same result as program HNF of Section 3 would on a matrix consisting of A with a suitable identity matrix appended. Before giving the algorithm we note that the argument of the above paragraph has shown the following classical result.

Theorem 4.2. *If $A(x)$ is an $m \times n$ matrix of polynomials of rank r then there exits a unimodular $n \times n$ matrix $K(x)$ and a permutation matrix P such that PAK is a lower triangular matrix with all but its first r columns equal to zero.*

Definition 4.3. We call a matrix PAK with the properties stated in the above theorem a *unimodularly column reduced form of A*, denoted UCR(A).

Program UCR(A)
Input. An $m \times n$ matrix $A(x)$.
  if all entries of A are zero **then** return A
  permute the rows and columns of A so that A_{11} is nonzero
  $j = 1$. *Comment*: j denotes the number of columns of A dealt with so far.
  for $i = 1$ **to** $n - 1$ **do**
    FLAG $= 0$. *Comment*: FLAG $= 1$ implies we may go to the next stage.
    while $j < n$ and FLAG $= 0$ **do**
      swap columns $i+1$ and $j+1$ of A
      $j = j + 1$
      call substage 1 of stage i
      $B =$ the matrix of the first j columns of $A^{(1)}$
      $C =$ the matrix of the first j columns of the current A
      Write and find a solution $K(x)$ to the system of equations $B(x)K(x) = C(x)$
      as in substage 2
      $A = AK$
      if some entry, say $A_{k,i+1}$, of column $i+1$ is nonzero **then do**
        Swap rows $i+1$ and k of A and $A^{(1)}$
        FLAG $= 1$
      end
    end
  if FLAG $= 0$ *Comment*: Then $j = n$ and all but the first i columns of A are zero.
  then return A
  end
  Return A
end UCR.

Theorem 4.4. *The above algorithm correctly finds a unimodular column reduced form of the input matrix A in polynomial time.*

Proof. At the beginning of stage i, the matrix A has the following form: The matrix consisting of the first j columns of it form a lower triangular matrix call it S (where j is some integer greater than or equal to i). If T is the matrix of the first j columns of the original input $A^{(1)}$ then there is a permutation matrix P and a unimodular matrix U such that $S = PTU$. Let V denote the matrix consisting of the first i rows of PT. Let W be the $j \times j$ matrix defined as follows: The first i rows of W are V, the bottom right $(j-i) \times (j-i)$ submatrix of W is the identity and the rest of the entries of W are zero. The reader can convince himself/herself that HNF(W) has in its first i rows the matrix S and indeed if the HNF algorithm is run on W, it executes the steps of our algorithm UCR(S) (plus some more pertaining to the last $j-i$ rows) and thus the proof that HNF is a polynomial time bounded algorithm tells us that the present one is too. □

We use the unimodularly reduced form to find a general solution to a system of equations.

Theorem 4.5. *Suppose a matrix $A(x)$ has the unimodularly column reduced form H which equals AK, K a unimodular matrix over $\mathbb{Q}[x]$. Suppose further that H is of the form*

$$H = \begin{pmatrix} B & 0 \\ D & 0 \end{pmatrix},$$

where B is an $r \times r$ nonsingular matrix. Then the system of equations $A(x)X(x) = b(x)$ (where we refer to the r vector of the first r entries of $b(x)$ as $b_1(x)$ and the vector of the other $m-r$ entries as $b_2(x)$) has a solution iff each entry of $B^{-1}b_1(x)$ belongs to $\mathbb{Q}[x]$ and $DB^{-1}b_1(x) = b_2(x)$.

Further if the system has a solution, then the set of all solutions is given by $\{K_1B^{-1}b_1 + K_2X_2 : X_2$ is any $n-r$ vector with entries in $\mathbb{Q}[x]\}$, where K_1 and K_2 are matrices consisting of the first r and the last $n-r$ columns of K respectively.

Proof. We may rewrite the system of equations as $A(x)K(x)[K^{-1}(x)X(x)] = b(x)$. Since K is unimodular K^{-1} is a matrix with polynomial entries and thus $X(x)$ has polynomial entries iff $K^{-1}X$ does. Clearly if the rewritten system has a solution over $\mathbb{Q}[x]$ it has got to be $B^{-1}b_1$ and thus the first part of the theorem follows. For the second part, note that the solutions to the rewritten system are precisely the vectors $X(x)$ with $B^{-1}b_1$ in the first r components and an arbitrary vector $X_2(x)$ in the others. This along with the one to one correspondence between the solutions of the original system and the rewritten one noted in the beginning of this proof gives us the theorem. □

The statement of the theorem has been made detailed enough that the algorithm implied by it can be easily worked out and hence we omit it. Note that the UCR of A is generally of the form PAK rather than AK, but since dealing with this extra complications involves no new conceptual problems we omitted P.

Finally, the algorithm for the Smith normal form uses repeated applications of the Hermite form algorithm and is developed on the same lines as Kannan and Bachem develop the algorithm for the Smith form of integer matrices from the Hermite form algorithm for integer matrices. The rational canonical form and invariant factors of a matrix A of rationals can be found from the Smith normal form of the polynomial matrix $A - xI$. In the interest of saving space we omit full descriptions here.

5. Conclusions

The algorithms presented here are not to be literally implemented without modification. As pointed out earlier, though the bounds are polynomial, they are very large. However, the purpose of this paper is theoretical—to describe the ideas that lead to a provably polynomial time algorithm. But I believe that a development

similar to those for integer matrices can be accomplished here. In that case while our first polynomial time algorithm had large bounds, a more careful analysis and modifications made by Chou and Collins [8] lead to a practical algorithm and has been implemented as part of the programming package SAC-2 (Collins [7]).

References

[1] S. Barnett and I.S. Pace, Efficient algorithms for linear system calculations, Part 1: Smith form and common divisor of polynomials, *Internat. J. System Sci.* **5** (1974) 403–411.

[2] E. Boedewig, *Matrix Calculus* (North-Holland, Amsterdam, 1956).

[3] G.H. Bradley, Algorithms for Hermite and Smith normal matrices and linear diophantine equations, *Math. Comput.* **25** (1971) 897–907.

[4] W.S. Brown, On Euclid's algorithm and the computation of polynomial greatest common divisors, *J. ACM* **18** (4) (1971) 478–504.

[5] S. Cabay, Exact solution of linear equations, in: *Proc. 2nd Symp. Symbolic and Algebraic Manipulation*, Los Angeles (ACM, New York, 1971) 392–398.

[6] A.E. Collins, Subresultants and reduced polynomial remainder sequences, *J. ACM* **14** (1) (1967) 128–142.

[7] A.E. Collins, *The SAC-2 Manual Version 1.*

[8] T.W. Chou and G.E. Collins, Algorithms for solutions of system of linear diophantine equations, *SIAM J. Comput.* **11** (4) (1982) 687.

[9] J. Edmonds, Systems of distinct representatives and linear algebra, *J. Res. National Bureau of Standards* **71B** (1967) 241–245.

[10] M.A. Frumkin, Polynomial time algorithms in the theory of linear diophantine equations, in: M. Karpinski, ed., *Fundamentals of Computation Theory*, Lecture Notes in Computer Science **56** (Springer, New York, 1977) 386–392.

[11] M. Harrison, *Lectures on Linear Sequential Machines* (Academic Press, New York, 1969).

[12] C. Hermite, Sur l'introduction des variables continuous dans la theorie des nombres, *J. Reine Angew. Math.* **41** (1851) 191–216.

[13] M. Heyman and J.A. Thrope, Transfer equivalence of linear dynamical systems, *SIAM J. Control.* **8** (1970) 19.

[14] E. Horowitz and S. Sahni, On computing the exact determinant of matrices with polynomial entries, *J. ACM* **22** (1975) 38–50.

[15] J. Howell and R.T. Gregory, Solving systems of linear algebraic equations using residue arithmetic I, II and III, *BIT* **9** (1969) 200–224, 324–337; *BIT* **10** (1970) 23–27.

[16] E. Isaacson and H.B. Keller, *Analysis of Numerical Methods* (Wiley, New York, 1966).

[17] T. Kailath, *Linear Systems* (Prentice-Hall, Englewood Cliffs, NJ, 1980).

[18] R.E. Kalman, Irreducible realizations and the degree of a rational matrix, *SIAM J. Appl. Math.* **13** (1965) 520.

[19] R. Kannan, The size of numbers in the analysis of certain algorithms, Ph.D. Dissertation, School of Operations Research, Cornell University, 1980.

[20] R. Kannan and A. Bachem, Polynomial algorithms for computing the Smith and Hermite normal forms of an integer matrix, *SIAM J. Comput.* **8** (4) (1979) 499–507.

[21] D.E. Knuth, *The Art of Computing Vol. 1* (Addison-Wesley, Reading, MA, 1971).

[22] D.E. Knuth, *The Art of Computing Vol. 2* (Addison-Wesley, Reading, MA, 1969) 391.

[23] J.D. Lipson, Symbolic methods for computer methods of linear equations with applications to flow graphs, *Proc. 1968 Inst. on Symbolic Math. Computation* (IBM, Boston, 1969).

[24] M. T. McClennan, The exact solution of systems of linear equations with polynomial coefficients, *J. ACM* **20** (4) (1973) 563–588.

[25] M. Newman, *Integral Matrices* (Academic Press, New York, 1972).

[26] I.S. Pace and S. Barnette, Efficient algorithms for linear system calculations, *Internat. J. System Sci.* **5** (5) (1974) 403–411.

[27] M.M. Rosenbrock, *State Space and Multivariable Theory* (Nelson Press, London, 1970).
[28] H.J.S. Smith, On systems of linear indeterminate equations and congruences, *Philosoph. Transact.* **151** (1861) 293–326.
[29] B.L. Van der Waerden, *Modern Algebra* (Ungar Publishing Company, New York, 1949).
[30] J. von zur Gathen and M. Sieveking, Weitere zum Erfullungsprobleme polynomial äquivalente kombinatorische Aufgaben, in: *Lecture Notes in Computer Science* **43** (Springer, Berlin, 1976).