ALGORITHMS FOR THE SOLUTION OF SYSTEMS OF LINEAR DIOPHANTINE EQUATIONS

by

Tsu-Wu Joseph Chou

Computer Sciences Technical Report #367

September 1979

ALGORITHMS FOR THE SOLUTION OF SYSTEMS OF LINEAR DIOPHANTINE EQUATIONS

BY

TSU-WU JOSEPH CHOU

A thesis submitted in partial fulfillment of the
requirements for the degree of

DOCTOR OF PHILOSOPHY

(Computer Sciences)

at the

UNIVERSITY OF WISCONSIN - MADISON

1979

ACKNOWLEDGEMENTS

I would like to express my sincere gratitude to Professor George E. Collins for all of the time and guidance which he contributed to the preparation of this thesis.

I would also like to acknowledge the support of the National Science Foundation and the Graduate School of the University of Wisconsin.

Finally, I would like to express my sincere gratitude to my mother-in-law Mrs. Dora Yau and my wife Leslie for their encouragement throughout my graduate studies.

TABLE OF CONTENTS

ALGORITHMS FOR THE SOLUTION OF SYSTEMS OF LINEAR DIOPHANTINE EQUATIONS

Tsu-Wu Joseph Chou

Under the supervision of Professor George Edwin Collins

ABSTRACT

A general theory of systems of linear Diophantine equations is described. Two algorithms for the solution of linear Diophantine systems, which well restrain intermediate expression swell, are presented. One, called LDSSBR, is based on ideas of Rosser which were originally used in finding a general solution with smaller coefficients to a linear Diophantine equation and in computing the exact inverse of a non-singular square integral matrix. A hypothetical worst case for this algorithm is analyzed. The other, called LDSMKB, is an extension and improvement of Kannan and Bachem's algorithm for the Smith and the Hermite normal forms of a non-singular square integral matrix. The complexity of this algorithm is investigated and a polynomial time bound is derived. Also a much better coefficient bound is obtained compared to Kannan and Bachem's analysis. These algorithms are implemented by using the infinite precision integer arithmetic capabilities of the SAC-2 system. Their performances are compared. Finally future studies are mentioned.

CHAPTER 1. INTRODUCTION

Section 1.1. Problem Description and Background

Given the following system of m linear equations in n variables with integer coefficients

$$\begin{aligned} a_{1,1}x_1 + a_{1,2}x_2 + \ldots + a_{1,n}x_n &= b_1 \\ a_{2,1}x_1 + a_{2,2}x_2 + \ldots + a_{2,n}x_n &= b_2 \\ &\vdots \\ a_{m,1}x_1 + a_{m,2}x_2 + \ldots + a_{m,n}x_n &= b_m , \end{aligned}$$

one may be interested in finding integral solutions for the variables x_i, $i=1,2,\ldots,n$. Such a system is referred to as a linear Diophantine system. For short, this system can be written in matrix form

$$\begin{aligned} Ax &= b \\ x &\text{ integer,} \end{aligned}$$

where A and b are m by n and m by 1 integral matrices respectively. In fact, A and b can be rational matrices. For if A and b are rational, one can always transform the system $Ax = b$, without changing its solutions, into a linear Diophantine system by multiplying this system with the least common multiple of the denominators of elements in A and b. In this thesis we shall assume A and b are integral for simplicity.

If the system is consistent, then there exists a vector y in Z^n, the set of all integral n-vectors, such that $Ay = b$. y is called

a particular solution of the system or a particular solution. Let S be the set of all the solutions of the homogeneous Diophantine system $Ax = 0$. Then S is an additive Abelian group. Since any solution y^* of the Diophantine system $Ax = b$ can be expressed as a sum of y and some s^* in S and vice versa, the pair (y,S) represents all the solutions of the system and is called a general solution of the system or a general solution. Let $y_1, y_2 \in S$ and $a_1, a_2 \in Z$, the ring of integers. Let "$\cdot$" be the operator of integer by vector multiplication. Since $a_1 \cdot (y_1+y_2) = a_1 \cdot y_1 + a_1 \cdot y_2$, $(a_1 a_2) \cdot y_1 = a_1 \cdot (a_2 \cdot y_1)$ and $(a_1+a_2) \cdot y_1 = a_1 \cdot y_1 + a_2 \cdot y_1$, $(S,\cdot)$ is a Z-module. Let $N=\{N_1,N_2,\ldots,N_p\}$, $p \geq 0$, be a basis of the module. We shall use the pair (y,N) instead of the pair (y,S) to represent the general solution due to complication that S may be an infinite set.

Theories of systems of linear Diophantine equations can be found in the papers by Charnes and Granot [CHG75] and Hurt and Waid [HUW70] and the book by Newman [NEW72]. They give necessary and sufficient conditions for the existence of a solution to a given linear Diophantine system and also formulate a general solution in case the system is consistent. Their formulations involve finding unimodular matrices, U and V, i.e., integral square matrices whose determinants are either 1 or -1, such that UAV is a diagonal matrix or, more restrictively, in the Smith normal form (see [NEW72] for the definition of the Smith normal form) of the coefficient matrix A. Algorithms for computing U, V and the Smith normal form of A are found in [BRA71a] and [KAB78].

Algorithms for solving a system of linear Diophantine equations can

be found in [BLA66], [KNU69], [BRA71a] and [FRU76]. The basic idea used in these algorithms except the one in [FRU76] is to triangularize or diagonalize the augmented coefficient matrix of the system by a sequence of the following elementary column operations: (1) multiplying a column by -1, (2) adding an integer multiple of a column to another column and (3) interchanging two columns. After transforming the augmented coefficient matrix into a triangular or diagonal matrix one can either determine that the system is inconsistent or get a general solution of the system by performing some simple matrix operations. This will be discussed in Chapter 2.

The main difficulty in solving systems of linear Diophantine equations is the very rapid growth of the intermediate results. This effect is called intermediate expression swell [MCC73]. Frumkin [FRU76] has observed that the order of intermediate expression swell in the algorithm by Bradley [BRA71a] can be exponential in the number of equations. Such an algorithm will be impractical even for large computers.

In Chapters 4 and 5 we shall present two algorithms which control intermediate expression swell very well. One algorithm is based on the ideas of Rosser [ROS41] and [ROS52]. His ideas were originally used to find a better general solution of a linear Diophantine equation and to control integer coefficient growth in an integral matrix inversion algorithm. Another algorithm is a modification and extension of the algorithm which computes the Hermite normal form of a non-singular square integral matrix by Kannan and Bachem [KAB78]. These two algorithms are

described as SAC-2 algorithms in the ALDES language; SAC-2 and ALDES are briefly described in Section 1.3. Analyses of these algorithms are also developed in Chapters 4 and 5.

Section 1.2. Some Applications

One important application of linear Diophantine equations is in the theory of equivalent integer programs [BRA71b]. The theory shows that every integer program is equivalent to infinitely many other integer programming problems. The solution to any one problem in this equivalence class is sufficient to determine the solution to every other problem in the class. However, to solve some problems in an equivalent class may be more difficult than to solve other problems in the same class. Given an integer programming problem, one may be interested in constructing an equivalent integer programming problem which has some computational advantages over the original problem.

Consider the integer programming problem

$$\begin{aligned} \text{(IP1)} \qquad & \text{maximize} && cx \\ & \text{subject to} && Ax \leq b \\ & && x \geq 0 \\ & && x \text{ integer,} \end{aligned}$$

where A is an m by n integral matrix, b is an integral m-vector and c is an integral n-vector. By introducing slack variables (IP1) can be rewritten as the integer programming problem with equality constraints

$$\begin{aligned} \text{(IP2)} \qquad & \text{maximize} && cx \\ & \text{subject to} && Ax + Is = b \\ & && x, s \geq 0 \\ & && x, s \text{ integer,} \end{aligned}$$

where I is an identity matrix. If we have an algorithm for solving systems of linear Diophantine equations, then we can use it to decide whether the system

$$Ax + Is = b$$
$$x \text{ , } s \text{ integer}$$

is consistent. If it is, then the solution of the system is given by

(1.1)
$$x = y + Nz$$
$$s = t + Kz$$
$$z \text{ integer}$$

for some integral matrices N and K and integral vectors y and t (which will be computed by our algorithm). z is a vector of integral variables which range over all the integers. Incorporating (1.1) with (IP2) , (IP2) , and hence (IP1) , is equivalent to the integer programming problem

(IP3)
$$\begin{array}{ll} \text{maximize} & cy + (cN)z \\ \text{subject to} & y + Nz \geq 0 \\ & t + Kz \geq 0 \\ & z \text{ integer} . \end{array}$$

If (IP3) has an optimal solution z^* , then $x^*=y+Nz^*$ is an optimal solution to (IP1).

The variables in (IP1) are restricted to be non-negative. However, the variables in (IP3) are arbitrary integers. Conceivably

(IP3) is easier than (IP1). One possible way to solve (IP3) is to solve the system of linear inequality constraints over the integers first, then to optimize the objective function over the set of solutions.

The theory of linear Diophantine equations is closely related to the works of Hermite [HER51] and Smith [SMI61]. The ideas employed in this thesis to restrain the intermediate expression swell while solving a system of linear Diophantine equations can be applied to the construction of the Hermite normal form [NEW72] or the Smith normal form of an integral matrix. The constructions have application to many mathematical problems, such as: elementary divisors of polynomial matrices [GAN59], system theory [ZAP69] and triangular bases for lattices in the geometry of numbers [CAS59].

Section 1.3. The SAC-2 System and the ALDES Language

Due to the inherent integral property of systems of linear Diophantine equations, exact arithmetic must be used in any algorithm for solving such systems. As mentioned in Section 1.1, the problem of solving systems of linear Diophantine equations is exposed to intermediate expression swell. At the present time most computers provide only finite-precision arithmetic. These computers will be overwhelmed very quickly before the parameters, such as the numbers of equations and variables and the sizes of coefficients, of these systems become very large. To overcome this problem, infinite-precision arithmetic should be used. One system which provides infinite-precision arithmetic is the SAC-2 system [COL79a]. The algorithms presented in this thesis are implemented in the SAC-2 system.

The SAC-2 system is a computer-independent system for Symbolic Algebraic Calculation. Its predecessor is the SAC-1 system [COS76a] which is operational on many different computers [COS76b]. The capabilities of the SAC-1 system include list processing, infinite-precision integer and rational number arithmetic, modular arithmetic, integral polynomial greatest common divisor and resultant calculation, integral polynomial real root calculation, integral polynomial factorization over the ring of integers, rational function partial fraction decomposition and integration, Gaussian polynomial complex root calculation, real algebraic number arithmetic and solution of systems of linear equations with polynomial coefficients. The SAC-2 system is still developing. At this writing it contains only partial capabilities of the SAC-1 system.

However, it will contain all the SAC-1 capabilities, and more, when it is fully developed.

SAC-2 differs from SAC-1 in several ways. First, SAC-1 uses the reference count method to reclaim available cells, while SAC-2 uses the garbage collection method. In the reference count method a reference counter is associated with each cell, which counts the number of references to the cell. When the reference count is zero, the corresponding cell is no long used and hence is returned to the available cell list. In the garbage collection method, cell reclamation will not occur until the available cell list becomes null. When cell reclamation, which is called garbage collection, is invoked, free cells are marked and collected together to form the available cell list. More detailed descriptions and comparisons of these two methods can be found in [KNU73] and [COL79a].

Secondly, the data structures in SAC-2 are different from those in SAC-1. In SAC-1 the atom set, that is, the set of all valid atoms, consists of all the single-precision integers on the particular computer of implementation. The location of a list is the location of the first cell of the list. The null list is represented by zero. In SAC-2 the atom set consists of all the single-precision integers whose moduli are less than a pre-selected single-precision positive integer β, a power of two. β is also used as the radix of infinite-precision integers. The location of a list is β plus the index, in the available space array, of the first cell of the list. The null list is represented by β.

Let a be a non-zero β-integer, that is, $a \neq 0$ and $|a| < \beta$.

In SAC-1 a is represented by the list (a) of length one, while in SAC-2 a is represented by the atom a .

Let R be an arbitrary ring. Let $A(x) = \sum_{i=1}^{n} a_i x^{e_i}$ be a non-zero polynomial in R[x] where $a_i \neq 0$ for $i = 1,\ldots,n$ and $0 \leq e_1 < e_2 < \ldots < e_n$. In SAC-1 A(x) is represented by the list $(x,a_n,e_n,\ldots,a_1,e_1)$, while in SAC-2 A(x) is represented by the list $(e_n,a_n,\ldots,e_1,a_1)$. These changes of data structures make SAC-2 somewhat more efficient and convenient than SAC-1.

Third, the host language for SAC-1 is ANSI Fortran [FOR64], while SAC-2 uses ALDES [L0076], [COL79a], which stands for ALgorithm DEScription language. ALDES has several advantages over Fortran. For example, ALDES is block structured, hence ALDES programs are clearer than Fortran programs. ALDES allows direct recursion. This makes programming recursive algorithms in ALDES much easier than in Fortran.

The ALDES language is designed by R. Loos, with some of its features suggested by G. E. Collins, to describe algebraic algorithms. However, it is quite general in nature and can be used to describe many other kinds of algorithms as well. There are two forms of ALDES, namely publication ALDES and implementation ALDES. These two forms differ only in the respect that implementation ALDES has a more restricted character set. In fact, implementation ALDES uses the same character set as ANSI Fortran does. Hence, ALDES programs can be translated into Fortran programs and run on most computers. Publication ALDES uses a larger character set and increases the readability of ALDES algorithms. Algorithms presented in this thesis are written in publication ALDES.

The following is an example of an ALDES algorithm. Each line is prefixed with a line number (not part of the algorithm) for explanation purpose. Line 1 is the algorithm declaration. a is the input. n is

```
1                    n ← EXTENT(a)
2       [Extent. a is an object. n is the extent of a,
3       that is, n = 0 if a is an atom or n is the
4       number of cells used by a if a is a list.]
5            safe a1.
6       (1) [ a an atom.] n←0; if a<β then return.
7       (2) [ a a list.] a'←a; while a'≠() do { ADV(a';
8            a1,a'); n←n+EXTENT(a1)+1 }; return □
```

the output. An ALDES algorithm can be a main program, a function subprogram or a subroutine subprogram. The declaration of a main program is the algorithm name followed by a period. The declaration of a function subprogram is illustrated by the above example. A function subprogram can have only one output. The declaration of a subroutine subprogram is the algorithm name followed by input and output arguments enclosed by parentheses. Inputs (which occur first) and outputs are separated by a semicolon. If the above algorithm is written as a subroutine subprogram, the declaration will be "EXTENT(a;n)". In SAC-2 a subprogram is always written as a function subprogram, if it has only one output argument. Lines 2 through 4 are a comment. Comments can be anywhere statements or declarations are allowed. Line 5 declares a_1 a safe variable. The main purpose in declaring a variable safe is to

improve the efficiency of the translated algorithm. The value of an unsafe variable will be stored in a global stack. Accessing its value will usually entail the loading of an index register. Storing the value of a variable in the stack makes the value accessible to the garbage collector and allows the resumption of an interrupted execution of a recursive algorithm. Lines 6 through 8 are the body of the algorithm. The body is composed of several steps, which end with periods except that the last step ends with the end-of-algorithm mark "□" . Each step is composed of several statements which are separated by semicolons. A formal specification of the ALDES language can be referred to in [L0076].

Section 1.4. Computing Time Analyses

The computing times of algorithms in this thesis will be analyzed by employing the concept of dominance and codominance introduced by Collins [COL71].

Let f and g be real-valued functions defined on a common domain S. We say that f is dominated by g, and write $f \preceq g$, in case there is a positive real number c such that $f(x) \leq c \cdot g(x)$ for all x in S. Note that f and g are not restricted to functions of one variable since the elements of S may be n-tuples. If $f \preceq g$ and $g \preceq f$, then we say that f and g are codominant, and write $f \sim g$. Codominance is clearly an equivalence relation. If $f \preceq g$ but not $g \preceq f$, then we say that f is strictly dominated by g, and write $f \prec g$.

Dominance and codominance have the following fundamental properties.

Theorem 1.1. Let f, f_1, f_2, g, g_1 and g_2 be non-negative real-valued functions on S, and let c be a positive real number. Then

(a) $f \sim cf$;

(b) if $f_1 \preceq g_1$ and $f_2 \preceq g_2$, then $f_1 + f_2 \preceq g_1 + g_2$ and $f_1 f_2 \preceq g_1 g_2$;

(c) if $f_1 \preceq g$ and $f_2 \preceq g$, then $f_1 + f_2 \preceq g$;

(d) $\max(f,g) \sim f + g$;

(e) if $1 \preceq f$ and $1 \preceq g$, then $f + g \preceq fg$;

(f) if $1 \preceq f$, then $f \sim f + c$;

(g) if $S = S_1 \cup S_2$, then $f \preceq g$ on S_1 and $f \preceq g$ on S_2

implies $f \preceq g$ on S .

Proof. See [COL74] for a proof □

Let A be any algorithm and let S be the set of all the valid inputs to A . Define the computing time function of A , $t_A(x)$, to be the number of basic operations performed by the algorithm A when its input is $x \in S$. The computing time function depends on the choice of basic operations. Let B and B' be two finite sets of basic operations such that each operation in B can be realized by a finite sequence of operations in B' , and vice versa. Let t_A and t'_A be the computing time functions of A associated with B and B' respectively. Then, for some $c > 0$, $t'_A(x) \leq ct_A(x)$ for all $x \in S$ since each operation in B can be realized by no more than c operations in B' . Hence, t'_A is dominated by t_A . Similarly t_A is dominated by t'_A . Therefore, t_A and t'_A are codominant. We will only be interested in the codominance equivalence class of t_A without worring about the choice of a specific set of basic operations.

In SAC-2 an integer is represented in radix form with radix β. More precisely, if a is any positive integer, then there exist some unique positive integer n and non-negative β-integers $a_0, a_1, \ldots, a_{n-1}$ such that $a = \sum_{i=0}^{n-1} a_i \cdot \beta^i$ and $a_{n-1} > 0$. If a is any negative integer, then we require instead that a_i be non-positive and $a_{n-1} < 0$. It is natural to define the length of an integer a to be the number of β-digits in its β-radix representation and denote it by $L_\beta(a)$, or just $L(a)$ if β is fixed in the context. $L_\beta(a)$ can be expressed by

the formula

$$(1.2) \qquad L_\beta(a) = \left\{ \begin{array}{ll} 1 & \text{if } a = 0 \\ \lfloor \log_\beta(|a|) \rfloor + 1 & \text{if } a \neq 0 \end{array} \right\} ,$$

where $\lfloor x \rfloor$ is the floor function of the real number x, that is, the largest integer n such that $n \leq x$. Also,

$$(1.3) \qquad L_\beta(a) = \left\{ \begin{array}{ll} 1 & \text{if } a = 0 \\ \lceil \log_\beta(|a| + 1) \rceil & \text{if } a \neq 0 \end{array} \right\} ,$$

where $\lceil x \rceil$ is the ceiling function of the real number x, that is, the smallest integer n such that $n \geq x$.

The length function has the following properties:

$$(1.4) \qquad L(a \pm b) \leq L(a) + L(b) ,$$

$$(1.5) \qquad L(a + b) \sim L(a) + L(b) \quad \text{for } ab \geq 0 ,$$

$$(1.6) \qquad L(ab) \sim L(a) + L(b) \quad \text{if } ab \neq 0 ,$$

$$(1.7) \qquad L(\prod_{i=1}^{n} a_i) \leq \sum_{i=1}^{n} L(a_i) ,$$

$$(1.8) \qquad L(\prod_{i=1}^{n} a_i) \sim \sum_{i=1}^{n} L(a_i) \quad \text{if } |a_i| > 1 \text{ for } 1 \leq i \leq n .$$

(1.7) and (1.8) hold with n variable, not just for each fixed n.

CHAPTER 2. THEORY OF SYSTEMS OF LINEAR DIOPHANTINE EQUATIONS

Let Z be the ring of integers. Let $Z(m,n)$, $m,n>0$, denote the ring of m by n matrices over Z. The <u>determinant</u> of a matrix $A \in Z(n,n)$, denoted by $\det(A)$, is the integer $\sum_{\pi \in P} \sigma(\pi) \prod_{i=1}^{n} a_{i,k_i}$, where P is the set of all permutations of $\{1,2,\ldots,n\}$, $\pi = (k_1,k_2,\ldots,k_n)$ is a permutation in P, $\sigma(\pi)$ is 1 if π is even, -1 if π is odd, and $a_{i,j}$ is the element in the $i^{\underline{th}}$ row and $j^{\underline{th}}$ column of A. The determinant has the property that $\det(AB) = \det(A)\det(B)$ for $A,B \in Z(n,n)$. A matrix A of $Z(n,n)$ is called <u>unimodular</u> if its determinant is 1 or -1. Unimodular matrices have the following properties.

<u>Theorem 2.1.</u> Let $A \in Z(n,n)$. If A is unimodular, then there exists a unimodular matrix $B \in Z(n,n)$ such that $AB = I$, the identity matrix of $Z(n,n)$.

<u>Proof.</u> It is well known that $A \cdot \mathrm{adj}(A) = \det(A)I$, see [HOK71], where $\mathrm{adj}(A)$ is the adjoint of A. Let $B = \det(A)\mathrm{adj}(A)$. Clearly $B \in Z(n,n)$ and $AB = \det(A) \cdot A \cdot \mathrm{adj}(A) = (\det(A))^2 I = I$. Since $1 = |\det(I)| = |\det(AB)| = |\det(A)| \cdot |\det(B)| = |\det(B)|$, B is unimodular. This completes the proof □

<u>Theorem 2.2.</u> Let $A,B \in Z(n,n)$. If A and B are unimodular, then so is AB.

<u>Proof.</u> Clearly $AB \in Z(n,n)$. Since $|\det(AB)| = |\det(A)| \cdot |\det(B)| = 1 \cdot 1 = 1$, AB is unimodular □

Corollary. Let $A_1, A_2, \ldots, A_k \in Z(n,n)$. If A_i are unimodular, then so is $A_1A_2\ldots A_k$.

Consider the following elementary column operations performed on a matrix A of $Z(m,n)$:

(1) multiplying column i by -1 , denoted by $c_1(i)$,

(2) adding k times column j to column i for any integer k , denoted by $c_2(i,j,k)$,

(3) interchanging columns i and j , denoted by $c_3(i,j)$.

Performing the operation $c_1(i)$ on A is equivalent to postmultiplying A by the matrix $[e_1,\ldots,e_{i-1},-e_i,e_{i+1},\ldots,e_n]$, where $e_h \in Z(n,1)$ and the only non-zero element of e_h is the $h^{\underline{th}}$ component, whose value is equal to 1 . Similarly, performing $c_2(i,j,k)$ and $c_3(i,j)$, assuming $i < j$, is equivalent to postmultiplying A by the matrices $[e_1,\ldots,e_i+ke_j,\ldots,e_j,\ldots,e_n]$ and $[e_1,\ldots,e_j,\ldots,e_i,\ldots,e_n]$ respectively. Elementary row operations are similarly defined and denoted by $r_1(i)$, $r_2(i,j,k)$ and $r_3(i,j)$ correspondingly. Performing elementary row operations on A is equivalent to premultiplying A by some matrices of $Z(m,m)$. Matrices which effect elementary operations are called elementary matrices. Clearly elementary matrices are unimodular.

Let A and B be matrices of $Z(m,n)$. We say that B is column (row) equivalent to A if there exists a unimodular matrix $U \in Z(n,n)$ $(Z(m,m))$ such that $B = AU$ $(B = UA)$. Clearly column (row) equivalence is an equivalence relation. We say that B is equivalent to A if there exist unimodular matrices $U \in Z(m,m)$ and $V \in Z(n,n)$

such that $B = UAV$.

Theorem 2.3. Let $A,B \in Z(m,n)$. If B can be obtained from A by a finite sequence of elementary operations, then B is equivalent to A.

Proof. Suppose B can be obtained from A by a sequence of k elementary operations. Then $B = F_1 \ldots F_r A E_1 \ldots E_{k-r}$, where F_i and E_j are elementary matrices which effect corresponding elementary operations. Let $U = F_1 \ldots F_r$ and $V = E_1 \ldots E_{k-r}$. By the corollary of Theorem 2.2, U and V are unimodular. Therefore, B is equivalent to A □

Corollary. Let $A,B \in Z(m,n)$. If B can be obtained from A by a finite sequence of elementary column (row) operations, then B is column (row) equivalent to A.

Let $a_1,\ldots, a_n$ be linearly independent real vectors in n-dimensional real Euclidean space, that is, $t_1a_1 + \ldots + t_na_n = 0$ with $t_1,\ldots,t_n$ real implies $t_1 = \ldots = t_n = 0$. The set L of all points $x = u_1a_1 + \ldots + u_na_n$ with $u_1,\ldots, u_n$ integers is called a lattice. The set $\{a_1,\ldots,a_n\}$ is called a basis of lattice L. A lattice which particularly interests us is the lattice Z^n consisting of all the integral n-vectors. Obviously $\{e_1,\ldots,e_n\}$ is a basis of Z^n, where e_i is the n-vector whose components are zero except that the $i^{\underline{th}}$ component is one. The bases of the lattice Z^n have very close relations with unimodular matrices.

Theorem 2.4. Let $\{a_1,\ldots,a_n\}$ be a basis of lattice L. Then $\{b_1, \ldots,b_n\}$ is a basis of L if and only if $[b_1,\ldots,b_n] = [a_1,\ldots,a_n]$ V

for some unimodular matrix $V \in Z(n,n)$.

Proof. Let $A = [a_1,\ldots,a_n]$ and $B = [b_1,\ldots,b_n]$ with $b_1,\ldots,b_n \in L$. Since $\{a_1,\ldots,a_n\}$ is a basis of L and each $b_i \in L$, each b_i is an integral linear combination of $a_1,\ldots,a_n$. Therefore, $B = AV$ for some $V \in Z(n,n)$.

If $\{b_1,\ldots,b_n\}$ is a basis of L , then $A = BU = AVU$ for some $U \in Z(n,n)$. Since $a_1,\ldots,a_n$ are linearly independent, $VU = I$ so $\det(V)\det(U) = \det(VU) = \det(I) = 1$. Since V and U are integral matrices, $|\det(V)| = 1$. Therefore, V is unimodular.

If V is unimodular, then $b_1,\ldots,b_n$ are linearly independent since $|\det(B)| = |\det(A)|\cdot|\det(V)| = |\det(A)| \neq 0$. By Theorem 2.1, there exists a matrix $U \in Z(n,n)$ such that $VU=I$. Hence, $A = AI = AVU = BU$. Let L' be the lattice with basis $\{b_1,\ldots,b_n\}$. For any $x \in L'$, $x = Bx_0$ for some $x_0 \in Z(n,1)$. So $x = A(Vx_0)$ and x is an integral linear combination of $a_1,\ldots,a_n$. Hence $x \in L$. For any $y \in L$, $y = Ay_0$ for some $y_0 \in Z(n,1)$. So $y = B(Uy_0)$ and $y \in L'$. This implies $L' = L$ and $\{b_1,\ldots,b_n\}$ is a basis of L □

Let $L = Z^n$ and $a_i = e_i$. Then we have the following corollary immediately from Theorem 2.4.

Corollary. $\{b_1,\ldots,b_n\}$ is a basis of Z^n if and only if $[b_1,\ldots,b_n]$ is unimodular.

Let R be a commutative ring with identity. A module over R, or a R-module, is an additive Abelian group M together with a binary

operation "$\cdot$" from $R \times M$ into M such that, for all $r,s \in R$ and $x,y \in M$, $(r + s)\cdot x = r\cdot x + s\cdot x$, $r\cdot(x + y) = r\cdot x + r\cdot y$, $(rs)\cdot x = r\cdot(s\cdot x)$ and $1\cdot x = x$, where 1 is the identity of R. Let $B = (b_1,\ldots,b_n)$ be a finite sequence of elements of M. Then M is said to be <u>finitely generated by B</u> if M is equal to the set $\{\sum_{i=1}^{n} r_i b_i \mid r_i \in R\}$ of all the linear combinations of $b_1,\ldots,b_n$. Furthermore, if $b_1,\ldots,b_n$ are linearly independent, then B is called a <u>basis</u> of M. The <u>rank</u> of a finitely generated module is the unique cardinality of any basis.

Let $A = [a_1,\ldots,a_n]$ be a matrix of $Z(m,n)$, and let $S = \{ k_1a_1 +\ldots+ k_na_n \mid k_i \in Z \}$, the set of all the integral linear combinations of columns of A. Clearly the set S is a Z-module. We will call S the <u>column module</u> of matrix A.

In the following we will prove a slight generalization of a theorem of Hermite [HER51]. The proof is based on integral matrices. However, it can be generalized to matrices over any principal ideal ring.

<u>Theorem 2.5.</u> (<u>Hermite normal form</u>) Every matrix $A \in Z(m,n)$ of rank r is column equivalent to a matrix $H = [h_1,\ldots,h_r,0,\ldots,0]$ such that if $h_{k_i,i}$, the $k_i^{\underline{th}}$ element of h_i, is the first non-zero element of h_i, then $k_1 < k_2 <\cdots< k_r$ and $0 \le h_{k_i,j} < h_{k_i,i}$ for $1 \le j < i \le r$. Such a matrix is unique.

<u>Proof.</u> First we will prove the existence of such a matrix H by induction on m. Let $A = [a_1,\ldots,a_n]$. If $m = 1$, then the rank of A is either 0 or 1. If $\text{rank}(A) = 0$, then $A = 0$ and, trivially, $H = 0$. If $\text{rank}(A) = 1$, then there exists at least one non-zero

element of A. H can be constructed by the following elementary column operations:

(1) For $i = 1,\ldots,n$, if a_i is negative (since $m = 1$, each column consisting of one integer can be treated as an integer), then multiply column i of A by -1.

(2) Let a_k be the smallest non-zero element of A. Interchange columns 1 and k. Such an a_k exists, since the rank of A is one.

(3) If for some i, $2 \leq i \leq n$, $a_i \neq 0$ then subtract $\lfloor a_i/a_1 \rfloor$, the floor function of a_i/a_1, times column 1 from column i and go back to step 2. If $a_i = 0$ for $i = 2,\ldots,n$, then the construction terminates.

Clearly this construction involves only elementary column operations. And it will terminate, since each time step 2 is executed, except the first time, a_1 is decreased at least by 1. At termination of the construction A is of the form $[h_1,0,\ldots,0]$, $h_1 > 0$. Therefore, the existence part of the theorem is true for $m = 1$.

Now assume that the Hermite normal form exists for all matrices in $Z(m,n)$. Let $A \in Z(m+1,n)$, and let A' be the submatrix consisting of the first m rows of A. Then A' is column equivalent to its Hermite normal form $H' = [h'_1,\ldots,h'_{r'},0,\ldots,0]$, r' is the rank of A', and there exists a unimodular matrix $V' \in Z(n,n)$ such that $H' = A'V'$. Let A'' be the submatrix consisting of the last row of A. Then

$$AV' = \begin{bmatrix} A' \\ A'' \end{bmatrix} V' = \begin{bmatrix} A'V' \\ A''V' \end{bmatrix} = \begin{bmatrix} h'_1 \cdots h'_{r'} & 0 & \cdots 0 \\ b_1 \cdots b_{r'} & b_{r'+1} & \cdots b_n \end{bmatrix} .$$

If $\text{rank}(A)$, the rank of A , is equal to r' , then $b_i = 0$ for $i = r'+1,\ldots,n$. In this case, $H = AV'$ is clearly the Hermite normal form of A . If $\text{rank}(A) \neq r'$, and hence $\text{rank}(A) = r'+1$, then $b_i \neq 0$ for some $i > r'$. Let $B = [b_{r'+1},\ldots,b_n] \in Z(1,n-r')$. Then B is column equivalent to the matrix $G = [h,0,\ldots,0]$, $h > 0$, and $G = BU$ for some unimodular matrix $U \in Z(n-r',n-r')$. Let U' be the n by n matrix

$$\begin{bmatrix} I & 0 \\ 0 & U \end{bmatrix} ,$$

where I is the r' by r' identity matrix. Obviously U' is unimodular. Let $H'' = AV'U'$, then

$$H'' = \begin{bmatrix} h'_1 \cdots h'_{r'} & 0 & 0 \cdots 0 \\ b_1 \cdots b_{r'} & h & 0 \cdots 0 \end{bmatrix} .$$

The next thing to do is to subtract $\lfloor b_i/h \rfloor$ times column $r'+1$ of H'' from column i for $i = 1,\ldots,r'$. After this has been done, H'' is in Hermite normal form. This completes the proof of the existence part of this theorem.

Now we will prove the uniqueness of matrix H. Suppose A is column equivalent to G and H which satisfy the hypotheses of this theorem. Let $G = [g_1,\ldots,g_r,0,\ldots,0]$ and $H = [h_1,\ldots,h_r,0,\ldots,0]$.

Let j_i and k_i, $1 \le i \le r$, be the row indices of the first non-zero elements of g_i and h_i respectively. We claim that $j_i = k_i$ for $i = 1,\ldots,r$. Since $g_1,\ldots,g_r$ are linearly independent and any integral linear combination of $g_1,\ldots,g_r$ is an integral linear combination of $a_1,\ldots,a_n$, and vice versa, $\{g_1,\ldots,g_r\}$ is a basis of the column module of A. Similarly $\{h_1,\ldots,h_r\}$ is a basis of the column module of A. Hence, each g_i is an integral linear combination of $h_1,\ldots,h_r$, and vice versa. Suppose $j_1 \neq k_1$. We can assume $j_1 < k_1$ without loss of generality. In this case, g_1 can not be expressed as an integral linear combination of $h_1,\ldots,h_r$, a contradiction. Therefore, $j_1 = k_1$. Now assume $j_i = k_i$ for $i \ge 1$. Then g_{i+1} is an integral linear combination of $h_{i+1},\ldots,h_r$ and h_{i+1} is an integral linear combination of $g_{i+1},\ldots,g_r$. By the same argument as for j_1 and k_1, $j_{i+1} = k_{i+1}$. This proves $j_i = k_i$ for $i = 1,\ldots,r$. Now we claim $g_j = h_j$ for $j = 1,\ldots,r$. Let $G' = [g_1,\ldots,g_r]$ and $H' = [h_1,\ldots,h_r]$. Since $j_i = k_i$ for $i = 1,\ldots,r$, h_j is an integral linear combination of $g_j,\ldots,g_r$ for $1 \le j \le r$. Therefore, there exists a lower triangular integral matrix V such that $H' = G'V$. Similarly there exists a lower triangular integral matrix U such that $G' = H'U$. Since $G' = H'U = G'VU$ and $g_1,\ldots,g_r$ are linearly independent, $VU = I$. Let $u_{i,j}$, $v_{i,j}$ and $w_{i,j}$ be the elements in the $i^{\underline{th}}$ row and $j^{\underline{th}}$ column of U, V and VU respectively. Then $w_{i,i} = v_{i,i}u_{i,i} = 1$, $1 \le i \le r$, since U and V are lower triangular and $VU = I$. This implies that each $u_{i,i}$ and $v_{i,i}$ is either 1 or -1. Since $g_{k_i,i}$ and $h_{k_i,i}$ are both positive, $u_{i,i}$ and $v_{i,i}$ are 1.

By the definition of U, $g_j = \sum_{i=1}^{r} u_{i,j} h_i = \sum_{i=j}^{r} u_{i,j} h_i = h_j + \sum_{i=j+1}^{r} u_{i,j} h_i$. Let $f_j = g_j - h_j = \sum_{i=j+1}^{r} u_{i,j} h_i$. Then $f_{k_{j+1},j} = g_{k_{j+1},j} - h_{k_{j+1},j} = \sum_{i=j+1}^{r} u_{i,j} h_{k_{j+1},i} = u_{j+1,j} h_{k_{j+1},j+1}$. Since $|f_{k_{j+1},j}| = |g_{k_{j+1},j} - h_{k_{j+1},j}| < h_{k_{j+1},j+1}$, $u_{j+1,j} = 0$. Now assume $u_{i,j} = 0$ for $j+1 \le i \le p$. Then $f_j = \sum_{i=p+1}^{r} u_{i,j} h_i$. By a similar argument, $u_{p+1,j} = 0$. This implies $u_{i,j} = 0$ for $j < i \le r$. Therefore, $g_j = h_j$ for $j = 1,\ldots,r$. This completes the proof. □

Constructions of the Hermite normal forms of the coefficient matrices of systems of linear Diophantine equations have been major parts of many algorithms for solving these systems. The performances of these algorithms heavily depend on what construction techniques they use. In the rest of Chapter 2 we would like to discuss how the construction of Hermite normal form can be used in solving systems of linear Diophantine equations. Specific algorithms for solving systems of linear Diophantine equations will be given and analyzed in Chapters 4 and 5.

Consider the system of linear Diophantine equations

$$\begin{aligned} &Ax = b \ , \\ &x \text{ integer} \ , \end{aligned} \tag{2.1}$$

where $A = [a_1,\ldots,a_n] \in Z(m,n)$, $b \in Z(m,1)$. Performing a finite sequence of elementary row operations on the augmented coefficient matrix $[A,b]$ will not change the solution of the system. Performing a finite sequence of elementary column operations on A will change the

solutions of the system. However, there exists a one-to-one relation between the solutions of the original system and the transformed system. Let

$$\text{(2.2)} \qquad \begin{aligned} &By = b \ , \\ &y \text{ integer} \ , \end{aligned}$$

where $B = AU$ for some unimodular matrix, be the transformed sytem. Since U is unimodular, there exists a $V \in Z(n,n)$ such that $UV = I$. Let x^* be a solution of (2.1). Then $b = Ax^* = AUVx^* = B(Vx^*)$, that is, Vx^* is a solution of (2.2). If y^* is a solution of (2.2), then $b = By^* = A(Uy^*)$, that is, Uy^* is a solution of (2.1). So we have the following theorem.

Theorem 2.6. Let $A,B \in Z(m,n)$ and $b \in Z(m,1)$. If $B = AU$ for some unimodular matrix U, then $Ax = b$ has an integral solution if and only if $By = b$ has integral solution. Furthermore y^* is an integral solution of $By = b$ if and only if Uy^* is an integral solution of $Ax = b$.

The basic idea of solving a system of linear Diophantine equations is based on Theorem 2.6. First, we transform the original system to a new system which can be easily solved. Then we solve the new system and transform the solutions of the new system back to the solutions of the original system. For example, let $B = [h_1,\ldots,h_r,0,\ldots,0]$ be the Hermite normal form of A, where $r = \text{rank}(A)$. If $Ax = b$ has an integral solution, then there is an integral vector $y^* = (y_1^*,\ldots,y_n^*)$

such that $By^* = \sum_{i=1}^{n} y_i^* h_i = \sum_{i=1}^{r} y_i^* h_i = b$. Let k_i be the row index of the first non-zero element of h_i for $i = 1,\ldots,r$. Then $y_1^*,\ldots, y_r^*$ can be determined inductively by the formulas

$$y_1^* = b_{k_1} / h_{k_1,1} . \tag{2.3}$$

$$y_i^* = (b_{k_i} - \sum_{j=1}^{i-1} y_j^* h_{k_j,j}) / h_{k_i,i} , \quad 1 < i \leq r . \tag{2.4}$$

y_i^*, $r < i \leq n$, can be any integers. Choose $y_i^* = 0$, $r < i \leq n$, arbitrarily. Then $y^* = (y_1^*,\ldots,y_r^*,0,\ldots,0)$ is a particular solution of the Diophantine system $By = b$. Hence, Uy^* is a particular solution of the Diophantine system $Ax = b$. Note that B is not necessarily in Hermite normal form. Formulas (2.3) and (2.4) will work well as long as $[h_1,\ldots,h_r]$ is in column echelon form.

Let $A \in Z(m,n)$. Let S be the set of all the solutions to the homogeneous linear Diophantine system

$$\begin{aligned} &Ax = 0 , \\ &x \text{ integer} . \end{aligned} \tag{2.5}$$

Then S is a Z-module. We will call S the <u>solution module</u> of system (2.5). The next theorem tells us how to find a basis of a homogeneous linear Diophantine system.

<u>Theorem 2.7.</u> Let $A \in Z(m,n)$, and let $U = [u_1,\ldots,u_n]$ be a unimodular matrix such that $AU = [h_1,\ldots,h_r,0,\ldots,0]$ and $h_1,\ldots,h_r$ are linearly independent. Then $N = \{u_{r+1},\ldots,u_n\}$ is a basis of the

solution module of the homogeneous linear Diophantine system $Ax = 0$.

Proof. Let M be the column module of $[u_{r+1},\ldots,u_n]$. Since U is unimodular, $u_{r+1},\ldots,u_n$ are linearly independent. Therefore, N is a basis of M . Let S be the solution module of the linear Diophantine system $Ax = 0$. Let $x \in S$. Then x is an element of the lattice Z^n . By the corollary of Theorem 2.4, $\{u_1,\ldots,u_n\}$ is a basis of Z^n . Hence, $x = \sum_{i=1}^{n} x_i u_i$. Since $x \in S$, $0 = Ax = \sum_{i=1}^{n} x_i(Au_i) = \sum_{i=1}^{r} x_i h_i$ since $Au_i = h_i$ if $1 \le i \le r$ and $Au_i = 0$ if $r < i \le n$. Since the h_i's are linearly independent, $x_i = 0$ for $i = 1,\ldots,r$. Hence, $x = \sum_{i=r+1}^{n} x_i u_i \in M$. This implies $S \subset M$. Let $x \in M$. Then $x = \sum_{i=r+1}^{n} x_i u_i$ and $Ax = \sum_{i=r+1}^{n} x_i(Au_i) = 0$, since $Au_i = 0$ for $r < i \le n$. Hence, $x \in S$. This implies $M \subset S$. Therefore, $S = M$ and N is a basis of S □

To compute N , one can construct the matrix C by adjoining the n by n identity matrix I to the bottom of A ,

$$C = \begin{bmatrix} A \\ I \end{bmatrix} ,$$

then perform a sequence of elementary column operations on C such that

$$C' = CU = \begin{bmatrix} AU \\ U \end{bmatrix} = \begin{bmatrix} h_1 \ldots h_r & 0 \ldots 0 \\ u_1 \ldots u_r & u_{r+1} \ldots u_n \end{bmatrix} ,$$

where U is the unimodular matrix which effects the sequence of elementary column operations and $h_1,\ldots,h_r$ constitute a column echelon

matrix. Two such sequences will be given in Sections 4.2 and 5.2. By Theorem 2.7, $N = \{u_{r+1},\ldots,u_n\}$ is a basis of the solution module of the Diophantine system $Ax = 0$.

To compute a particular solution x^*, if one exists, of the Diophantine system $Ax = b$, one can construct the matrix B by adjoining the n by 1 zero matrix to $-b$,

$$B = \begin{bmatrix} -b \\ 0 \end{bmatrix} .$$

If any y_i^*, defined by (2.3) and (2.4), is not an integer, then no solution exists. Otherwise add y_i^* times column i of C' to B for $i = 1,\ldots,r$. The first m elements of B must be zero. And B is of the form

$$\begin{bmatrix} 0 \\ b^* \end{bmatrix} .$$

Since $Ab^* = A \sum_{i=1}^{r} y_i^* u_i = \sum_{i=1}^{r} y_i^*(Au_i) = \sum_{i=1}^{r} y_i^* h_i = b$, $x^* = b^*$ is a particular solution.

CHAPTER 3. AUXILIARY ALGORITHMS

In this chapter we will present subalgorithms used by the main algorithms for solving linear Diophantine systems. These algorithms will be described in publication ALDES. For those algorithms previously existing in the SAC-2 system we will list their specifications only. The algorithm descriptions for these algorithms will be available soon in [COL79b]. The computing times for these algorithms will be given without proofs, which will be in [COL79a]. The computing times for algorithms in this thesis do not include the time for garbage collections. Ignoring the time for garbage collections will not change the codominance equivalence class of the computing time of any main program as explained in [COL79a].

Algorithms presented in this thesis can be divided into four categories: list processing algorithms, integer arithmetic algorithms, integral vector algorithms and integral matrix algorithms.

Let S be an arbitrary set called an atom set. Elements in S are called atoms. A list over S is recursively defined to be a finite sequence $(a_1,\ldots,a_n)$, $n \geq 0$, such that each a_i is either an atom in S or a list over S. The length of a list a, written as length(a), is the length of a as a sequence. The list of length zero is called the null list. Let a be the list $(a_1,\ldots,a_n)$, $n \geq 0$. Then the inverse of a, written as inv(a), is the list $(a_n,\ldots,a_1)$. Let a be the non-null list $(a_1,a_2,\ldots,a_n)$, $n > 0$. Then the reductum of a, written as red(a), is the list $(a_2,\ldots,a_n)$. If a

is the list $(a_1,\ldots,a_m)$, $m \geq 0$, and b is the list $(b_1,\ldots,b_n)$, $n \geq 0$, then the concatenation of the lists a and b , written as conc(a,b) , is the list $(a_1,\ldots,a_m,b_1,\ldots,b_n)$. An object is either an atom or a list. Let a be an object and let b be the list $(b_1,\ldots,b_n)$. Then the composition of a and b , written as comp(a,b), is the list $(a,b_1,\ldots,b_n)$. Let a be the non-null list $(a_1,\ldots,a_n)$, $n > 0$. Then the first of a , written as first(a), is the object a_1 .

Now we shall discuss how SAC-2 lists are represented in a computer. Let M be any positive integer. An integer m is called an M-integer if $|m| < M$. Let β be a preselected positive power of 2, say 2^{ζ} for some $\zeta > 0$. The atom set in the SAC-2 system is the set of β-integers. In order to represent the 47 characters in the ANSI Fortran character set, one must choose $\zeta \geq 6$. In order to represent atoms as β-integers and include as many atoms in the atom set as possible, one should choose ζ as large as possible but not greater than or equal to ξ , the number of bits in a computer word in implementation. In general one may choose $\zeta = \xi - 3$. One bit is reserved for the sign of a β-integer and two bits are reserved for overflows in β-integer arithmetic. The null list is represented by the integer β . A non-null list is represented by linked cells from a large linear array called SPACE. Each cell consists of two consecutive elements in the array SPACE. The location of a cell is β plus the index in the array SPACE of the first of the two array elements. The location of a list L is the location of the first cell of L if L is a non-null list, or the integer β if L is the null list. After defining the location

of a list the representation of a non-null list L in a computer can be described inductively as follows: the first array element of the first cell of L stores the location of red(L) , and the second array element of the first cell of L stores first(L) if first(L) is an atom or the location of first(L) if first(L) is a list.

Following are specifications, descriptions and computing time analyses of the list processing algorithms used in this thesis.

a←FIRST(L)

[First. L is a non-null list. a is the first element of L.]

L'←RED(L)

[Reductum. L is a non-null list. L' is the reductum of L.]

SFIRST(L,a)

[Set first. L is a non-null list. a is an object. The first element of L is changed to a.]

SRED(L,L')

[Set reductum. L is a non-null list. L' is a list. The reductum of L is changed to L'.]

ADV(L;a,L')

[Advance. L is a non-null list. a=first(L). L'=red(L).]

M←COMP(a,L)

[Composition. a is an object. L is a list. M is the composition of a and L.]

Theorem 3.1. The computing times of the algorithms FIRST, RED, SFIRST, SRED, ADV and COMP are all codominant with 1.

n←LENGTH(L)

[Length. L is a list. n=length(L).]

M←INV(L)

[Inverse. L is a list. M is the inverse of L. The list L is transformed into M.]

L'←SUFFIX(L,b)

[Suffix. L is a list $(a_1,\ldots,a_n)$, $n\geq 0$. b is an object. L' is the list $(a_1,\ldots,a_n,b)$. L is modified to produce L'.]

Theorem 3.2. Let L be a list of length n and let b be any object. Then $t_{LENGTH}(L) \sim t_{INV}(L) \sim t_{SUFFIX}(L,b) \sim n + 1$.

L←CONC(L_1,L_2)

[Concatenation. L_1 and L_2 are lists. L is the concatenation of L_1 and L_2. The list L_1 is modified.]

Theorem 3.3. $t_{CONC}(L_1,L_2) \sim 1$ if L_2 is a null list; $t_{CONC}(L_1,L_2) \sim$ length(L_1) + 1 if L_2 is a non-null list.

a←LELT(A,i)

[List element. A is a list. $1\leq i\leq$length(A). a is the $i^{\underline{th}}$ element of A.]

Theorem 3.4. $t_{LELT}(A,i) \sim i$.

L←LEINST(A,i,a)

[List element insertion. A is the list $(a_1,\ldots,a_n)$ of objects. i is a β-integer, $0\le i\le n$. a is an object. If i=0, then L= $(a,a_1,\ldots,a_n)$. If i=n, then L=$(a_1,\ldots,a_n,a)$. Otherwise, L= $(a_1,\ldots,a_i,a,a_{i+1},\ldots,a_n)$. A is modified.]

safe j.

(1) [i=0.] if i=0 then { L←COMP(a,A); return }.

(2) [i>0.] L←A; A'←A; for j=2,...,i do A'←RED(A'); A"← RED(A'); A"←COMP(a,A"); SRED(A',A"); return □

Theorem 3.5. $t_{LEINST}(A,i,a) \sim i + 1$.

Proof. Let t_1 and t_2 be the computing times of steps 1 and 2 respectively. Clearly $t_1 \sim 1$ and, if $i > 0$, then $t_2 \sim i$. If i=0, then $t_{LEINST}(A,i,a) = t_1 \sim 1 = i + 1$. If $i > 0$, then $t_{LEINST}(A,i,a) = t_1 + t_2 \sim i + 1$. □

M←LEROT(L,i,j)

[List element rotation. L is a list $(a_1,\ldots,a_n)$ of objects, n>0. i and j, $1\le i\le j\le n$, are β-integers. If i=j, then M=L. Otherwise $M=(a_1,\ldots,a_{i-1},a_j,a_i,\ldots,a_{j-1},a_{j+1},\ldots,a_n)$. L is modified.]

safe LEROT.

(1) [i=j.] M←L; if i=j then return.

(2) [i<j.] for k=1,...,i-1 do L←RED(L); ADV(L;a,L'); for k=i,...,j-1 do { b←FIRST(L'); SFIRST(L',a); a←b; L'←RED(L') }; SFIRST(L,a); return □

Theorem 3.6. $t_{LEROT}(L,i,j) \sim 1$ if $i = j$; $t_{LEROT}(L,i,j) \sim j$ if $i < j$.

Proof. Let t_1 and t_2 be the computing times of steps 1 and 2 respectively. Then $t_1 \sim 1$ and $t_2 \sim (i-1) + (j-i) + 1$. If $i = j$ then $t_{LEROT}(A,i,j) = t_1 \sim 1$. If $i < j$ then $t_{LEROT}(A,i,j) = t_1 + t_2 \sim 1 + (i-1) + (j-1) + 1 = i + j \sim j$. □

B←REDUCT(A,i)

[Reductum. A is a list. i is a non-negative β-integer not greater than length(A). B=A if i=0. Otherwise, B is the $i^{\underline{th}}$ reductum of A.]

safe REDUCT.

(1) B←A; for j=i,...,i do B←RED(B); return □

Theorem 3.7. $t_{REDUCT}(A,i) \sim i + 1$.

Proof. Obvious. □

Let a be an integer. For $|a| < \beta$, a is represented as a β-integer. For $|a| \geq \beta$, let $\sum_{i=0}^{n-1} a_i \beta^i$ be the β-radix representation of a , that is, $a_0,\ldots,a_{n-1}$ are β-integers such that $a_i \geq 0$ for $i = 0,\ldots, n-2$ and $a_{n-1} > 0$ if a is positive, or $a_i \leq 0$ for $i = 0,\ldots, n-2$ and $a_{n-1} < 0$ if a is negative. Then a is represented by the list $(a_0,a_1,\ldots,a_n)$.

Following are specifications, descriptions and computing time analyses of integer arithmetic algorithms.

s←ISIGNF(A)

[Integer sign function. A is an integer. s=sign(A).]

Theorem 3.8. $t_{ISIGNF}(A) \preceq L(A)$.

B←INEG(A)

[Integer negation. A is an integer. B=-A.]

Theorem 3.9. $t_{INEG}(A) \sim L(A)$.

C←ISUM(A,B)

[Integer sum. A and B are integers. C=A+B.]

Theorem 3.10. $t_{ISUM}(A,B) \preceq L(A) + L(B) \sim \max \{L(A),L(B)\}$.

C←IDIF(A,B)

[Integer difference. A and B are integers. C=A-B.]

Theorem 3.11. $t_{IDIF}(A,B) \preceq L(A) + L(B) \sim \max \{L(A),L(B)\}$.

s←ICOMP(A,B)

[Integer comparison. A and B are integers. s=sign(A-B).]

Theorem 3.12. $t_{ICOMP}(A,B) \preceq L(A) + L(B) \sim \max \{L(A),L(B)\}$.

C←IPROD(A,B)

[Integer product. A and B are integers. C=A•B.]

Theorem 3.13. $t_{IPROD}(A,B) \sim 1$ if $AB = 0$; $t_{IPROD}(A,B) \preceq L(A)L(B)$ if $AB \neq 0$.

IQR(A,B;Q,R)

[Integer quotient and remainder. A and B are integers, B≠0. Q is the quotient [A/B] and R=A-B·Q.]

Theorem 3.14. $t_{IQR}(A,B) \sim L(B)$, if $L(A) < L(B)$; $t_{IQR}(A,B) \preceq L(B)\ \{\ L(A)-L(B)+1\ \}$, if $L(A) \geq L(B)$.

C←IQ(A,B)

[Integer quotient. A and B are integers, B≠0. C is the quotient [A/B].]

Theorem 3.15. $t_{IQ}(A,B) \sim L(B)$, if $L(A) < L(B)$; $t_{IQ}(A,B) \preceq L(B)\ \{\ L(A)-L(B)+1\ \}$, if $L(A) \geq L(B)$.

IDEGCD(a,b;c,u_1,v_1,u_2,v_2)

[Integer doubly extended greatest common divisor algorithm. a and b are integers. c=gcd(a,b). $au_1+bv_1=c$ and $au_2+bv_2=0$. If a≠0 and b≠0 then $u_1 \leq |b|/2c$, $v_1 \leq |a|/2c$, $u_2=-b/c$ and $v_2=a/c$. Otherwise $u_1=v_2=\text{sign}(a)$, $v_1=\text{sign}(b)$ and $u_2=-\text{sign}(b)$.]

Theorem 3.16. $t_{IDEGCD}(a,b) \preceq n(m-k+1)$ where $m = \max\{L(a),L(b)\}$, $n = \min\{L(a),L(b)\}$ and $k = L(\gcd(a,b))$.

Let V be a vector in Z^n . We shall represent V by the list $(v_1,\ldots,v_n)$, where each v_i is the integer representation of the $i^{\underline{th}}$ component of V . The norm of a vector $V = (v_1,\ldots,v_n) \in Z^n$, denoted by norm(V) or |V| , is defined to be the non-negative integer $\max_{1\leq i\leq n} |v_i|$.

Following are descriptions and computing time analyses of integral

vector algorithms.

B←VIAZ(A,n)

[Vector of integers, adjoin zeros. A is the vector $(a_1,\ldots,a_m)$. n is a non-negative β-integer. B is the vector $(a_1,\ldots,a_m, 0,\ldots,0)$ of m+n components. A is modified.]

safe k.

(1) B←(); for k=1,...,n do B←COMP(0,B); B←CONC(A,B); return □

Theorem 3.17. Let $A \in Z^m$. Then $t_{VIAZ}(A,n) \sim m + n$.

Proof. Constructing the zero vector in Z^n requires computing time $\sim n$. By Theorem 3.3, it requires computing time $\sim m$ to concatenate A and the zero vector. Thus, $t_{VIAZ}(A,n) \sim m + n$. □

B←VINEG(A)

[Vector of integers negation. A is an integral vector. B=-A.]

safe a,A'.

(1) B←(); A'←A; repeat { ADV(A';a,A'); b←INEG(a); B←COMP(b,B) } until A'=(); B←INV(B); return □

Theorem 3.18. Let $A \in Z^n$. Then $t_{VINEG}(A) \preceq nL(|A|)$.

Proof. The computing time of each execution of the repeat-loop is $\preceq L(|A|)$. The repeat-loop will be executed n times. The computing time for inverting the intermediate list B is $\sim n$. Therefore, $t_{VINEG}(A) \preceq nL(|A|) + n \sim nL(|A|)$. □

C←VISUM(A,B)

[Vector of integers sum. A and B are vectors in Z^n. C=A+B.]

safe a,A',b,B'.

(1) C←(); A'←A; B'←B; repeat { ADV(A';a,A'); ADV(B';b,B');
c←ISUM(a,b); C←COMP(c,C) } until A'=(); C←INV(C); return □

Theorem 3.19. Let $A, B \in Z^n$. Then $t_{VISUM}(A,B) \preceq n\{L(|A|)+L(|B|)\}$.

Proof. The computing time of each execution of the repeat-loop is $\preceq L(|A|) + L(|B|)$. The repeat-loop will be executed n times. The computing time of inverting the intermediate list C is $\sim n$.
Therefore, $t_{VISUM}(A,B) \preceq n\{L(|A|)+L(|B|)\} + n \sim n\{L(|A|)+L(|B|)\}$. □

C←VIDIF(A,B)

[Vector of integers difference. A and B are vectors in Z^n.
C=A-B.]

(1) C←VISUM(A,VINEG(B)); return □

Theorem 3.20. Let $A, B \in Z^n$. Then $t_{VIDIF}(A,B) \preceq n\{L(|A|)+L(|B|)\}$.

Proof. Immediate from Theorem 3.18 and Theorem 3.19. □

C←VISPR(a,A)

[Vector of integers scalar product. a is an integer. A is an integral vector. C=a·A.]

safe a',A',i,n.

(1) [a=0.] if a=0 then { n←LENGTH(A); C←(); for i=1,...,n do
C←COMP(0,C); return }.

(2) [a=1.] if a=1 then { C←A; return }.

(3) [a=-1.] if a=-1 then { C←VINEG(A); return }.

(4) [General case.] C←(); A'←A; repeat { ADV(A';a',A')
c←IPROD(a,a'); C←COMP(c,C) } until A'=(); C←INV(C);
return □

Theorem 3.21. Let $A \in Z^n$. Then $t_{VISPR}(a,A) \sim n$ if $a = 0$; $t_{VISPR}(a,A) \sim 1$ if $a = 1$; $t_{VISPR}(a,A) \preceq nL(a)L(|A|)$ otherwise.

Proof. The cases that $a = 0$ and $a = 1$ are trivial. If $a = -1$, then $t_{VISPR}(a,A) \sim t_{VINEG}(A) \preceq nL(|A|) \sim nL(a)L(|A|)$. For $|a| > 1$, the repeat-loop will be executed n times and the computing time of each execution of the loop is $\preceq L(a)L(|A|)$. Therefore, $t_{VISPR}(a,A) \preceq nL(a)L(|A|)$. □

C←VILCOM(a,b,A,B)

[Vector of integers linear combination. a and b are integers. A and B are integral vectors in Z^n. C=a•A+b•B.]

safe C.

(1) S←VISPR(a,A); T←VISPR(b,B); C←VISUM(S,T); return □

Theorem 3.22. Let $A, B \in Z^n$. Then

(1) $t_{VILCOM}(a,b,A,B) \sim n$, for $a = b = 0$,

(2) $t_{VILCOM}(a,b,A,B) \preceq nL(b)L(|B|)$, for $a = 0$ and $b \neq 0$,

(3) $t_{VILCOM}(a,b,A,B) \preceq n\,L(a)L(|A|)$, for $a \neq 0$ and $b = 0$,

(4) $t_{VILCOM}(a,b,A,B) \preceq n\{L(a)L(|A|)+L(b)L(|B|)\}$, for $ab \neq 0$.

Proof. Case 1. $t_{VILCOM}(a,b,A,B) \sim t_{VISPR}(0,A) + t_{VISPR}(0,B) + t_{VISUM}(0,0) \sim n + n + n \sim n$.

Case 2. If $b = 1$, then $t_{VILCOM}(a,b,A,B) \sim t_{VISPR}(0,A) +$

$t_{VISPR}(1,B) + t_{VISUM}(0,B) \preceq n + 1 + nL(|B|) \sim nL(|B|) \sim nL(b)L(|B|)$.
If $b \neq 1$, then $t_{VILCOM}(a,b,A,B) \sim t_{VISPR}(0,A) + t_{VISPR}(b,B) +$
$t_{VISUM}(0,bB) \preceq n + nL(b)L(|B|) + nL(|bB|) \sim nL(b)L(|B|)$ since
$L(|bB|) = L(|b|\cdot|B|) \sim L(b) + L(|B|)$.

Case 3. Similar to case 2.

Case 4. This case can be divided into four subcases: (i) $a = 1$, $b = 1$, (ii) $a = 1$, $b \neq 1$, (iii) $a \neq 1$, $b = 1$ and (iv) $a \neq 1$, $b \neq 1$. We are going to show subcase (iv). The other three subcases can be shown in a similar way. For $a \neq 1$ and $b \neq 1$,
$t_{VILCOM}(a,b,A,B) \sim t_{VISPR}(a,A) + t_{VISPR}(b,B) + t_{VISUM}(aA,bB) \preceq$
$nL(a)L(|A|) + nL(b)L(|B|) + n\{L(|aA|) + L(|bB|)\} \sim n\{L(a)L(|A|) +$
$L(b)L(|B|) + L(a) + L(|A|) + L(b) + L(|B|)\} \sim n\{L(a)L(|A|) +$
$L(b)L(|B|)\}$. □

Combining the four cases of Theorem 3.22 we have immediately the following corollary.

Corollary. Let $A, B \in Z^n$. Then $t_{VILCOM}(a,b,A,B) \preceq$
$n\{sign(|a|)L(a)L(|A|) + sign(|b|)L(b)L(|B|) + 1\}$.

W←VIERED(U,V,i)

[Vector of integers, element reduction. $U=(u_1,\ldots,u_n)$ and $V=(v_1,\ldots,v_n)$ are integral n-vectors. $1\leq i\leq n$. $v_i\neq 0$. $W=U-qV$, where $q=[u_i/v_i]$.]

safe u,v.

(1) u←LELT(U,i); v←LELT(V,i); q←IQ(u,v); if q=0 then W←U else { q←INEG(q); W←VISPR(q,V); W←VISUM(U,W) }; return □

Theorem 3.23. Let $U, V \in Z^n$. Let u_i and v_i be the $i^{\underline{th}}$ elements of U and V respectively. If $|u_i| < |v_i|$, then $t_{VIERED}(U,V,i) \sim L(v_i) + i$. If $|u_i| \geq |v_i|$, then $t_{VIERED}(U,V,i) \preceq n\{L(|V|)\{L(u_i)-L(v_i)+1\} + L(|U|)\}$.

Proof. Let $q = [u_i/v_i]$. If $|u_i| < |v_i|$, then $q = 0$ so $t_{VIERED}(U,V,i) \sim t_{LELT}(U,i) + t_{LELT}(V,i) + t_{IQ}(u_i,v_i) \sim i + i + L(v_i) \sim L(v_i) + i$. If $|u_i| \geq |v_i|$, then $t_{VIERED}(U,V,i) \sim t_{LELT}(U,i) + t_{LELT}(V,i) + t_{IQ}(u_i,v_i) + t_{INEG}(q) + t_{VISPR}(q,V) + t_{VISUM}(U,qV) \preceq i + i + L(q)L(v_i) + L(q) + nL(q)L(|V|) + n\{L(|U|)+L(|qV|)\} \sim nL(q)L(|V|) + nL(|U|) \sim n\{L(|V|)\{L(u_i)-L(v_i)+1\} + L(|U|)\}$. □

VIUT(U,V,i;U',V')

[Vector of integers, unimodular transformation. $U=(u_1,\ldots,u_n)$ and $V=(v_1,\ldots,v_n)$ are vectors in Z^n with $u_i \neq 0$. $[U',V']=[U,V]K$ where K is a unimodular matrix, depending on u_i and v_i, whose elements are obtained from IDEGCD.]

safe c,i,u,v.

(1) u←LELT(U,i); v←LELT(V,i); IDEGCD(u,v;c,p,q,r,s);
U'←VILCOM(p,q,U,V); V'←VILCOM(r,s,U,V); return □

Theorem 3.24. Let $U = (u_1,\ldots,u_n) \in Z^n$ and $V = (v_1,\ldots,v_n) \in Z^n$. Then $t_{VIUT}(U,V,i) \preceq n\{L(v_i)L(|U|) + L(u_i)L(|V|)\}$.

Proof. Let p, q, r and s be the integers computed by the algorithm IDEGCD such that $pu_i + qv_i = \gcd(u_i,v_i)$ and $ru_i + sv_i = 0$. Then $L(p) \leq L(v_i)$, $L(q) \leq L(u_i)$, $L(r) \leq L(v_i)$ and $L(s) \leq L(u_i)$. The computing time for accessing u_i and v_i is $\sim i$, by Theorem 3.4.

The time for computing p, q, r and s is $\preceq L(u_i)L(v_i)$ from Theorem 3.16. The computing time of the calls to the algorithm VILCOM is $\preceq n \{ L(v_i)L(|U|) + L(u_i)L(|V|) \}$ by Theorem 3.22. Therefore $t_{VIUT}(U,V,i) \preceq i + L(u_i)L(v_i) + n \{ L(v_i)L(|U|) + L(u_i)L(|V|) \}$. □

Let A be an integral matrix in Z(m,n) . We shall represent A by the list $(A_1,\ldots,A_n)$, where each A_i is the list representation of the $i^{\underline{th}}$ column (as an m-vector) of A . The <u>norm</u> of a matrix $A = (a_{ij}) \varepsilon Z(m,n)$, denoted by norm(A) or |A| , is defined to be the non-negative integer $\max_{1\le i,j\le n} |a_{ij}|$.

Following are descriptions and computing time analyses of integral matrix algorithms.

B←MIAIM(A)

[Matrix of integers, adjoin identity matrix. A is an m by n matrix of integers. B is the matrix obtained by adjoining an n by n identity matrix to the bottom of A. A is modified.]

safe i,j,n.

(1) n←LENGTH(A); A'←INV(A); B←(); for i=1,...,n do { ADV(A';A_1,A'); T←(); for j=1,...,n do if i=j then T←COMP(1,T) else T←COMP(0,T); A_1←CONC(A_1,T); B←COMP(A_1,B) }; return □

<u>Theorem 3.25.</u> Let $A \varepsilon Z(m,n)$. Then $t_{MIAIM}(A) \sim n(m+n)$.

<u>Proof.</u> The computing time for computing the length of A and inverting A is $\sim n$. The computing time for constructing the $i^{\underline{th}}$ unit vector $E_i \varepsilon Z^n$ is $\sim n$. The computing time for concatenating the $i^{\underline{th}}$

column of A and E_i is $\sim m$. Thus, $t_{MIAIM}(A) \sim n + n(n+m)$ $\sim n(m+n)$.

B←MINNCT(A)

[Matrix of integers, non-negative column transformation. $A=(a_{ij})$ is an m by n integral matrix. $B=(b_{ij})$ is the m by n integral matrix with $b_{ij}=a_{ij}$ if $a_{1j}\geq 0$ and $b_{ij}=-a_{ij}$ if $a_{1j}<0$. A is modified.]

safe MINNCT.

(1) B←A; A'←A; repeat { A_1←FIRST(A'); a←FIRST(A_1); if ISIGNF(a)<0 then { A_1←VINEG(A_1); SFIRST(A',A_1) }; A'←RED(A') } until A'=(); return □

Theorem 3.26. Let $A \in Z(m,n)$. Then $t_{MINNCT}(A) \preceq mnL(|A|)$.

Proof. Let $A_i = (a_{1i},\ldots,a_{mi})$ be the $i^{\underline{th}}$ column of A . The computing time of the $i^{\underline{th}}$ execution of the repeat-loop is $\sim t_{ISIGNF}(a_{1i}) + t_{VINEG}(A_i) \preceq L(|A|) + mL(|A|) \sim mL(|A|)$. The repeat-loop will be executed n times. Therefore, $t_{MINNCT}(A) \preceq mnL(|A|)$. □

B←MICS(A)

[Matrix of integers column sort. A is an integral matrix with non-negative elements in the first row. B is an integral matrix obtained by sorting columns of A such that elements of the first row are in descending order. A is modified.]

safe MICS.

(1) [Bubble sort columns.] repeat { A'←A; s←0; while

RED(A')≠() do { ADV(A';A_1,A"); A_2←FIRST(A"); a_1←FIRST(A_1); a_2←FIRST(A_2); if ICOMP(a_1,a_2)<0 then { SFIRST(A',A_2) SFIRST(A",A_1); s←1 }; A'←A" } } until s=0; B←A; return □

Theorem 3.27. Let $A \in Z(m,n)$. Then $t_{MICS}(A) \preceq n^2L(|A|)$.

Proof. The computing time of ICOMP is the most significant part of the computing time of MICS. Since there are at most $n(n-1)$ integer comparisons in MICS and each integer comparison needs time $\preceq L(|A|)$, $t_{MICS}(A) \preceq n(n-1)L(|A|) + 1 \preceq n^2L(|A|)$. □

B←MICINS(A,V)

[Matrix of integers column insertion. A is an m by n integral matrix represented by the list $(A_1,A_2,\ldots,A_n)$, where A_i is the list $(a_{1i},\ldots,a_{mi})$ representing column i of A and $a_{11} \geq a_{12} \geq \ldots \geq a_{1n}$. $V=(v_1,\ldots,v_m)$ is an integral vector with $v_1 < a_{11}$. Let i be the largest integer such that $a_{1i} \geq v_1$. Then B is the matrix represented by the list $(A_1,\ldots,A_i,V,A_{i+1},\ldots,A_n)$. A is modified.]

safe MICINS.

(1) [Initialize.] A'←A; A"←RED(A'); v←FIRST(V).

(2) [Loop.] while A"≠() & ICOMP(FIRST(FIRST(A")),v)≥0 do { A'←A"; A"←RED(A") }.

(3) [Finish.] B←COMP(V,A"); SRED(A',B); B←A; return □

Theorem 3.28. Let $A \in Z(m,n)$ and $V \in Z^m$. Then $t_{MICINS}(A,V) \preceq n\{L(|A|) + L(|V|)\}$.

Proof. Let t_i be the computing time of step i. It can be easily verified that $t_1 \sim 1$, $t_2 \preceq n \{ L(|A|) + L(|V|) \}$ and $t_3 \sim 1$. Thus, $t_{MICINS}(A,V) \preceq n \{ L(|A|) + L(|V|) \}$. □

CHAPTER 4. AN ALGORITHM BASED ON IDEAS OF ROSSER

Section 4.1. Introduction

The main difficulty in solving linear Diophantine systems is the very rapid growth of coefficients. In this chapter we shall present an algorithm, called LDSSBR, which restrains coefficient growth very well in the early stage of the algorithm. Although the growth becomes fast in the final stage of the algorithm, it is still quite moderate compared to some other algorithms. As our empirical results will show, the length of the largest coefficients while solving the Diophantine system $Ax = b$ is typically bounded by $nL(n|A|)$, where n is the number of variables in the system.

The ideas employed in the algorithm LDSSBR to control coefficient growth come from J. B. Rosser. He used the ideas in finding a general solution with much smaller coefficients to a linear Diophantine equation (see [ROS41]) and computing the exact inverse of a non-singular square integral matrix to minimize computing time (see [ROS52]). Consider the following linear Diophantine equation

$$a_1x_1 + a_2x_2 + \ldots + a_nx_n = b \ . \tag{4.1}$$

Without loss of generality, let us assume $a_1 \geq a_2 \geq \ldots \geq a_n \geq 0$ and $a_1 > 0$, since if $a_i < 0$ we can replace x_i by $-x_i$, if $a_i < a_j$ for some $i < j$ we can interchange x_i and x_j and if $a_1 = 0$ the equation becomes trivial. Rosser's algorithm begins with the following matrix

$$C = \begin{bmatrix} a_1 & a_2 & \dots & a_n \\ 1 & 0 & \dots & 0 \\ 0 & 1 & \dots & 0 \\ . & . & \dots & . \\ 0 & 0 & \dots & 1 \end{bmatrix} .$$

Let $c_{i,j}$ be the element in the $i^{\underline{th}}$ row and $j^{\underline{th}}$ column of C, and let C_j be the $j^{\underline{th}}$ column of C. Then the algorithm consists of the following steps:

(1) while $c_{1,2} \neq 0$ do $\{ C_1 \leftarrow C_1 - \lfloor c_{1,1}/c_{1,2} \rfloor C_2$; sort $C_1, \dots, C_n$ in descending order according to their leading elements $\}$.

(2) At this point the matrix C has the form

$$\begin{bmatrix} c & 0 & \dots & 0 \\ U_1 & U_2 & \dots & U_n \end{bmatrix} ,$$

where $U_j \in Z^n$ and $c = \gcd(a_1,\dots,a_n)$. If $c \nmid b$ then (4.1) has no solution, otherwise $X = qU_1 + y_2U_2 + \dots + y_nU_n$, where $X = (x_1,\dots,x_n)^T$, $q = b/c$ and $y_2, \dots, y_n$ are arbitrary integers, is a general solution of (4.1).

One may easily verify that the matrix $U = [U_1,\dots,U_n]$ is unimodular. Since $c_{1,1}$ and $c_{1,2}$ are the largest and the second largest elements in the first row of C, the integer $\lfloor c_{1,1}/c_{1,2} \rfloor$ computed in step 1 is

usually small. So Rosser's algorithm usually will find a U with evenly small elements, especially when n is large, while other methods usually will find a U with some large elements and the rest quite small, including many zeros and ones.

In solving the linear Diophantine system $Ax = b$, one would like to compute a unimodular matrix U such that $A' = AU$ is a column echelon matrix, i.e., if $A' \in Z(m,n)$, $r = \text{rank}(A')$ and k_j, $1 \le j \le r$, is the row index of the leading non-zero element of column j of A' then $1 \le k_1 < \ldots < k_r$ and columns $r+1,\ldots,n$ of A' are zero. In the algorithm LDSSBR, U is obtained by executing statements similar to those in step 1 of Rosser's algorithm r, the rank of A, times. In fact U is the product of r unimodular matrices $U_1,\ldots, U_r$ associated with executions of step 1 of Rosser's algorithm. The sizes of the elements in the U_i's will directly affect the rate of coefficient growth in solving a linear Diophantine system. Since, in general, Rosser's algorithm computes U_i's with evenly small elements, the algorithm controls the coefficient growth very well.

Rosser's algorithm is very simple. However, analysis of this algorithm is not as simple as one might suppose. Only recently has a thorough analysis been obtained (see [KNU69], Section 4.5.3) for the special case $n = 2$.

In Section 4.2 we will describe the algorithm LDSSBR, which employs the ideas of Rosser. In Section 4.4 we will analyze the computing time of a "special case" of the algorithm LDSSBR. The coefficients of the special case are related to the members of a sequence generalized from

the famous Fibonacci sequence. We will called it the $n^{\underline{th}}$ order Fibonacci sequence. Some important properties of the $n^{\underline{th}}$ order Fibonacci sequence will be investigated in Section 4.3 and used in the computing time analysis of the algorithm LDSSBR in Section 4.4.

Section 4.2. The Algorithm

LDSSBR(A,b;x*,N)

[Linear Diophantine system solution, based on Rosser's ideas. A is an m by n integral matrix. b is an integral m-vector. If the Diophantine system Ax=b is consistent, then x* is a particular solution and N is a list of basis vectors of the solution module of Ax=0. Otherwise, x* and N are null lists. A and b are modified.]

safe b_1,C',C_2,m,N,n,s,x*.

(1) [Initialize.] n←LENGTH(A); m←LENGTH(b).

(2) [Adjoin identity matrix to A and zero vector to -b.] C←MIAIM(A); B←VIAZ(VINEG(b),n).

(3) [Sort columns of C.] C←MINNCT(C); C←MICS(C).

(4) [Pivot row zero.] C_1←FIRST(C); if FIRST(C_1)=0 then go to 6.

(5) [Eliminate pivot row.] repeat { B←VIERED(B,C_1,1); C←RED(C); if C=() then s←0 else { C_2←FIRST(C); s←FIRST(C_2); if s≠0 then { C_1←VIERED(C_1,C_2,1); C←MICINS(C,C_1); C_1←C_2 } } } until s=0; n←n-1.

(6) [System inconsistent?] ADV(B;b_1,B); if b_1≠0 then { x*←(); N←(); return }.

(7) [Remove pivot row.] C'←C; while C'≠() do { C_1←FIRST(C'); C_1←RED(C_1); SFIRST(C',C_1); C'←RED(C') }; m←m-1.

(8) [Finished?] if m>0 then { if n>0 then go to 3 else go to 6 }; x*←B; N←C; return □

Step 1 computes the number of variables n and the number of equations m in the system. Step 2 constructs the matrix $C = \begin{bmatrix} A \\ I \end{bmatrix}$, where I is the identity matrix in $Z(n,n)$, and the vector $B = \begin{bmatrix} -b \\ 0 \end{bmatrix}$, where 0 is the zero vector in Z^n. Steps 3 to 8 form a loop which computes a unimodular matrix U such that AU is a column echelon matrix by repeatedly executing a step similar to step 1 in Rosser's algorithm described in Section 4.1, checks the consistency of the system and computes a particular solution if there exists one. Step 3 makes the elements in the first row non-negative and sorts them in descending order by performing two kinds of elementary column operations, multiplying a column by -1 and interchanging two columns. This is a preparatory step for step 5 which employs Rosser's ideas. Step 4 checks whether the first row is zero and, if so, skips the execution of step 5. Step 5 basically does two things: (1) reduce the size of the first element of B by repeatedly subtracting multiples of the first column of C from B and (2) perform step 1 of Rosser's algorithm. Note that two kinds of elementary column operations are involved in this step, i.e., subtracting a multiple of a column from another column and interchanging two columns. Also note that, because the first column of C is no longer useful in later computations after leaving the repeat-loop, it is deleted from C through the algorithm RED before leaving the repeat-loop. Step 6 checks the consistency of the system. If it is inconsistent, then it returns the null list for x^* and N. Step 7 removes the first row of C, whose elements are zero. The deletions of these unnecessary columns and rows make this algorithm more

efficient. Later when we verify the validity of the algorithm, we will assume these deleted columns and rows are attached. Step 8 decides whether there are remaining rows and columns.

Theorem 4.1. The algorithm LDSSBR is valid.

Proof. Let C be the $(m+n)$ by n matrix which initially is $\begin{bmatrix} A \\ I \end{bmatrix}$ and let B be the $(m+n)$-vector which initially is $\begin{bmatrix} -b \\ 0 \end{bmatrix}$. Let C' denote the matrix consisting of rows $1,\ldots,m$ of C, let C'' denote the matrix consisting of rows $m+1,\ldots,m+n$ of C, let B' denote the vector consisting of the first m components of B and let B'' denote the vector consisting of the last n components of B. Performing any one of the three kinds of elementary column operations on C will preserve (1) the unimodularity of C'', (2) the validity of the relation $C' = AC''$ and (3) the consistency of $C'x = B'$. Suppose $\bar{C}$ is obtained from C by performing an elementary column operation on C. Then $\bar{C} = CE$, where E is the elementary matrix (hence, a unimodular matrix) reflecting the elementary column operation. Let $\bar{C}'$ and $\bar{C}''$ be the submatrices of $\bar{C}$ corresponding to C' and C'' of C. Then $\bar{C}' = C'E$ and $\bar{C}'' = C''E$. Since $\det(\bar{C}'') = \det(C'')\det(E) = \pm \det(C'')$, $\bar{C}''$ is unimodular iff C'' is unimodular. Since $\bar{C}' - A\bar{C}'' = C'E - AC''E = (C' - AC'')E$ and $\det(E) \neq 0$, $\bar{C}' - A\bar{C}'' = 0$ iff $C' - AC'' = 0$, i.e., $\bar{C}' = A\bar{C}''$ iff $C' = AC''$. Since $\bar{C}' = C'E$ and E is unimodular, $\bar{C}'x = B'$ is consistent iff $C'x = B'$ is consistent by Theorem 2.6. Obviously, C'' is unimodular and $C' = AC''$ when C is constructed in step 2. So we can conclude that C'' is unimodular and $C' = AC''$ at

any point of the algorithm LDSSBR.

Adding an integral multiple of a column of C to B will preserve (i) the validity of the relation $B' = AB'' - b$ and (ii) the consistency of $C'x = B'$. Suppose $\bar{B}$ is obtained from B by adding kC_i, where $C_i = \begin{pmatrix} C_i' \\ C_i'' \end{pmatrix}$ is the $i^{\underline{th}}$ column of C, to B. Then $\bar{B} = B + kC_i$. Let $\bar{B}'$ and $\bar{B}''$ be the vectors consisting of the first m elements and the last n elements of $\bar{B}$ respectively. Then $\bar{B}' = B' + kC_i'$ and $\bar{B}'' = B'' + kC_i''$. Note that $C_i' = AC_i''$ by the fact that $C' = AC''$. Since $\bar{B}' - A\bar{B}'' = (B' + kC_i') - A(B'' + kC_i'') = (B' + kAC_i'') - (AB'' + kAC_i'')$ $= B' - AB''$, $\bar{B}' = A\bar{B}'' - b$ iff $B' = AB'' - b$. If $x = (x_1,\ldots,x_n)^T$ is a solution of the system $C'x = B'$, then $y = (x_1,\ldots,x_i+k,\ldots,x_n)^T$ is a solution of the system $C'y = \bar{B}'$, since $C'y = C'\{x + (0,\ldots,k,\ldots,0)^T\} = C'x + kC_i' = B' + kC_i' = \bar{B}'$. Similarly, if $y = (y_1,\ldots,y_n)^T$ is a solution of the system $C'y = \bar{B}'$, then $x = (y_1,\ldots,y_i-k,\ldots,y_n)^T$ is a solution of the system $C'x = B'$. This implies $C'x = B'$ is consistent iff $C'x = \bar{B}'$ is consistent. Obviously, $B' = AB'' - b$ when B is constructed in step 2. So we can conclude that $B' = AB'' - b$ and that $C'x = B'$ is consistent iff $Ax = b$ is consistent at any point in the algorithm LDSSBR.

If B' passes all the tests in step 6 (when we say B' passes the $i^{\underline{th}}$ test, we mean that the $i^{\underline{th}}$ element of B' at the $i^{\underline{th}}$ execution of step 6 is equal to zero), then the final value of B' is zero. Thus, $AB'' - b = 0$, and hence, B'' is a particular solution of the system $Ax = b$. At this point C' is in column echelon form. Since $C' = AC''$

and C'' is unimodular, by Theorem 2.7, $\{C''_{r+1},\ldots,C''_n\}$ is a basis of the solution module of the system $Ax = 0$, where $C'' = [C''_1,\ldots,C''_n]$ and $r = \text{rank}(A)$.

If $Ax = b$ is consistent, then $C'x = B'$ is consistent at any point in the algorithm LDSSBR. Suppose B' has passed the first $i-1$ tests and is ready for the $i^{\underline{th}}$ test. Let $C' = [C'_1,\ldots,C'_n]$ and let t_j, $1 \le j \le n$, be the row index of the leading non-zero element of C'_j. Then $t_1 < \ldots < t_p \le i$ and $t_{p+1},\ldots, t_n > i$ for some $p \ge 0$. Let $x^* = (x_1,\ldots,x_n)^T$ be a solution of $C'x = B'$. Then $B' = C'x^* = \sum_{j=1}^n x_j C'_j$. We claim $x_1 = \ldots = x_p = 0$. Since B' passed the first $i-1$ tests, $B'_1 = \ldots = B'_{i-1} = 0$, where $B' = (B'_1,\ldots,B'_n)^T$.

Let $1 \le h \le p$ and assume that $x_1 = \ldots = x_{h-1} = 0$. Then $B' = \sum_{j=h}^n x_j C'_j$. Let $k = t_h$. Then $B'_k = \sum_{j=h}^n x_j C'_{k,j}$ where $C'_{k,j}$ is the $k^{\underline{th}}$ component of C'_j. But $C'_{k,j} = 0$ for $j > h$ so $B'_k = x_h C'_{k,h}$. If $k < i$ then $B'_k = 0$ and $C'_{k,h} \ne 0$ so $x_h = 0$. If $k = i$ then $h = p$ and $|B'_k| < C'_{k,h}$ by virtue of step 5 so again $x_h = 0$. By induction on h, therefore, $x_1 = \ldots = x_p = 0$ and $B' = \sum_{j=p+1}^n x_j C'_j$. But $C'_{i,j} = 0$ for $j > p$ so $B'_i = \sum_{j=p+1}^n x_j C'_{i,j} = 0$. Hence B' also passes the $i^{\underline{th}}$ validity test. This completes the validity proof for LDSSBR. □

Section 4.3. The n^{th} Order Fibonacci Sequences

Given a sequence of numbers $A(0), A(1), \ldots$, an equation relating $A(k)$ to its predecessors in the sequence, valid for all integers k greater than some integer k_0, is called a recurrence relation. If $A(k)$ is determined by its immediate n predecessors for $k \geq n$, then the recurrence relation is said to be of order n. A recurrence relation is linear if $A(k)$ is a linear combination of $A(k-i)$, $1 \leq i \leq n$, for $k \geq n$. Consider the linear recurrence relation of order n with constant coefficients

$$A(k) = c_1A(k-1) + c_2A(k-2) + \ldots + c_nA(k-n) , \tag{4.2}$$

for $k \geq n$, $c_1, c_2, \ldots, c_n$ constant and $c_n \neq 0$. With the recurrence relation (4.2) we associate the equation

$$x^n - c_1x^{n-1} - c_2x^{n-2} - \ldots - c_n = 0 , \tag{4.3}$$

which is called the characteristic equation for (4.2). The n complex roots $\alpha_1, \alpha_2, \ldots, \alpha_n$, which may not be distinct, of (4.3) are called the characteristic roots for (4.3). A sequence $A(0), A(1), A(2), \ldots$ of complex numbers, which satisfies the recurrence relation (4.2) is called a solution of (4.2).

Theorem 4.2. Let $\alpha_1, \alpha_2, \ldots, \alpha_n$ be the characteristic roots of the characteristic equation for the recurrence relation (4.2). Then the sequence

$$A(k) = \sum_{i=1}^{n} e_i\alpha_i^k \quad , \quad k=0,1,2,\ldots \tag{4.4}$$

is a solution of the recurrence relation (4.2) for any constants e_1, e_2, ... , e_n .

Proof. For any $k \geq n$, $A(k) = \sum_{i=1}^{n} e_i \alpha_i^k = \sum_{i=1}^{n} e_i \alpha_i^{k-n} \alpha_i^n$ $= \sum_{i=1}^{n} e_i \alpha_i^{k-n} \sum_{j=1}^{n} c_j \alpha_i^{n-j} = \sum_{i=1}^{n} e_i \sum_{j=1}^{n} c_j \alpha_i^{k-j} = \sum_{j=1}^{n} c_j \sum_{i=1}^{n} e_i \alpha_i^{k-j}$ $= \sum_{j=1}^{n} c_j A(k-j)$. This implies that the sequence (4.4) satisfies the recurrence relation (4.2), and hence, is a solution of (4.2). □

Theroem 4.3. Let $\alpha_1, \alpha_2, \ldots, \alpha_n$ be the characteristic roots of the characteristic equation for the recurrence relation (4.2). Let B(0), B(1), B(2), ... be a solution of the recurrence relation (4.2). If $\alpha_1, \alpha_2, \ldots, \alpha_n$ are distinct, then $B(k) = \sum_{i=1}^{n} e_i \alpha_i^k$, $k = 0, 1, 2, \ldots$, for some constants $e_1, e_2, \ldots, e_n$.

Proof. Let $A(k) = \sum_{i=1}^{n} e_i \alpha_i^k$, $k = 0, 1, 2, \ldots$, where $e_1, e_2, \ldots, e_n$ are any constants. By Theorem 4.2, the sequence A(k), $k = 0, 1, 2, \ldots$, is a solution of the recurrence relation (4.2). Since a solution of a recurrence relation is completely determined by the first n members of the solution, it suffices to show that there exist some constants $e_1, e_2, \ldots, e_n$ such that $B(k) = \sum_{i=1}^{n} e_i \alpha_i^k$ for $k = 0, 1, \ldots, n-1$. The coefficient matrix of this system is

$$\begin{bmatrix} 1 & 1 & \cdots & 1 \\ \alpha_1 & \alpha_2 & \cdots & \alpha_n \\ \cdot & \cdot & \cdots & \cdot \\ \alpha_1^{n-1} & \alpha_2^{n-1} & \cdots & \alpha_n^{n-1} \end{bmatrix} .$$

This matrix is well known as the Vandermonde matrix. Its determinant is the product $\prod_{1\le i<j\le n} (\alpha_j - \alpha_i)$ (see [KNU73]). Since $\alpha_1, \alpha_2, \ldots, \alpha_n$ are distinct, the determinant of the coefficient matrix is non-zero. Therefore, there is a unique solution for $e_1, e_2, \ldots, e_n$. □

By Theorems 4.2 and 4.3, (4.4) represents all the solutions of the recurrence relation (4.2), hence, is called a general solution of (4.2) provided that the characteristic roots of the characteristic equation for (4.2) are distinct.

One sequence of special interest to us is the sequence $F_n(k)$, $k = 0, 1, 2, \ldots$, satisfying the linear recurrence relation of order $n \ge 2$

$$(4.5) \qquad F_n(k) = F_n(k-1) + F_n(k-n) , \quad k \ge n ,$$

with initial values $F_n(k) = 0$ for $k = 0, 1, \ldots, n-2$ and $F_n(n-1) = 1$. We will call this sequence the n^{th} order Fibonacci sequence, members of this sequence the n^{th} order Fibonacci numbers and the linear recurrence relation (4.5) the n^{th} order Fibonacci recurrence relation.

Let $C_n(x) = x^n - x^{n-1} - 1$, $n \ge 2$. Then $C_n(x) = 0$ is the characteristic equation for the n^{th} order Fibonacci recurrence relation. Let $\alpha_1, \ldots, \alpha_n$ be the characteristic roots of $C_n(x) = 0$. Then $\alpha_1, \ldots, \alpha_n$ are non-zero and distinct, since $C_n(0) \ne 0$ and $\gcd(C_n(x), C_n'(x)) = 1$. By Theorem 4.3, $F_n(k) = \sum_{i=1}^{n} e_i \alpha_i^k$ for some constants $e_1, \ldots, e_n$. $e_1, \ldots, e_n$ can be obtained by solving

the system

$$\begin{bmatrix} 1 & 1 & \cdots & 1 \\ \alpha_1 & \alpha_2 & \cdots & \alpha_n \\ . & . & \cdots & . \\ \alpha_1^{n-1} & \alpha_2^{n-1} & \cdots & \alpha_n^{n-1} \end{bmatrix} \begin{bmatrix} e_1 \\ e_2 \\ . \\ e_n \end{bmatrix} = \begin{bmatrix} F_n(0) \\ F_n(1) \\ . \\ F_n(n-1) \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ . \\ 1 \end{bmatrix} .$$

Let D be the coefficient matrix. D is a Vandermonde matrix and $\det(D) = \prod_{1\le j<k\le n} (\alpha_k - \alpha_j)$. Let D_i be the matrix obtained by replacing the $i^{\underline{th}}$ column of D by the right-hand side of the system. It is easy to see that $\det(D_i) = (-1)^{n+i} \prod_{1\le j<k\le n \& j\ne i \& k\ne i} (\alpha_k-\alpha_j)$.

By Cramer's rule, $e_i = \det(D_i)/\det(D) = (-1)^{n+i}\{\prod_{k=1}^{i-1} (\alpha_i - \alpha_k)\}^{-1}$.

$\{ \prod_{k=i+1}^{n}(\alpha_k - \alpha_i)\}^{-1} = (-1)^{2i} \prod_{1\le k\le n \ \&\ k\ne i} (\alpha_i - \alpha_k)^{-1}$

$= \{ \prod_{1\le k\le n \ \&\ k\ne i} (\alpha_i - \alpha_k)\}^{-1} = \{ C_n'(\alpha_i) \}^{-1} = \{ n\alpha_i^{n-1} - (n-1)\alpha_i^{n-2} \}^{-1}$

$= \alpha_i\{ n(\alpha_i^n - \alpha_i^{n-1}) + \alpha_i^{n-1}\}^{-1} = \alpha_i (\alpha_i^{n-1} + n)^{-1}$. Therefore,

$$(4.6) \qquad F_n(k) = \sum_{i=1}^{n} \alpha_i^{k+1} (\alpha_i^{n-1} + n)^{-1} .$$

Since the number of sign variations of the coefficients of $C_n(x)$ is 1, the equation $C_n(x) = 0$ has one and only one positive real root by Descartes' rule of signs. Let α be the positive real root of $C_n(x) = 0$. Then $1 < \alpha < 2$, since $C_n(1) = -1 < 0$ and $C_n(2) = 2^n - 2^{n-1} - 1 > 0$ for $n \ge 2$. We claim that α is the root of largest modulus of $C_n(x) = 0$. Let $D_n(x) = C_n(x)/(x-\alpha)$ and let $E_n(x) = D_n(\alpha x)/\alpha^{n-2}$. Then it is equivalent to show that all the roots

of $E_n(x) = 0$ lie within the unit circle $|x| = 1$, x a complex variable. We need the following theorem due to Rouché in order to establish the above claim.

Theorem 4.4 (Rouché). Let C be the unit circle $|x| = 1$. If $R(x)$ and $S(x)$ are two univariate polynomials with complex coefficients and $|R(x)| > |S(x)|$ on C, then $R(x) + S(x)$ and $R(x)$ have the same number of roots inside C.

Proof. See [MAR69]. □

Let $P_{n,i}(x) = (i\alpha-i+1)x^{n-i} + (\alpha-1)x^{n-i-1} + \ldots + (\alpha-1)$ for $1 \le i < n$. And let $P^*_{n,i}(x) = x^{n-i}P_{n,i}(1/x) = (\alpha-1)x^{n-i} + \ldots + (\alpha-1)x + (i\alpha-i+1)$. Then, on C, $|P^*_{n,i}(x)| = |\overline{P^*_{n,i}(x)}| = |\bar{x}^{n-i}||\overline{P_{n,i}(1/x)}| = |\overline{P_{n,i}(1/x)}| = |P_{n,i}(x)|$, where $\bar{\gamma}$ denotes the complex conjugate of γ. Let $R(x) = (i\alpha-i+1)P_{n,i}(x)$ and let $S(x) = -(\alpha-1)P^*_{n,i}(x)$. Then $|R(x)| > |S(x)|$ on C since $|i\alpha-i+1| > |\alpha-1|$. Hence, by Theorem 4.4, $R(x) + S(x)$ and $P_{n,i}(x)$ have the same number of roots inside C. Note that $R(x) + S(x) = (i\alpha-i+1)P_{n,i}(x) - (\alpha-1)P^*_{n,i}(x) = \{(i\alpha-i+1)^2 - (\alpha-1)^2\}x^{n-i} + \{(i\alpha-i+1)(\alpha-1) - (\alpha-1)^2\}x^{n-i-1} + \ldots + \{(i\alpha-i+1)(\alpha-1) - (\alpha-1)^2\}x$ $= \{(i-1)(\alpha-1)+1\}x\{((i+1)\alpha-i)x^{n-i-1} + (\alpha-1)x^{n-i-2} + \ldots + (\alpha-1)\}$ $= \{(i-1)(\alpha-1) + 1\}xP_{n,i+1}(x)$ and $R(x) + S(x)$ has the same degree as $P_{n,i}(x)$. So $R(x) + S(x)$, and hence $P_{n,i+1}(x)$, and $P_{n,i}(x)$ have the same number of roots outside or on C. Note that this is true for $1 \le i \le n-2$. Since $D_n(x) = (x^n-x^{n-1}-1)/(x-\alpha) = x^{n-1} + (\alpha-1)x^{n-2} + \ldots + \alpha^{n-2}(\alpha-1)$ and $E_n(x) = D_n(\alpha x)/\alpha^{n-2} = \alpha x^{n-1} + (\alpha-1)x^{n-2} + \ldots +$

$(\alpha-1) = P_{n,1}(x)$, $E_n(x)$ and $P_{n,n-1}(x)$ have the same number of roots outside C . Obviously, $P_{n,n-i}(x) = \{(n-1)\alpha - (n-2)\}x + (\alpha-1)$ has no root outside or on C , since $|(n-1)\alpha - (n-2)| > |\alpha-1|$. So we have the following theorem.

Theorem 4.5. Let α be the positive real root of the characteristic equation $C_n(x) = x^n - x^{n-1} - 1 = 0$ for the $n^{\underline{th}}$ order Fibonacci sequence. Then α is the characteristic root of largest modulus.

The next theorem gives bounds on $n^{\underline{th}}$ order Fibonacci numbers.

Theorem 4.6. Let $F_n(k)$, $k = 0, 1, 2, \ldots$, be the $n^{\underline{th}}$ order Fibonacci sequence, and let α be the positive real root of the characteristic equation $C_n(x) = x^n - x^{n-1} - 1 = 0$ for the $n^{\underline{th}}$ order Fibonacci sequence. Then $\alpha^{k-2n+2} \le F_n(k) \le \alpha^{k-n+1}$ for $k \ge n-1$.

Proof by induction on k. Obviously, $\alpha^{k-2n+2} \le F_n(k) \le \alpha^{k-n+1}$ for $k = n-1, n, \ldots, 2n-2$, since $F_n(n-1) = F_n(n) = \ldots = F_n(2n-2) = 1$ and $\alpha > 1$. Now assume that the induction hypothesis is true for any $k \le m$, with $m \ge 2n-2$. Then $F_n(m+1) = F_n(m) + F_n(m-n+1) \ge \alpha^{m-2n+2} + \alpha^{m-3n+3} = \alpha^{m-3n+3}(\alpha^{n-1}+1) = \alpha^{m-3n+3}\cdot\alpha^n = \alpha^{(m+1)-2n+2}$ and $F_n(m+1) = F_n(m) + F_n(m-n+1) \le \alpha^{m-n+1} + \alpha^{m-2n+2} = \alpha^{m-2n+2}(\alpha^{n-1}+1) = \alpha^{m-2n+2}\cdot\alpha^n = \alpha^{(m+1)-n+1}$. □

Let $a_1, a_2, \ldots, a_n$ be non-negative integers, $n \ge 2$, with $a_1 \ge a_2 \ge \ldots \ge a_n$ and $a_1 \ne 0$. Let V_i , $1 \le i \le n$, be the $i^{\underline{th}}$ column of the matrix

$$\begin{bmatrix} a_1 & a_2 & \dots & a_n \\ 1 & 0 & \dots & 0 \\ 0 & 1 & \dots & 0 \\ . & . & \dots & . \\ 0 & 0 & \dots & 1 \end{bmatrix} .$$

Consider the following steps:

(1) $k \leftarrow -1$.

(2) if $v_2 = 0$ then go to 4, where v_2 is the first element of V_2 .

(3) $k \leftarrow k + 1$; $A_k \leftarrow V_1$; $q_k \leftarrow \lfloor v_1/v_2 \rfloor$, where v_1 is the first element of V_1; $V \leftarrow V_1 - q_k V_2$; set s to be the largest integer such that $v_s \geq v$, where v_s and v are the first elements of V_s and V respectively; for $i = 2, \dots, s$ do $V_{i-1} \leftarrow V_i$; $V_s \leftarrow V$; go to 2.

(4) for $i = 1, \dots, n$ do $A_{k+i} \leftarrow V_i$.

Let r be the final value for the index k . The above steps generate a sequence $q_0, q_1, \dots, q_r$ of quotients, which is called the quotient sequence of $a_1, \dots, a_n$, and a sequence $A_0, A_1, \dots, A_{r+n}$ of vectors. Let c_j and $x_{i,j}$, $1 \leq i \leq n$, denote the first and the $(i+1)^{\text{th}}$ elements of A_j respectively. Then the sequence $c_0, c_1, \dots, c_{r+n}$ and the sequence $x_{i,0}, x_{i,1}, \dots, x_{i,r+n}$, $1 \leq i \leq n$, are called the remainder sequence and the i^{th} multiplier sequence of $a_1, \dots, a_n$ respectively. Note that $c_{r+1} = \gcd(a_1, \dots, a_n)$, $c_{r+2} = \dots = c_{r+n} = 0$ and $c_i = \sum_{j=1}^{n} x_{i,j} a_j$ for $0 \leq i \leq r+n$.

Now we would like to derive bounds on the lengths of the remainder and quotient sequences of n non-negative integers $a_1 \geq a_2 \geq \ldots \geq a_n \geq 0$ with $a_1 \neq 0$.

Theorem 4.7. Let $F_n(0), F_n(1), F_n(2), \ldots$ be the n^{th} order Fibonacci sequence. Then $F_n(m) = \sum_{i=0}^{m-n} F_n(i) + F_n(n-1)$ for $m \geq n$.

Proof by induction on m. For $m = n$, $F_n(n) = F_n(0) + F_n(n-1) = \sum_{i=0}^{m-n} F_n(i) + F_n(n-1)$. Assume the induction hypothesis is true for $m = k \geq n$. $F_n(k+1) = F_n(k) + F_n(k-n+1) = \sum_{i=0}^{k-n} F_n(i) + F_n(n-1) + F_n(k-n+1) = \sum_{i=0}^{k-n+1} F_n(i) + F_n(n-1)$. □

For $n \geq 2$ let $S_{n,m}$ be the set $\{(a_1,a_2,\ldots,a_n) \mid a_1 \geq a_2 \geq \ldots \geq a_n \geq 0, a_1 \neq 0$ and the length of the remainder sequence of $a_1, a_2, \ldots, a_n$ is $m+1\}$.

Theorem 4.8. Let $c_0, c_1, \ldots, c_m$ be the remainder sequence of $(a_1, a_2, \ldots, a_n) \in S_{n,m}$. Then $c_i \geq F_n(m-i)$ for $i = 0, 1, \ldots, m$.

Proof by induction on m. For $m=n-1$, $c_0 \geq 1 = F_n(n-1)$ and $c_i = 0 = F_n(n-i-1)$ for $i = 1, \ldots, n-1$. Now assume the induction hypothesis is true for $m = k \geq n-1$. Let $(a_1, a_2, \ldots, a_n) \in S_{n,k+1}$ and let $c_0, c_1, \ldots, c_{k+1}$ and $q_0, q_1, \ldots, q_{k-n+1}$ be the remainder and the quotient sequences of $a_1, a_2, \ldots, a_n$. Let $a_1' \geq a_2' \geq \ldots \geq a_n'$ be a permutation of $a_2, \ldots, a_n, a$, where $a = a_1 - q_1 a_2$. Then $c_1, \ldots, c_{k+1}$ is the remainder sequence of $(a_1', a_2', \ldots, a_n') \in S_{n,k}$. By the induction hypothesis, we have $c_i \geq F_n(k-i+1)$ for $i=1,2,\ldots,k+1$. It remains to show that $c_0 \geq F_n(k+1)$. For

$0 \le i \le k-n+1$, $c_i = q_i c_{i+1} + c_{j_i}$ where $j_i > i+1$ is the index associated with the remainder of c_i and c_{i+1} in the remainder sequence of $(a_1, a_2, \ldots, a_n)$. Note that the j_i's are distinct. Consider

$$(4.7) \qquad c_0 = \sum_{i=0}^{k-n+1} c_i - \sum_{i=0}^{k-n+1} c_{i+1} + c_{k-n+2}$$

$$= \sum_{i=0}^{k-n+1} (q_i c_{i+1} + c_{j_i}) - \sum_{i=0}^{k-n+1} c_{i+1} + c_{k-n+2}$$

$$= \sum_{i=0}^{k-n+1} (q_i - 1) c_{i+1} + \sum_{i=0}^{k-n+1} c_{j_i} + c_{k-n+2}$$

$$\ge \sum_{i=0}^{k-n+1} c_{j_i} + c_{k-n+2} \quad , \text{ since } q_i \ge 1 \ .$$

Since $c_0, c_1, \ldots, c_{k+1}$ is a decreasing sequence, the sum of any $k - n + 2$ distinct elements of the sequence is greater than or equal to the sum of the last $k - n + 2$ elements of the sequence. So

$$(4.8) \qquad \sum_{i=0}^{k-n+1} c_{j_i} \ge \sum_{i=n}^{k+1} c_i \ .$$

Substituting (4.8) into (4.7),

$$c_0 \ge \sum_{i=n}^{k+1} c_i + c_{k-n+2}$$

$$\ge \sum_{i=n}^{k+1} F_n(k-i+1) + F_n(n-1)$$

$$= \sum_{j=0}^{k-n+1} F_n(j) + F_n(n-1) \ , \quad j=k-i+1$$

$$= F_n(k+1) \ , \text{ by Theorem 4.7.}$$

This completes the proof. □

Theorem 4.9. Let $(a_1,\ldots,a_n) \in S_{n,m}$. Then $m < \{2n^2/(2n-1)\}(\ln a_1) + 2(n-1)$.

Proof. By Theorems 4.6 and 4.8, $a_1 \geq F_n(m) \geq \alpha^{m-2n+2}$, where α is the only positive real root of $C_n(x) = x^n - x^{n-1} - 1 = 0$. Taking natural logrithms, we get $\ln a_1 \geq (m-2n+2)(\ln \alpha)$. Since $C_n(1+1/n)$ $= (1+1/n)^n - (1+1/n)^{n-1} - 1 = \{(1+1/n)^n - 1\} - (1+1/n)^{n-1}$ $= (1/n)\{(1+1/n)^{n-1} + (1+1/n)^{n-2} + \ldots + 1\} - (1+1/n)^{n-1} <$ $(1/n)\{n(1+1/n)^{n-1}\} - (1+1/n)^{n-1} = 0$ and $C_n(2) = 2^n - 2^{n-1} - 1 > 0$ for $n \geq 2$, $\alpha > 1+1/n$. Hence, $\ln a_1 > (m-2n+2)\ln(1+1/n)$. Since $\ln(1+x) = \sum_{i=1}^{\infty} (-1)^{i+1}x^i/i > x - x^2/2$ for $x > 0$, $\ln(1+1/n) > 1/n - 1/2n^2 = (2n-1)/2n^2$. Therefore, $\ln a_1 > (m-2n+2)(2n-1)/2n^2$. Now this theorem becomes evident. □

Since the length of the quotient sequence is n less than the length of the remainder sequence, we have immediately the following corollary.

Corollary. Let s be the number of divisions required to compute the remainder sequence for $(a_1,\ldots,a_n) \in S_{n,m}$. Then $s \leq \{2n^2/(2n-1)\}(\ln a_1) + n - 1$.

Let $a_i = F_n(m-i+1)$, $1 \leq i \leq n$. Then the $i^{\underline{th}}$ multiplier sequence $x_{i,k}$, $k = 0, 1, \ldots, m$ of $a_1, \ldots, a_n$ satisfies the recurrence relation

$$x_{i,k} = -x_{i,k-n+1} + x_{i,k-n} \quad , \text{ for } k \geq n \; . \tag{4.9}$$

We will call (4.9) the n^{th} order multiplier recurrence relation. Let $C_n^*(x) = x^n + x - 1$. Then $C_n^*(x) = 0$ is the characteristic equation for the n^{th} order multiplier recurrence relation. Note that $C_n^*(x) = -x^n C_n(1/x)$. So the characteristic roots of $C_n^*(x) = 0$ are the inverses of the characteristic roots of $C_n(x) = 0$. Let $\beta_1, \ldots, \beta_n$ be the roots of $C_n^*(x) = 0$. Since $\beta_1, \ldots, \beta_n$ are distinct, $x_{i,k} = \sum_{j=1}^{n} c_{i,j}\beta_j^k$, where the $c_{i,j}$'s are constant for a given n. Considering the generating function $G_i(z)$ of the i^{th} multiplier sequence $x_{i,k}$, $k = 0, 1, 2, \ldots$, $G_i(z) = \sum_{k=0}^{\infty} x_{i,k}z^k = \sum_{k=0}^{n-1} x_{i,k}z^k$

$+ z^n \sum_{k=0}^{\infty} x_{i,k+n} z^k = \sum_{k=0}^{n-1} x_{i,k} z^k + z^n \sum_{k=0}^{\infty} (x_{i,k} - x_{i,k+1}) z^k$

$= \sum_{k=0}^{n-1} x_{i,k} z^k + z^n \sum_{k=0}^{\infty} x_{i,k} z^k - z^{n-1} \sum_{k=1}^{\infty} x_{i,k} z^k = \sum_{k=0}^{n-1} x_{i,k} z^k$

$+ z^n G_i(z) - z^{n-1} G_i(z) + x_{i,0} z^{n-1}$. Thus,

$$(4.10) \quad G_i(z) = \left(\sum_{k=0}^{n-1} x_{i,k} z^k + x_{i,0} z^{n-1} \right)\left(1 + z^{n-1} - z^n \right)^{-1}$$

$$= \left(\sum_{k=0}^{n-1} x_{i,k} z^k + x_{i,0} z^{n-1} \right) \prod_{k=1}^{n} \left(1 - \beta_k z \right)^{-1} .$$

From the definition of $G_i(z)$,

$$(4.11) \quad G_i(z) = \sum_{k=0}^{\infty} x_{i,k} z^k = \sum_{k=0}^{\infty} \left(\sum_{j=1}^{n} c_{i,j} \beta_j^k \right) z^k$$

$$= \sum_{j=1}^{n} c_{i,j} \sum_{k=0}^{\infty} \left(\beta_j z \right)^k = \sum_{j=1}^{n} c_{i,j} \left(1 - \beta_j z \right)^{-1}$$

$$= \left\{ \sum_{j=1}^{n} c_{i,j} \prod_{1 \le k \le n \ \& \ k \ne j} (1 - \beta_k z) \right\} \prod_{k=1}^{n} (1 - \beta_k z)^{-1}$$

Comparing (4.10) and (4.11), we get

$$(4.12)\qquad \sum_{j=1}^{n} c_{i,j} \prod_{1\le k\le n \;\&\; k\ne j} (1-\beta_k z) = \sum_{k=0}^{n-1} x_{i,k} z^k + x_{i,0} z^{n-1} .$$

Let $z = \beta_m^{-1}$. Multiplied by β_m^{n-1}, (4.12) becomes

$$(4.13)\qquad c_{i,m} \prod_{1\le k\le n \;\&\; k\ne m} (\beta_m-\beta_k) = \sum_{k=0}^{n-1} x_{i,k} \beta_m^{n-k-1} + x_{i,0} .$$

Since $\prod_{1\le k\le n \;\&\; k\ne m} (\beta_m-\beta_k) = C_n^{*\prime}(\beta_m) = n\beta_m^{n-1} + 1$ and $x_{i,i-1} = 1$ and $x_{i,k} = 0$ for $k < n - 1$ & $k \ne i - 1$,

$$(4.14)\qquad c_{i,m} = \left\{ \begin{array}{ll} (\beta_m^{n-1} + 1)(n\beta_m^{n-1} + 1)^{-1} & \text{if } i = 1 \\ \beta_m^{n-i} (n\beta_m^{n-1} + 1)^{-1} & \text{if } i > 1 \end{array} \right\} .$$

The rate of growth of the multiplier sequences depends on the maximum modulus of the roots of the characteristic equation $C_n^*(x) = x^n + x - 1 = 0$, for which we now derive a bound.

Theorem 4.10. If ρ is a root of $C_n^*(x) = x^n + x - 1 = 0$ then $|\rho| < 2^{1/(n-1)}$.

Proof. Since $2^{1/(n-1)} > 1$, the theorem is obviously true if $|\rho| \le 1$. Assume $|\rho| > 1$. Then $|\rho^{-1}| < 1$. Since $\rho^n + \rho - 1 = 0$, $\rho^{n-1} = \rho^n/\rho = (1-\rho)/\rho = \rho^{-1} - 1$. Thus, $|\rho|^{n-1} = |\rho^{-1} - 1| \le |\rho^{-1}| + 1 < 2$. Therefore, $|\rho| < 2^{1/(n-1)}$. □

Theorem 4.11. Let $c_{i,m}$ be defined by (4.14). Then $|c_{1,m}| < 1$ and $|c_{i,m}| < 2$ for $i > 1$.

Proof. Let $\rho = \gamma e^{i\theta} = a + bi$ be a complex root of $C_n^*(x) = x^n+x-1 = 0$. Then $\gamma^{2n} = |\rho^n|^2 = |1-\rho|^2 = (1 - \gamma \cos\theta)^2 + (\gamma \sin\theta)^2 = \gamma^2 + 1 - 2\gamma \cos\theta = \gamma^2 + 1 - 2a$. Thus, $a = -(\gamma^{2n} - \gamma^2 - 1)/2$.

By elementary calculus one can show that a has a maximum a^* at $\gamma^* = n^{-1/(2n-2)}$. Hence $a \le a^* \le \gamma^* = n^{-1/(2n-2)} < 1$ for $n \ge 2$ and

(4.15) $\quad |n-(n-1)\rho| \ge |n-(n-1)a| > n - (n-1) = 1 .$

For $i = 1$, $|c_{i,m}| = |(\beta_m^{n-1} + 1)(n\beta_m^{n-1} + 1)^{-1}| =$

$|(\beta_m^n + \beta_m)(n\beta_m^n + \beta_m)^{-1}| = | (n\beta_m^n + \beta_m)^{-1}| = | n(1-\beta_m) + \beta_m |^{-1}$

$= | n - (n-1)\beta_m |^{-1} < 1$. For $i > 1$, $|c_{i,m}| = | \beta_m^{n-i}(n\beta_m^{n-1} + 1)^{-1}|$

$= | \beta_m^{n-i+1} (n\beta_m^n + \beta_m)^{-1} | = |\beta_m^{n-i+1}| \cdot |n - (n-1)\beta_m|^{-1} < |\beta_m^{n-i+1}|$

$< 2^{(n-i+1)/(n-1)} \le 2 .$ □

Section 4.4. Analysis of the Algorithm

Table 4.1 shows the experimental probability p that an element in the quotient sequence of n non-negative integers $a_1, \ldots, a_n$ is equal to 1. Each entry in Table 4.1 is an average of 30 examples in which the a_i's are random integers 60 bits or less in length. p is larger than 90 percent when n is greater than or equal to 8.

Table 4.1

n	p
3	.62
4	.74
5	.82
8	.92
11	.94
15	.96

Because of the high probability that quotients in quotient sequences are equal to 1, we will analyze a "special case" of the algorithm LDSSBR. Let r be the rank of the coefficient matrix A, and let $r_1 = \min\{r, n-1\}$. We will assume that at the beginning of the $i^{\underline{th}}$ execution of step 3, $1 \le i \le r_1$, the elements in the first row of C are consecutive $(n-i+1)^{\underline{th}}$ order Fibonacci numbers.

Steps 3 to 8 form the major loop of LDSSBR. Note that at the beginning of the $i^{\underline{th}}$ execution of the loop C is an $(m+n_i)$ by n_i matrix, where $n_i = n - i + 1$ and B is an $(m+n_i)$-vector. First we

will derive a bound on the sizes of the elements of C during the $i^{\underline{th}}$ execution of the loop.

Let $c^0_{j,k}$ be the element in the $j^{\underline{th}}$ row and $k^{\underline{th}}$ column of C at the beginning of the $i^{\underline{th}}$ execution of the loop. We assume $c^0_{1,1}, \ldots, c^0_{1,n_i}$ are consecutive $n_i^{\underline{th}}$ order Fibonacci numbers, say $c^0_{1,i} = F_{n_i}(s-i)$ for some positive integer s and $i = 1, \ldots, n_i$. Let d_i be a bound on the elements of C during the $i^{\underline{th}}$ execution of the major loop. Then, for $1 \le i \le r_1$, the element in the $j^{\underline{th}}$ row and $k^{\underline{th}}$ column of C during the $i^{\underline{th}}$ execution is given by $c_{j,k} = \sum_{h=1}^{n_i} c^0_{j,h}\, x_{h,s_i}$ for some s_i, $0 \le s_i \le s$, where x_{h,s_i} is the $(s_i+1)^{\underline{th}}$ element of the $h^{\underline{th}}$ multiplier sequence of $c^0_{1,1}, \ldots, c^0_{1,n_i}$. Since $x_{h,s_i} = \sum_{t=1}^{n_i} e_{h,t}\, \beta_t^{s_i}$, where $\beta_1, \ldots, \beta_{n_i}$ are the roots of $C^*_{n_i}(x) = x^{n_i} + x - 1 = 0$ and $e_{h,1}, \ldots, e_{h,n_i}$ are given by (4.14), we have

$$
\begin{aligned}
(4.16) \qquad |c_{j,k}| &\le \sum_{h=1}^{n_i} |c^0_{j,h}| \sum_{t=1}^{n_i} |e_{h,t}||\beta_t|^{s_i} \\
&\le d_{i-1} \sum_{h=1}^{n_i} \sum_{t=1}^{n_i} |e_{h,t}||\beta_t|^{s_i} \\
&\le 2^{s_i/(n_i-1)}\, d_{i-1} \sum_{h=1}^{n_i} \sum_{t=1}^{n_i} |e_{h,t}| \text{ , by Theorem 4.10 ,} \\
&< 2^{s_i/(n_i-1)+1}\, n_i^2\, d_{i-1} \text{ , by Theorem 4.11 ,} \\
&< 2^{s/n_{i+1}+1}\, n_i^2\, d_{i-1} \text{ , since } n_{i+1}=n_i-1 .
\end{aligned}
$$

Obviously, $(c^0_{1,1}, c^0_{1,2}, \ldots, c^0_{1,n_i}) \in S_{n_i,s-1}$. By Theorem 4.9,

$s < \{ 2n_i^2(\ln c^0_{1,1})/(2n_i-1) + 2n_i - 1\} < n_i^2(\ln d_{i-1})/n_{i+1} + 2n_{i+1}$.

Thus, from (4.16), taking base β logarithms, we have $\log_\beta d_i <$

$\{ n_i^2(\ln \beta)(\log_\beta d_{i-1})/n_{i+1}^2 + 3 \} \log_\beta 2 + 2\log_\beta n_i + \log_\beta d_{i-1}$

$= n_i^2 (\ln 2)(\log_\beta d_{i-1})/n_{i+1}^2 + \log_\beta 8 + 2\log_\beta n_i + \log_\beta d_{i-1}$. Hence ,

(4.17) $\quad L(d_i) \le \{ (\ln 2)n_i^2/n_{i+1}^2 + 1 \} L(d_{i-1}) + 2L(n_i) + L(8)$

$\le (1+(\ln 2))n_i^2 L(d_{i-1})/n_{i+1}^2 + 2L(n_i) + L(8)$.

Inductively, (4.17) becomes

(4.18) $\quad L(d_i) \le \{1+(\ln 2)\}^i n_1^2 n_{i+1}^{-2} L(d_0) + \sum_{j=0}^{i-1} \{1+(\ln 2)\}^j n_{i-j+1}^2 \cdot$

$n_{i+1}^{-2} \{2L(n_{i-j}) + L(8)\}$

$= \{1+(\ln 2)\}^i n^2 (n-i)^{-2} L(d_0) + \sum_{j=0}^{i-1} \{1+(\ln 2)\}^j \cdot$

$(n-i+j)^2 (n-i)^{-2} \{2L(n-i+j+1) + L(8)\}$

$\le \{1+(\ln 2)\}^i n^2 L(d_0) + 5L(n+1)\, n^2 \sum_{j=0}^{i-1} \{1+(\ln 2)\}^j$

$\le \{1+(\ln 2)\}^i n^2 L(d_0) + 5(\ln 2)^{-1}\{1+(\ln 2)\}^i n^2 L(n+1)$

$\le \{1+(\ln 2)\}^i n^2 \{ L(d_0) + 8L(n+1) \}$.

For $i > r_1$, the $i^{\underline{th}}$ execution of the major loop of LDSSBR does not change the elements of the matrix C . Thus, for $i > r_1$,

(4.19) $L(d_i) \le \{1+(\ln 2)\}^{r_1} n^2 \{ L(d_0) + 8L(n+1)\}$

$\le \{1+(\ln 2)\}^{r} n^2 \{ L(d_0) + 8L(n+1) \}$.

At the beginning of the $i^{\underline{th}}$ execution of the major loop, where $1 \le i \le r$ if $r < n$ or $1 \le i < r$ if $r = n$, B is an $(m+n_i)$-vector. Let B_0 be the value of B at the beginning of the $i^{\underline{th}}$ execution of the major loop. Step 5 generates a sequence $B_0, B_1, \ldots, B_t$ for the vector B, where t is the integer such that $c^0_{1,j} = F_n(t+n_i-j)$ for $j = 1, \ldots, n_i$. The first call to subroutine VIERED in the repeat-loop in step 5 implicitly generates a sequence $q_0, q_1, \ldots, q_t$ such that $b_0 = \sum_{j=0}^{t} q_j F_{n_i}(t+n_i-j-1)$, where $q_j=[b_j/F_{n_i}(t+n_i-j-1)]$ and b_j is the first element of B_j for $j = 0, 1, \ldots, t$. Note that $|q_j| = \lfloor |b_j|/F_{n_i}(t+n_i-j-1) \rfloor \le \lfloor (F_{n_i}(t+n_i-j) - 1)/F_{n_i}(t+n_i-j-1) \rfloor$ ≤ 1, since $|b_j| < F_{n_i}(t+n_i-j)$ for $j = j, \ldots, t$. Therefore,

(4.20) $\sum_{j=0}^{t} |q_j| = |q_0| + t = \lfloor |b_0|/F_{n_i}(t+n_i-1) \rfloor + t \le |b_0| + t$

$< |b_0| + \{ 2n_i^2 (\ln d_i)/(2n_i-1) \} + n_i - 1$, by the corollary of Theorem 4.9.

Let e_i be a bound on the elements of B during the $i^{\underline{th}}$ execution of the major loop, $1 \le i \le r_1$. Then $e_i \le e_{i-1} + \sum_{j=0}^{t} |q_j| d_{i-1}$ $= e_{i-1} + d_{i-1} \sum_{j=0}^{t} |q_j| < e_{i-1} + d_{i-1} \{ e_{i-1} + n_i^2 (1+(\ln d_{i-1}))/n_{i+1} +$ $n_{i+1} \} < (d_{i-1}+1) \{ e_{i-1} + n_i^2 (1 + (\ln d_{i-1}))/n_{i+1} + n_{i+1} \}$ $< (d_{i-1}+1) \{ e_{i-1} + n_i^2 (1 + d_{i-1})/n_{i+1} + n_{i+1} \}$. Thus,

$$(4.21)\quad L(e_i) \le L(d_{i-1}+1) + L(e_{i-1}) + L(\lceil n_i^2(d_{i-1}+1)/n_{i+1}\rceil) + L(n_{i+1}) + 2$$

$$\le L(d_{i-1}+1) + L(e_{i-1}) + L(d_{i-1}+1) + 2L(n_i) + 3$$

$$\le L(e_{i-1}) + 2L(d_{i-1}) + 2L(n_i) + 5$$

$$\le L(e_0) + 2\sum_{j=0}^{i-1} L(d_j) + 2\sum_{j=1}^{i} L(n_j) + 5i \text{ , by induction ,}$$

$$\le L(e_0) + 2\sum_{j=0}^{i-1} \{1+(\ln 2)\}^j n^2 \{L(d_0) + 8L(n+1)\} + 2iL(n) + 5i \text{ , by (4.18) ,}$$

$$\le L(e_0) + 2(\ln 2)^{-1} \{1+(\ln 2)\}^i n^2 \{L(d_0) + 8L(n+1)\} + 2iL(n) + 5i \text{ .}$$

For the case that $r = n$, $e_r \le e_{r-1} + e_{r-1}d_{r-1}$. Hence it is clear that (4.21) holds for $i \le r$ whether or not $r = n$.

For $i > r$, the $i^{\underline{th}}$ execution of the major loop does not change the elements of B . Thus, for $i > r$,

$$(4.22)\quad L(e_i) \le L(e_0) + 2(\ln 2)^{-1} \{1+(\ln 2)\}^r n^2 \{ L(d_0) + 8L(n+1) \} + 2rL(n) + 5r \text{ .}$$

We are ready now to analyze the computing time of the algorithm LDSSBR. Let k_i be the number of executions of step i . Then the values of the k_i's are given by Table 4.2.

Let $t_{i,j}$ be the computing time of the $j^{\underline{th}}$ execution of step i , and let t_i be the computing time of step i . Then

$$(4.23)\quad t_1 \sim n + m \text{ ,}$$

Table 4.2

i	1	2	3	4	5	6	7	8
k_i	1	1	n	n	r	m	m	m

(4.24) $\quad t_2 \preceq n(m+n) + nL(d_0) + (m+n) \sim n(m+n+L(d_0))$.

By Theorem 3.28, Theorem 3.29, (4.18) and (4.19),

(4.25) $\quad t_3 = \sum_{j=1}^{k_3} t_{3,j} \preceq n \{ (m+n)nL(d_r) + n^2L(d_r) \} \sim (m+n)n^2L(d_r)$.

Obviously,

(4.26) $\quad t_4 = \sum_{j=1}^{n} t_{4,j} \sim n$.

The computing time of step 5 is codominant with the computing time of repeated execution of VIERED and MICINS. Let s_i be the number of executions of the repeat loop. Let $p_1, \ldots, p_{s_i}$ and $q_1, \ldots, q_{s_i-1}$ be the quotient sequences implicitly generated by the first and second subroutine calls to VIERED in the repeat loop, respectively. Then $q_j = 1$ for $j = 1, \ldots, s_i-1$ by assumption and $|p_1| \leq e_0$ and $|p_j| = 1$ for $j = 2, \ldots, s_i$ (see the derivation of (4.20)). Therefore, by Theorems 3.23 and 3.28, $t_{5,i} \preceq (m+n_i) \{ L(d_i) \sum_{j=1}^{s_i} L(p_j) + L(e_i) \}$
$+ (m+n_i) \{ L(d_i) \sum_{j=1}^{s_i-1} L(q_j) + L(d_i) \} + n_i(s_i-1)L(d_i)$
$\preceq (m+n_i) \{ L(d_i) \{ L(e_i) + s_i - 1 \} + L(e_i) \} + (m+n_i)s_iL(d_i) +$
$n_i(s_i-1)L(d_i) \preceq (m+n_i)L(d_i)L(e_i) + (m+n_i)s_iL(d_i) \sim (m+n_i)L(d_i) \cdot$
$\{ L(e_i) + s_i \}$. By the corollary of Theorem 4.9, we have

$t_{5,i} \preceq (m+n_i)L(d_i)\ \{\ L(e_i) + n_i^2\ (\ln d_i)/n_{i+1} + n_{i+1}\ \} \sim (m+n_i)L(d_i)\cdot$

$\{\ n_i L(d_i) + L(e_i)\ \}$. Thus,

(4.27) $\quad t_5 = \sum_{j=1}^{r} t_{5,j} \preceq r(m+n)L(d_r)\ \{\ nL(d_r) + L(e_r)\ \}$.

The computing times for steps 6 to 8 are obvious and given by

(4.28) $\quad t_6 \sim t_8 \sim m$, and

(4.29) $\quad t_7 \preceq mn$.

By (4.19), (4.22), (4.23), (4.24), (4.25), (4.26), (4.27), (4.28) and (4.29),

$$
\begin{aligned}
t_{LDSSBR}(A,b) &\preceq n^2(m+n)L(d_r) + r(m+n)L(d_r)\{nL(d_r) + L(e_r)\} \\
&\sim (m+n)L(d_r)\ \{\ n^2 + rn\ L(d_r) + rL(e_r)\ \} \\
&\preceq (m+n)n^2\{1+(\ln 2)\}^{r+1}\{\ L(|A|) + L(n)\ \}\cdot \\
&\quad \{\ n^2 + rn^3\{\ L(|A|) + L(n)\ \} + rL(|b|) + \\
&\quad rn^2\{1+(\ln 2)\}^r\{\ L(|A|) + L(n)\ \}\ \} \\
&\sim rn^4(m+n)\{1+(\ln 2)\}^r\{n+\{1+(\ln 2)\}^r\}\{L(|A|)+L(n)\}^2 \\
&\quad + rn^2(m+n)\ \{1+(\ln 2)\}^r\ \{\ L(|A|)+L(n)\ \}\ L(|b|)\ .
\end{aligned}
$$

CHAPTER 5. A MODIFICATION OF KANNAN AND BACHEM'S ALGORITHM

Section 5.1. Introduction

In this chapter we will present and analyze another algorithm, called LDSMKB, for solving systems of linear Diophantine equations. The basic ideas for the algorithm come from Kannan and Bachem [KAB78]. They used these ideas in computing the Smith and the Hermite normal forms of a non-singular square integral matrix A and showed that the lengths of the coefficients of A during the computation of the Hermite normal form of A can be bounded by a polynomial function of order $n^4L(n|A|)$, where n is the common row and column dimension of A.

Usual methods for computing the Hermite normal form of A transform successively the submatrix consisting of the first i rows of A into its Hermite normal form for $i=1,\ldots,n$. So elements common to row 1, ..., i and columns $i+1$, ..., n of A are zero after transforming the first i rows of A into a Hermite normal matrix.

Kannan and Bachem's method successively transforms the i by i principal minor of A into its Hermite normal form for $i=1,\ldots,n$. So columns $i+1$, ..., n of A remain unchanged before transforming the $i+1$ by $i+1$ principal minor of A. Preconditioning is required to ensure that all the principal minors of A are non-singular. During the $i^{\underline{th}}$ execution of the major loop in Kannan and Bachem's algorithm, which transforms the $i+1$ by $i+1$ principal minor into its Hermite normal form, two processes are involved, namely, elimination and normalization. Elimination refers to the process of forcing $A_{j,i+1}$,

the element in row j and column $i+1$ of A, to zero for $j=1,\ldots,i$ by a sequence of unimodular column transformations on A, that is, postmultiplication of A by a sequence of unimodular matrices. Note that any unimodular transformation is equivalent to a sequence of elementary column operations, since every unimodular matrix is equal to a product of some sequence of elementary matrices. Normalization refers to the process of making elements to the left of the diagonal non-negative and less than the diagonal elements to their right. The normalization order is from top to bottom and from left to right. The normalization order in the third execution of the major loop is illustrated by the following figure.

*	0	0	0
1	*	0	0
2	3	*	0
4	5	6	*

Each "*" represents a diagonal element. Each positive number denotes the order of the corresponding element in the normalization process.

Let A' be the matrix obtained from A whose i by i principal minor A'_i is the Hermite normal form of the i by i principal minor A_i of A. Then $A' = AU$ for some unimodular matrix U. Since A' is obtained from A by a sequence of unimodular transformation on the first i columns of A, $U = \begin{bmatrix} U_i & 0 \\ 0 & I \end{bmatrix}$ where U_i is an i by i unimodular matrix, and hence, $A'_i = A_i U_i$. Since A_i is non-singular,

$U_i = A_i^{-1} A_i' = \mathrm{adj}(A_i)\, A_i' / \det(A_i)$, and hence, $|U_i| \le i|\mathrm{adj}(A_i)||A_i'|/|\det(A_i)|$. Since A_i' is a Hermite matrix, every element in the $j^{\underline{th}}$ row of A_i' is non-negative and not greater than the diagonal element in the $j^{\underline{th}}$ row of A_i' for $j=1,\ldots,i$ and each diagonal element of A_i' is not greater than the product of all the diagonal elements of A_i' , which is the determinant of A_i' . Thus, $|A_i'| \le |\det(A_i')| = |\det(A_i U_i)| = |\det(A_i)||\det(U_i)| = |\det(A_i)|$, and hence, $|U_i| \le i|\mathrm{adj}(A_i)|$. Since every element of the adjoint of A_i is an $i-1$ by $i-1$ minor of A whose magnitude is bounded by $(i-1)!|A|^{i-1}$ (see Theorem 5.2 in Section 5.3), $|U_i| \le i\cdot(i-1)!|A|^{i-1} = i!|A|^{i-1} \le (i|A|)^{i-1}$. Since $A' = AU$, $|A'| \le n|A||U| = n|A||U_i| \le (n|A|)^i$. Therefore, the length of the norm of A' is bounded by $iL(n|A|)$.

Using this bound for the lengths of the elements of A' at the end of the $i^{\underline{th}}$ iteration, Kannan and Bachem derive a bound of order $n^4L(n|A|)$ on the sizes of the coefficients during the $(i+1)^{\underline{th}}$ iteration. Without modifying their algorithm one can derive a better bound of order $n^2L(n|A|)$ on the sizes of the coefficients during the $(i+1)^{\underline{th}}$ iteration by employing once again the fact that A_i' is the Hermite normal form of A_i , so that the product of all the diagonal elements of A_i' , being the magnitude of the determinant of A_i , is bounded by $(i|A|)^i$.

One can obtain an even better bound, of order $nL(n|A|)$, by changing the normalization order of Kannan and Bachem's algorithm. The new normalization order is from right to left and from top to bottom. A typical normalization order for the third execution of the major loop

of this modified Kannan and Bachem's algorithm is illustrated by the following figure.

$$\begin{matrix} * & 0 & 0 & 0 \\ 4 & * & 0 & 0 \\ 5 & 2 & * & 0 \\ 6 & 3 & 1 & * \end{matrix}$$

This modification enables us to derive a better bound because column j is normalized by using only the already normalized columns $j+1,\ldots,i+1$ to its right during the $i^{\underline{th}}$ iteration of the major loop of the modified Kannan and Bachem's algorithm. The derivation of this new bound is very similar to the one given in Section 5.3 for the algorithm LDSMKB.

In the algorithm LDSMKB we extend Kannan and Bachem's ideas with the modification described in the previous paragraph to the problem of solving a system of linear Diophantine equations $Ax = b$ for any $A \in Z(m,n)$ of rank r and $b \in Z^m$. In order to employ Kannan and Bachem's ideas for a square non-singular preconditioned matrix, the n by n identity matrix is adjoined to the bottom of A, and we call the new matrix $\bar{A}$. Since $\bar{A}$ is of rank n, there exists a non-singular submatrix A^* of $\bar{A}$ consisting of r linearly independent rows of A and $n-r$ rows of the n by n identity matrix. The major work for the algorithm LDSMKB is to transform A^* into a pseudo-Hermite matrix by applying a sequence of unimodular transformations to $\bar{A}$. A square non-singular matrix is a <u>pseudo-Hermite matrix</u>, or in <u>pseudo-Hermite form</u>, if it is lower-triangular and the absolute value of any

off-diagonal element is less than the absolute value of the diagonal element to its right. The transformation of A^* into a pseudo-Hermite matrix enables us to derive a very good bound, of order $n + rL(r|A|)$, on the lengths of the coefficients pertaining to $\bar{A}$.

Some helpful notations are introduced here before we present and analyze the algorithm LDSMKB. Let V be a vector. The $i^{\underline{th}}$ component of V is denoted by V_i. Let A be an m by n matrix. The $j^{\underline{th}}$ column of A is denoted by A_j. The element in the $i^{\underline{th}}$ row and $j^{\underline{th}}$ column of A is denoted by $A_{i,j}$. Let $i_1,\ldots,i_s$ and $j_1,\ldots,j_t$ be sequences of integers such that $1 \le i_k \le m$ and $1 \le j_h \le n$. Then the matrix consisting of the elements of A in rows $i_1,\ldots,i_s$ and columns $j_1,\ldots,j_t$ in that order is denoted by $A\begin{bmatrix} i_1,\ldots,i_s \\ j_1,\ldots,j_t \end{bmatrix}$. If $s = t$, its determinant is denoted by $A\begin{pmatrix} i_1,\ldots,i_s \\ j_1,\ldots,j_t \end{pmatrix}$.

Given an m by n matrix A, let $\bar{A}$ be the $m + n$ by n matrix $\begin{bmatrix} A \\ I \end{bmatrix}$, where I is the n by n identity matrix. Define inductively the <u>row-sequence</u> of A, $R = (i_1,\ldots,i_n)$, as follows. For $n = 1$, $R = (i)$ where i is the smallest positive integer such that $\bar{A}\begin{pmatrix} i \\ 1 \end{pmatrix} \neq 0$. For $n > 1$, let $(i_1,\ldots,i_{n-1})$ be the row-sequence of $A\begin{bmatrix} 1,\ldots, m \\ 1,\ldots,n-1 \end{bmatrix}$ and let i be the smallest positive integer such that $\bar{A}\begin{pmatrix} i_1,\ldots,i_{n-1},i \\ 1,\ldots\ldots\ldots,n \end{pmatrix} \neq 0$. Then $R = (i_1,\ldots,i_k,i,i_{k+1},\ldots,i_{n-1})$ where, with $i_0 = 0$, k is the largest integer, $0 \le k \le n-1$, such that $i_k < i$.

The submatrix A^* discussed above will be the matrix $\bar{A}\begin{bmatrix} i_1,\ldots,i_n \\ 1,\ldots,n \end{bmatrix}$.

where $(i_1,\ldots,i_n)$ is the row-sequence of A.

Section 5.2. The Algorithm

In this section we will describe the algorithm LDSMKB and prove that the algorithm is valid.

The algorithm has two inputs and two outputs. The inputs are the coefficient matrix A, an integral m by n matrix, and the right-hand-side b, an integral m-vector, of the linear Diophantine system to be solved. The outputs are a particular solution x^* of the Diophantine system $Ax = b$ and a basis N of the solution module of the homogeneous Diophantine system $Ax = 0$ if the Diophantine system $Ax = b$ is consistent. Otherwise, the null list is returned for x^* and N.

The algorithm begins by constructing the $m + n$ by n matrix C whose initial value is $\bar{A} = \begin{bmatrix} A \\ I \end{bmatrix}$, where I is the n by n identity matrix, and the $(m+n)$-vector B whose initial value is $\bar{b} = \begin{bmatrix} -b \\ 0 \end{bmatrix}$, where 0 is the zero n-vector.

Let $R_h = (i_1,\ldots,i_h)$ be the row-sequence of the submatrix A^h consisting of the first h columns of A. Let D^h denote the square non-singular matrix $\bar{A}\begin{bmatrix} i_1,\ldots,i_h \\ 1,\ldots,h \end{bmatrix}$. Let r_h be the rank of A^h. Then the algorithm computes $R_1 = (i_1)$, where i_1 is the row index of the leading non-zero element of $\bar{A}_1$, that is, the smallest positive integer such that $\bar{A}\begin{pmatrix} i_1 \\ 1 \end{pmatrix} \neq 0$. If $i_1 > m$, which implies that A_1 is a zero vector, then $r_1 = 0$; otherwise, $r_1 = 1$. We claim that if $R = (j_1,\ldots,j_n)$ is the row-sequence of an m by n matrix of rank r, then $j_1 < j_2 < \ldots < j_n$ and $j_h > m$ for $r < h \leq n$. This is obviously true for R_1. Note that $C\begin{bmatrix} i_1 \\ 1 \end{bmatrix}$ is obviously a pseudo-

Hermite matrix.

Now the algorithm enters a loop which computes r_{k+1} and R_{k+1} and transforms D^{k+1} into a pseudo-Hermite matrix for $k=1,\ldots,n-1$ inductively. At the beginning of the $k^{\underline{th}}$ iteration $C\begin{bmatrix} i_1,\ldots,i_k \\ 1,\ldots,k \end{bmatrix}$ is in pseudo-Hermite form where $R_k = (i_1,\ldots,i_k)$ is the row-sequence of A^k. To transform D^{k+1} into a pseudo-Hermite matrix one can first eliminate $C_{i_j,k+1}$ by applying a unimodular transformation to columns C_j and C_{k+1} for $j=1,\ldots,k$ in that order and then normalizing elements to the left of the diagonal of $C\begin{bmatrix} i_1,\ldots,i_k \\ 1,\ldots,k \end{bmatrix}$ from right to left and from top to bottom.

However, a slight modification can be made to improve the efficiency. The algorithm LDSMKB first finds the row index j of the leading non-zero element of C_{k+1}. If j is in $\{i_1,\ldots,i_k\}$, say $j = i_h$, then the algorithm eliminates $C_{j,k+1}$ by applying a unimodular transformation to C_h and C_{k+1} and repeats the above process. Eventually, j will be different from any index in $\{i_1,\ldots,i_k\}$, since the j's found in the above process are increasing and C_{k+1} is linearly independent of C_1 through C_k. Let j^* be the last j found. Then j^* is the smallest positive integer such that $C\begin{pmatrix} i_1,\ldots,i_k,j^* \\ 1,\ldots,\ldots,k+1 \end{pmatrix} \neq 0$. If $j^* > i_k$, then R_{k+1} is obviously the sequence $(i_1,\ldots,i_k,j^*)$. If, with $i_0 = 0$, h is the largest integer, $0 \leq h \leq k-1$, such that $i_h < j^* < i_{h+1}$, then $R_{k+1} = (i_1,\ldots,i_h,j^*,i_{h+1},\ldots,i_k)$. Such an h exists by the hypothesis that $i_1 < i_2 < \ldots < i_k$. Obviously, R_{k+1} is also an increasing sequence. If $j^* \leq m$, then A_{k+1} is linearly independent of $A_1,\ldots,A_k$, and

hence $r_{k+1} = r_k + 1$. Otherwise, $r_{k+1} = r_k$. Thus if $R_{k+1} = (i'_1,\ldots,i'_{k+1})$ then $i'_h > m$ for $h > r_{k+1}$. For the case that $i_h < j^* < i_{h+1}$, the algorithm rotates columns $h+1,\ldots,k+1$ so that column $k+1$ becomes the $(h+1)^{\underline{th}}$ column, and then the matrix $C\begin{bmatrix} i'_1,\ldots,i'_{k+1} \\ 1,\ldots,k+1 \end{bmatrix}$ is lower-triangular. Since now columns $h+2,\ldots,k+1$ of C were columns $h+1,\ldots,k$ of C at the beginning of the $k^{\underline{th}}$ iteration, whose elements were not changed during the elimination, the normalization can proceed from column $h+1$, rather than from column k , to column 1 .

Let $\bar{A}'$ be the final value of C after finishing the loop $n-1$ times, and let $R_n = (i_1,\ldots,i_n)$. Then a particular solution can be obtained, if the system is consistent, by reducing B_{i_h} for $h=1,\ldots,r_n$ through applications of the algorithm VIERED to B and $\bar{A}'_h$ with respect to their $i_h^{\underline{th}}$ elements. Let $(b'_1,\ldots,b'_{m+n})$ be the final value of B . If $b'_i = 0$ for $i=1,\ldots,m$, then the Diophantine system $Ax = b$ is consistent and the algorithm returns $(b'_{m+1},\ldots,b'_{m+n})$ and $\bar{A}'\begin{bmatrix} m+1,\ldots,m+n \\ r_n+1,\ldots,n \end{bmatrix}$ for x^* and N respectively. Otherwise, the Diophantine system is inconsistent and the algorithm returns the null list for x^* and N .

Following is the algorithm description of LDSMKB. The validity proof of LDSMKB follows the algorithm description.

LDSMKB(A,b;x*,N)

[Linear Diophantine system solution, modified Kannan and Bachem algorithm. A is an m by n integral matrix. b is an integral

m-vector. If the Diophantine system Ax=b is consistent, then x* is a particular solution and N is a list of basis vectors of the solution module of Ax=0. Otherwise, x* and N are null lists. A and b are modified.]

safe $c,C_1,C',C_1',C^*,h,i,j,k,m,N,n,r,R',x^*$.

(1) [Adjoin identity matrix to A and zero vector to -b.]
n←LENGTH(A); C←MIAIM(A); B←VIAZ(VINEG(b),n).

(2) [Initialize.] m←LENGTH(b); C_1←FIRST(C); j←0; repeat { j←j+1; ADV(C_1;c,C_1) } until c≠0; R←COMP(j,()); if j≤m then r←1 else r←0; k←1; if n=1 then go to 5.

(3) [Eliminate column k+1 and augment row-sequence.]
C*←REDUCT(C,k); C_1^*←FIRST(C*); C'←C; R'←R; for h=1,...,k+1 do { if h≤k then ADV(R';i,R') else i←m+n+1; C_1'←C_1^*; j←0; repeat { j←j+1; ADV(C_1';c,C_1') } until c≠0; if j≥i then { if j=i then { C_1←FIRST(C'); VIUT(C_1,C_1^*,i;C_1,C_1^*); SFIRST(C',C_1) }; C'←RED(C') } else { SFIRST(C*,C_1^*); C←LEROT(C,h,k+1); R←LEINST(R,h-1,j); if j≤m then r←r+1; go to 4 } }.

(4) [Normalize off-diagonal elements.] for j=h,h-1,...,1 do { C*←REDUCT(C,j-1); ADV(C*;T,C'); R'←REDUCT(R,j); while R'≠() do { ADV(C';C_1',C'); ADV(R';i,R'); T←VIERED(T,C_1',i) }; SFIRST(C*,T) }; k←k+1; if k<n then go to 3.

(5) [Check consistency of the system.] for j=1,...,r do { ADV(C;T,C); ADV(R;i,R); B←VIERED(B,T,i) }; j←0; repeat { j←j+1; ADV(B;c,B) } until j=m ∨ c≠0.

(6) [System consistent.] if c=0 then { $C' \leftarrow C$; while $C' \neq ()$ do { $C'_1 \leftarrow FIRST(C')$; $C'_1 \leftarrow REDUCT(C'_1, m)$; $SFIRST(C', C'_1)$; $C' \leftarrow RED(C')$ }; $x^* \leftarrow B$; $N \leftarrow C$; return }.

(7) [System inconsistent.] $x^* \leftarrow ()$; $N \leftarrow ()$; return □

Theorem 5.1. The algorithm LDSMKB is valid.

Proof. Let C be the $(m+n)$ by n matrix which is initially $\begin{bmatrix} A \\ I \end{bmatrix}$ and let B be the $(m+n)$-vector which is initially $\begin{bmatrix} -b \\ 0 \end{bmatrix}$. Let C' denote the matrix consisting of rows $1,\ldots,m$ of C, let C'' denote the matrix consisting of rows $m+1,\ldots,m+n$ of C, let B' denote the vector consisting of the first m components of B and let B'' denote the vector consisting of the last n components of B. Performing unimodular transformations on C will preserve (1) the unimodularity of C'', (2) the validity of the relation $C' = AC''$ and (3) the consistency of $C'x = B'$. The proof is same as that for elementary column operations as shown in Theorem 4.1. Let $\bar{C}'$ and $\bar{C}''$ be the final values of C' and C'' respectively. Obviously, C'' is unimodular and $C' = AC''$ when C is constructed in step 1. So $\bar{C}''$ is unimodular, $\bar{C}' = A\bar{C}''$ and $\bar{C}'x = b$ is consistent iff $Ax = b$ is consistent. Note that $\bar{C}'$ has the form $[\bar{C}'_1,\ldots,\bar{C}'_r,0,\ldots,0]$ where $r = \text{rank}(A)$. Therefore, by Theorem 2.7, $\{\bar{C}''_{r+1},\ldots,\bar{C}''_n\}$ is a basis of the solution module of $Ax = 0$.

Adding an integral multiple of a column of $\bar{C} = \begin{bmatrix} \bar{C}' \\ \bar{C}'' \end{bmatrix}$ to B will preserve (i) the validity of the relation $B' = AB'' - b$ and (ii) the consistency of $\bar{C}'x = b$ (see the proof of Theorem 4.1). Let $\bar{B}'$ and

$\bar{B}''$ be the final values of B' and B''. Obviously, $B' = AB'' - b$ when B is constructed in step 1. So $\bar{B}' = A\bar{B}'' - b$ and $\bar{C}'x = \bar{B}'$ is consistent iff $\bar{C}'x = b$ is consistent, and hence, iff $Ax = b$ is consistent.

If $\bar{B}' = 0$ then $A\bar{B}'' = b$. Therefore, $\bar{B}''$ is a particular solution of the system $Ax = b$. If $Ax = b$ is consistent, then $\bar{C}'x = \bar{B}'$ is consistent. Let $x^* = (x_1,\ldots,x_n)^T$ be a solution of $\bar{C}'x = \bar{B}'$. Then $\bar{B}' = \bar{C}'x^* = [\bar{C}'_1,\ldots,\bar{C}'_r,0,\ldots,0]\ (x_1,\ldots,x_n)^T$ $= \sum_{j=1}^{r} x_j\bar{C}'_j$. We claim $x_1 = \ldots = x_r = 0$. Let $1 \le h \le r$ and assume $x_1 = \ldots = x_{h-1} = 0$. Then $\bar{B}' = \sum_{j=h}^{r} x_j\bar{C}'_j$. Let $R = (i_1,\ldots,i_n)$ be the row-sequence of A and let $k = i_h$. Then $\bar{B}'_k = \sum_{j=h}^{r} x_j\bar{C}'_{k,j}$. But $\bar{C}'_{k,j} = 0$ for $j > h$ so $\bar{B}'_k = x_h\bar{C}'_{k,h}$. Since $|\bar{B}'_k| < |\bar{C}'_{k,h}|$ by virtue of step 5, $x_h = 0$. By induction on h, therefore, $x_1 = \ldots = x_r = 0$ and $\bar{B}' = 0$. This completes the proof. □

Section 5.3. Analysis of the Algorithm

In this section we will first derive bounds on the coefficients for the algorithm LDSMKB, then use these bounds to derive a bound on the computing time.

Theorem 5.2. Let A be an n by n matrix, and let d_i be the norm of the $i^{\underline{th}}$ row of A, i.e., $d_i = \max_{1\le j\le n} |A_{i,j}|$. Then $|\det(A)| \le n! \prod_{i=1}^{n} d_i$.

Proof. Since the determinant of A is a sum of $n!$ terms and each term is a product of n elements from n different rows of A, the theorem becomes evident. □

Steps 3 to 4 form the major loop of the algorithm LDSMKB. Let C^0 be the initial value of the matrix C. Let C^k, $1 \le k \le n-1$, be the value of C at the end of the $k^{\underline{th}}$ iteration of step 4. Theorem 5.3 gives bounds on the coefficients of C^k for $k=1,\ldots,n-1$.

Theorem 5.3. Let r_{k+1} and $(i_1,\ldots,i_{k+1})$ be the rank and the row-sequence of $A\begin{bmatrix}1,\ldots,n\\ 1,\ldots,k+1\end{bmatrix}$ respectively with $A \ne 0$. Then

(1) $C^k_{i,j} = C^0_{i,j}$ for $1\le i\le m+n$ & $k+2\le j\le n$,

(2) $C^k_{i_h,j} = 0$ for $1\le h<j\le k+1$,

(3) $|C^k_{i_h,j}| < |C^k_{i_h,h}|$ for $1\le j<h\le k+1$,

(4) $\prod_{h=1}^{k+1} |C^k_{i_h,h}| \le r_{k+1}!\ |A|^{r_{k+1}}$,

(5) $|C^k_{i,j}| \le (k+1)^2\, r_{k+1}!\ |A|^{r_{k+1}+1}$ for $1\le i\le m+n$ & $i \notin \{i_1,\ldots,i_{k+1}\}$ & $1\le j\le k+1$.

Proof. Let $k' = k+1$. During the first k iterations of the major loop no operations have been performed on columns $j > k'$, so (1) is true. Let D' and D denote the submatrices consisting of the first k' columns of C^k and C^0 respectively. Then $D' = DK$ for some k' by k' unimodular matrix K , since only unimodular transformations have been applied on the first k' columns of C . Let $H' = D' \begin{bmatrix} i_1,\ldots,i_{k'} \\ 1,\ldots,k' \end{bmatrix}$. By virtue of the algorithm, H' is a pseudo-Hermite matrix. So (2) and (3) are true. Let $H = D \begin{bmatrix} i_1,\ldots,i_{k'} \\ 1,\ldots,k' \end{bmatrix}$. Then $H' = HK$. Since H' is a triangular matrix, $\det(H') = \prod_{h=1}^{k'} D'_{i_h,h}$ $= \prod_{h=1}^{k'} C^k_{i_h,h}$. Also, $\det(H') = \det(HK) = \det(H)\det(K) = \pm \det(H)$. Therefore, $\prod_{h=1}^{k'} |C^k_{i_h,h}| = |\det(H)|$. Note that the h^{th} row of H is the i_h^{th} row of D , which consists of the first k' elements of row i_h of C^0 . Since $i_h \le m$ for $1 \le h \le r_{k'}$ and $i_h > m$ for $r_{k'} < h \le k'$, the norm of the h^{th} row of H is bounded by $|A|$ for $1 \le h \le r_{k'}$ and 1 for $r_{k'} < h \le k'$. By Theorem 5.2, $|\det(H)| \le r_{k'}!\ |A|^{r_{k'}}$. Thus, (4) is true. Since H is non-singular, H^{-1} exists. So $K = H^{-1} H' = \text{adj}(H)\, H' / \det(H)$, and hence, $|K| = |\text{adj}(H)\, H'| / |\det(H)| \le k' |\text{adj}(H)|\, |H'| / |\det(H)|$. Since H' is a pseudo-Hermite matrix, $|H'| = \max_{1\le h\le k'} |C^k_{i_h,h}| \le \prod_{h=1}^{k'} |C^k_{i_h,h}| = |\det(H)|$. So $|K| \le k' |\text{adj}(H)|$. Since every element of $\text{adj}(H)$ is the determinant of some k' by k' minor of H and at most $r_{k'}$ rows of H come from the coefficient matrix A and the rest come from the identity matrix I , $|\text{adj}(H)| \le r_{k'}!\ |A|^{r_{k'}}$.

Therefore, $|K| \le k' r_{k'}! |A|^{r_{k'}}$. Now $D' = DK$, hence $|D'| \le k' |D| |K| \le k' |A| (k' r_{k'}! |A|^{r_{k'}}) \le k'^2 r_{k'}! |A|^{r_{k'}+1}$. So (5) is true. □

Let $(i_1,\ldots,i_k)$ be the row-sequence of $A\begin{bmatrix}1,\ldots,m\\ 1,\ldots,k\end{bmatrix}$. During the $k^{\underline{th}}$ iteration of the major loop, step 3 eliminates elements in column $k+1$ of C by a sequence of unimodular transformations. The $j^{\underline{th}}$ iteration of the for-loop in step 3 will change only columns j and $k+1$ of C. Let $\bar{C}^k_j = (\bar{C}^k_{1,j},\ldots,\bar{C}^k_{m+n,j})$ and $\bar{C}^{k,j} = (\bar{C}^{k,j}_1,\ldots,\bar{C}^{k,j}_{m+n})$ be columns j and $k+1$ of C after the $j^{\underline{th}}$ iteration of the for-loop during the $k^{\underline{th}}$ iteration of the major loop. The following theorem gives bounds on the coefficients of $\bar{C}^k_j$ and $\bar{C}^{k,j}$.

Theorem 5.4. Let $d_h = |C^{k-1}_{i_h,h}|$ for $h=1,\ldots,k$. If the rank of $A\begin{bmatrix}1,\ldots,m\\ 1,\ldots,k\end{bmatrix}$ is r_k and $A \neq 0$, then

(1) $|\bar{C}^k_{i,j}|$, $|\bar{C}^{k,j}_i| \le |A| d_h \prod_{t=1}^{j} (2d_t)$ for $i=i_h \in \{i_{j+1},\ldots,i_k\}$,

(2) $|\bar{C}^k_{i,j}|$, $|\bar{C}^{k,j}_i| \le k^2 |A| r_k! |A|^{r_k+1} \prod_{t=1}^{j} (2d_t)$ for $i \notin \{i_1,\ldots,i_k\}$.

Proof by induction on j. Let $k' = k + 1$, and let $d_0 = k^2 r_k! |A|^{r_k+1}$. The theorem is obviously true for $k = 1$ if VIUT is not applied in the first iteration of the for-loop in step 3. If VIUT is applied in the first iteration of the for-loop in step 3, then $\bar{C}^k_{i,1} = u_1 C^{k-1}_{i,1} + v_1 C^{k-1}_{i,k'}$ and $\bar{C}^{k,1}_i = u_2 C^{k-1}_{i,1} + v_2 C^{k-1}_{i,k'}$, where u_1, v_1, u_2 and v_2

are integers obtained by applying IDEGCD to $C^{k-1}_{i_1,1}$ and $C^{k-1}_{i_1,k'}$. If $C^{k-1}_{i_1,k'} \neq 0$, then $|u_1| \leq |C^{k-1}_{i_1,k'}| = |C^{0}_{i_1,k'}| \leq |A|$ and $|v_1| \leq |C^{k-1}_{i_1,1}| = d_1$. Therefore, $|\bar{C}^{k}_{i,1}| \leq |A|\,|C^{k-1}_{i,1}| + d_1\,|A| \leq 2\,|A|\max\{|C^{k-1}_{i,1}|, d_1\}$. If $C^{k-1}_{i_1,k'} = 0$, then $|u_1| = |\mathrm{sign}(C^{k-1}_{i_1,1})| = 1$ and $|v_1| = |\mathrm{sign}(C^{k-1}_{i_1,k'})| = 0$. Again, $|\bar{C}^{k}_{i,1}| = |C^{k-1}_{i,1}| \leq 2\,|A|\max\{|C^{k-1}_{i,1}|, d_1\}$. If $i = i_h \in \{i_2,\ldots,i_k\}$, then $|C^{k-1}_{i,1}| < |C^{k-1}_{i_h,h}| = d_h$. Hence, $|\bar{C}^{k}_{i,1}| \leq 2|A|\max\{d_h, d_1\} \leq |A|\,d_h\,(2d_1)$. If $i \notin \{i_1,\ldots,i_k\}$, then $|C^{k-1}_{i,1}| \leq d_0$. Hence, $|\bar{C}^{k}_{i,1}| \leq |A|\,d_0\,(2d_1)$. So the theorem is true for $\bar{C}^{k}_{i,1}$. With a similar argument, the theorem is also true for $\bar{C}^{k,1}_{i}$.

Now assume the induction hypothesis is true for $j \geq 1$. The theorem is obviously true for $k = j + 1$ if VIUT is not applied in the $(j+1)^{\underline{th}}$ iteration of the for-loop in step 3. If VIUT is applied in the $(j+1)^{\underline{th}}$ iteration of the for-loop in step 3, then $\bar{C}^{k}_{i,j+1} = u_1\,C^{k-1}_{i,j+1} + v_1\,\bar{C}^{k,j}_{i}$ and $\bar{C}^{k,j+1}_{i} = u_2\,C^{k-1}_{i,j+1} + v_2\,\bar{C}^{k,j}_{i}$, where u_1, v_1, u_2 and v_2 are integers obtained by applying IDEGCD to $a_1 = C^{k-1}_{i_{j+1},j+1}$ and $a_2 = \bar{C}^{k,j}_{i_{j+1}}$. Let $d = |A|\,d_{j+1}\prod_{t=1}^{j}(2d_t)$. If $a_2 \neq 0$, then $|u_1| \leq |a_2| \leq d$ by induction hypothesis and $|v_1| \leq |a_1| = d_{j+1}$. If $a_2 = 0$, then $|u_1| = |\mathrm{sign}(a_1)| = 1$ and $|v_1| = |\mathrm{sign}(a_2)| = 0$. For either case, $|\bar{C}^{k}_{i,j+1}| \leq d\,|C^{k-1}_{i,j+1}| + d_{j+1}\,|\bar{C}^{k,j}_{i}|$. If $i = i_h \in \{i_{j+2},\ldots,i_k\}$, then $|C^{k-1}_{i,j+1}| < |C^{k-1}_{i_h,h}| = d_h$ and $|\bar{C}^{k,j}_{i}| \leq$

$|A|\ d_h \prod_{t=1}^{j} (2d_t)$ by induction hypothesis. Thus, $|\bar{C}^k_{i,j+1}| \le 2\ |A|\ d_h\ d_{j+1} \prod_{t=1}^{j} (2d_t) = |A|\ d_h \prod_{t=1}^{j+1} (2d_t)$. If $i \notin \{ i_1,\ldots,i_k \}$, then $|C^{k-1}_{i,j+1}| \le d_0$ by Theorem 5.3(5) and $|\bar{C}^{k,j}_i| \le |A|\ d_0 \prod_{t=1}^{j} (2d_t)$. Thus, $|\bar{C}^k_{i,j+1}| \le 2\ |A|\ d_0\ d_{j+1} \prod_{t=1}^{j} (2d_t) = |A|\ d_0 \prod_{t=1}^{j+1} (2d_t)$. So the theorem is true for $\bar{C}^k_{i,j+1}$. Similarly for $\bar{C}^{k,j+1}_i$. □

Corollary. During the k^{th} iteration of step 3 of the algorithm LDSMKB, if $A \neq 0$ then the norm of C is bounded by $2^k\ k^2\ r_k!^2\ |A|^{2r_k+2}$.

Proof. The h^{th} column of C, $1 \le h \le k$, is changed only once during the k^{th} iteration of step 3, and $\bar{C}^k_h$ is the new value of the h^{th} column of C. By Theorem 5.4 and the fact that $|\bar{C}^k_{i_h,h}| = \gcd(C^{k-1}_{i_h,h}, \bar{C}^{k,h-1}_{i_h}) \le |C^{k-1}_{i_h,h}| = d_h$, $|C| \le k^2\ r_k!\ |A|^{r_k+2} \prod_{t=1}^{k} (2d_t)$ during the k^{th} iteration of step 3. Since $\prod_{t=1}^{k} (2d_t) = 2^k \prod_{t=1}^{k} |C^{k-1}_{i_t,t}| \le 2^k\ r_k!\ |A|^{r_k}$ by Theorem 5.3(4), $|C| \le 2^k\ k^2\ r_k!^2\ |A|^{2r_k+2}$. □

Let $R_{k+1} = (i_1,\ldots,i_{k+1})$ be the row-sequence of $A\begin{bmatrix} 1,\ldots,m \\ 1,\ldots,k+1 \end{bmatrix}$. Let $\tilde{C}^{k,p}_j = (\tilde{C}^{k,p}_{1,j},\ldots,\tilde{C}^{k,p}_{m+n,j})$ be the j^{th} column of C after the i_p^{th} element, $j < p \le k+1$, of the j^{th} column of C is normalized during the k^{th} iteration of the major loop. The following theorem gives bounds on $\tilde{C}^{k,p}_{i,j}$.

Theorem 5.5. Let $d_h = |C^k_{i_h,h}|$ for $h = 1,\ldots,k+1$. If $A \neq 0$ then

$$(1)\quad |\tilde{C}^{k,p}_{i,j}| \le 2^{k+1}\,(k+1)^2\, r_{k+1}!^2\, |A|^{2(r_{k+1}+1)}\, d_h \prod_{t=j+2}^{p} (2d_t)$$

$$\text{if } i = i_h \in \{ i_{p+1},\ldots,i_{k+1} \},$$

$$(2)\quad |\tilde{C}^{k,p}_{i,j}| \le 2^{k+1}\,(k+1)^4\, r_{k+1}!^3\, |A|^{3(r_{k+1}+1)} \prod_{t=j+2}^{p} (2d_t)$$

$$\text{if } i \notin \{ i_1,\ldots,i_{k+1} \}.$$

Proof by induction on p. Let $\tilde{C}^k_j = (\tilde{C}^k_{1,j},\ldots,\tilde{C}^k_{m+n,j})$ be the $j^{\underline{th}}$ column of C before the normalization of the $j^{\underline{th}}$ column begins. Let $d = 2^k\,(k+1)^2\, r_{k+1}!^2\, |A|^{2(r_{k+1}+1)}$. Then $|\tilde{C}^k_{i,j}| \le 2^k\, k^2\, r_k!^2\, |A|^{2r_k+2} \le d$ by the corollary of Theorem 5.4. Let $s = i_{j+1}$. Then $\tilde{C}^{k,j+1}_{i,j} = \tilde{C}^k_{i,j} - q\, C^k_{i,j+1}$, where $q = [\, \tilde{C}^k_{s,j} / C^k_{s,j+1} \,]$. Since $|\tilde{C}^k_{i,j}| \le d$ and $|q| \le |\tilde{C}^k_{s,j}| \le d$, $|\tilde{C}^{k,j+1}_{i,j}| \le d + d\,|C^k_{i,j+1}|$. If $i = i_h \in \{ i_{j+2},\ldots,i_{k+1} \}$, then $|C^k_{i,j+1}| < |C^k_{i,h}| = d_h$. Hence, $|\tilde{C}^{k,j+1}_{i,j}| \le d\,(1+d_h) \le 2d\, d_h$. If $i \notin \{ i_1,\ldots,i_{k+1} \}$, then by Theorem 5.3(5) $|\tilde{C}^{k,j+1}_{i,j}| \le 2d\,|C^k_{i,j+1}| \le 2d\,(k+1)^2\, r_{k+1}!\, |A|^{r_{k+1}+1} = 2d'$, where $d' = 2^k\,(k+1)^4\, r_{k+1}!^3\, |A|^{3(r_{k+1}+1)}$. Therefore, the theorem is true for $p = j+1$.

Now assume inductively that the theorem is true for $p \ge j+1$. Let $s = i_{p+1}$. Then $\tilde{C}^{k,p+1}_{i,j} = \tilde{C}^{k,p}_{i,j} - [\, \tilde{C}^{k,p}_{s,j} / C^k_{s,p+1} \,]\, C^k_{i,p+1}$ and hence $|\tilde{C}^{k,p+1}_{i,j}| \le |\tilde{C}^{k,p}_{i,j}| + |\tilde{C}^{k,p}_{s,j}|\,|C^k_{i,p+1}| \le |\tilde{C}^{k,p}_{i,j}| + 2d\, d_{p+1} \cdot \{\prod_{t=j+2}^{p} (2d_t)\}\, |C^k_{i,p+1}|$. If $i = i_h \in \{ i_{p+2},\ldots,i_{k+1} \}$, then $|\tilde{C}^{k,p}_{i,j}| \le 2d\, d_h \prod_{t=j+2}^{p} (2d_t)$ and $|C^k_{i,p+1}| < |C^k_{i,h}| = d_h$. Hence, $|\tilde{C}^{k,p+1}_{i,j}| \le 2d\, d_h\,(1+d_{p+1}) \prod_{t=j+2}^{p} (2d_t) \le 2d\, d_h \prod_{t=j+2}^{p+1} (2d_t)$. If

$i \notin \{ i_1,\ldots,i_{k+1} \}$, then $|\tilde{C}^{k,p}_{i,j}| \le 2d' \prod_{t=j+2}^{p} (2d_t)$ by induction hypothesis and $|C^{k}_{i,p+1}| \le (k+1)^2 \, r_{k+1}! \, |A|^{r_{k+1}+1}$ by Theorem 5.3(5). Hence, $|\tilde{C}^{k,p+1}_{i,j}| \le 2d' \, (1+d_{p+1}) \prod_{t=j+2}^{p} (2d_t) \le 2d' \prod_{t=j+2}^{p+1} (2d_t)$. Therefore, the theorem is true for $p+1$.

Corollary. During the $k^{\underline{th}}$ iteration of step 4 of the algorithm LDSMKB, if $A \neq 0$, $|C| \le 2^{2k} \, (k+1)^4 \, r_{k+1}!^4 \, |A|^{4(r_{k+1}+1)}$, where r_{k+1} is the rank of $A \begin{bmatrix} 1,\ldots, m \\ 1,\ldots,k+1 \end{bmatrix}$.

Proof. Setting $p = k + 1$ in Theorem 5.5, and using Theorem 5.3(4), $|C| \le 2d' \prod_{t=j+2}^{k+1} (2d_t) \le 2^k \, d' \prod_{t=1}^{k+1} d_t \le 2^k \, d' \, r_{k+1}! \, |A|^{r_{k+1}+1} = 2^{2k} \, (k+1)^4 \, r_{k+1}!^4 \, |A|^{4(r_{k+1}+1)}$. □

Since $k \le n - 1$, $r_k \le r_{k+1} \le \text{rank}(A) = r$, $r! \le r^{r+1}$ and $n^4 < 2^{2n+2}$ for $n > 0$, we have the following theorem immediately from the corollaries of Theorems 5.4 and 5.5.

Theorem 5.6. At any point of the algorithm LDSMKB, if $A \neq 0$, $|C| < 2^{4n} \, (r \, |A|)^{4(r+1)}$, where $r = \text{rank}(A)$.

The next theorem gives bounds on elements of the vector B.

Theorem 5.7. Let $(i_1,\ldots,i_n)$ be the row-sequence of A . Let C* be the final value of C . Let $e_h = |C^*_{i_h,h}|$ for $h = 1,\ldots,n$. Let B^k be the value of B right after the $k^{\underline{th}}$ application of the algorithm VIERED in step 6. Then, if $A \neq 0$, (1) $|B^k_i| < e_h$ if $i = i_h \in \{ i_1,\ldots,i_k \}$, (2) $|B^k_i| \le 2^k \, |b| \, e_h$ if $i = i_h \in$

$\{ i_{k+1},\ldots,i_n \}$, (3) $|B_i^k| \le 2^k\,|b|\,n^2\,(r|A|)^{r+1}$ if $i \notin \{ i_1,\ldots,i_n \}$.

Proof by induction on k. Let B^0 be the initial value of B, i.e., $B^0 = \begin{bmatrix} -b \\ 0 \end{bmatrix}$. For $k=1$, let $t = i_1$. Since $B^1 = B^0 - [B_t^0/C_{t,1}^*]\,C_1^*$, $B_i^1 = B_i^0 - [B_t^0/C_{t,1}^*]\,C_{i,1}^*$. If $i \in \{ i_1 \}$, then $B_i^1 = B_t^0 - [B_t^0/C_{t,1}^*]\,C_{t,1}^*$, and hence, $|B_i^1| < |C_{t,1}^*| = e_1$. Therefore, (1) is true for $k = 1$. Note that $|B_i^1| \le |B_i^0| + |B_t^0|\,|C_{i,1}^*| \le |b| + |b|\,|C_{i,1}^*|$. If $i = i_h \in \{ i_2,\ldots,i_n \}$, then $|C_{i,1}^*| < |C_{i,h}^*| = e_h$. Therefore, $|B_i^1| \le |b| + |b|\,e_h \le 2\,|b|\,e_h$. That is, (2) is true for $k = 1$. If $i \notin \{ i_1,\ldots,i_n \}$ then by Theorem 5.3(5), $|B_i^1| \le 2\,|b|\,|C_{i,1}^*| \le 2\,|b|\,n^2\,(r\,|A|)^{r+1}$ so (3) is true for $k = 1$.

Now assume the theorem is true for $k = p \ge 1$. Let $t = i_{p+1}$. Since $B^{p+1} = B^p - [\,B_t^p / C_{t,p+1}^*\,]\,C_{p+1}^*$, $B_i^{p+1} = B_i^p - [\,B_t^p / C_{t,p+1}^*\,]\,C_{i,p+1}^*$. If $i = i_h \in \{ i_1,\ldots,i_p \}$, then since $C_{i,p+1}^* = 0$, $|B_i^{p+1}| = |B_i^p| < e_h$ by induction hypothesis. If $i = i_{p+1} = t$, then $|B_i^{p+1}| = |B_t^p - [\,B_t^p / C_{t,p+1}^*\,]\,C_{t,p+1}^*| < |C_{t,p+1}^*| = e_{p+1}$. Therefore, (1) is true for $k = p + 1$. If $i = i_h \in \{ i_{p+2},\ldots,i_n \}$, then by induction hypothesis $|[B_t^p / C_{t,p+1}^*]| \le |B_t^p| / |C_{t,p+1}^*| \le (2^p\,|b|\,e_{p+1}) / e_{p+1} = 2^p\,|b|$. Thus, $|B_i^{p+1}| \le |B_i^p| + 2^p\,|b|\,|C_{i,p+1}^*| \le 2^p\,|b|\,e_h + 2^p\,|b|\,|C_{i,h}^*| = 2^p\,|b|\,e_h + 2^p\,|b|\,e_h = 2^{p+1}\,|b|\,e_h$. Therefore, (2) is true for $k = p+1$. If $i \notin \{ i_1,\ldots,i_n \}$, then $|B_i^p| \le 2^p\,|b|\,n^2\,(r|A|)^{r+1}$ by induction hypothesis (3) and $|C_{i,p+1}^*| \le n^2\,(r|A|)^{r+1}$ by Theorem 5.3(5). As shown in the proof for (2) $|[B_t^p/C_{t,p+1}^*]| \le 2^p\,|b|$. Therefore, $|B_i^{p+1}| \le 2^p\,|b|\,n^2\,(r|A|)^{r+1} + 2^p\,|b|\,n^2\,(r|A|)^{r+1} = 2^{p+1}\,|b|\,n^2(r|A|)^{r+1}$.

This completes the proof. □

Corollary. At any point of the algorithm LDSMKB, if $A \neq 0$, $|B| \leq 2^{2n+1} |b| (r|A|)^{r+1}$.

Proof. By Theorem 5.3(4), $e_h \leq (r|A|)^r$ for $h = 1,\ldots,n$. Therefore by Theorem 5.7, at any point of the algorithm LDSMKB, $|B| \leq 2^n |b| n^2 (r|A|)^{r+1} \leq 2^n n^2 |b| (r|A|)^{r+1} \leq 2^{2n+1} |b| (r|A|)^{r+1}$. □

Finally Theorem 5.8 gives a bound on the computing time of the algorithm LDSMKB.

Theorem 5.8. Let $A \in Z(m,n)$, with $A \neq 0$, and let $b \in Z^n$. Then $t_{LDSMKB}(A,b) \preceq n^3(m+n)\{n + r\,L(r|A|)\}^2 + r(m+n)\,L(|b|)\cdot\{n + r\,L(r|A|)\}$.

Proof. Let t_i be the computing time of step i. Obviously, $t_1 \preceq n(m+n) + m\,L(|b|)$ and $t_2 \sim m$ by Theorems 3.2, 3.17, 3.18 and 3.25. Let $t_{i,k}$, $3 \leq i \leq 4$ and $1 \leq k \leq n-1$, be the computing time of step i in the $k^{\underline{th}}$ iteration of the major loop. The computing time for VIUT is the most significant one in step 3. VIUT is called at most k times during the $k^{\underline{th}}$ iteration. Therefore, by Theorems 3.24 and 5.6, $t_{3,k} \preceq k(m+n)\{L(d^*)\}^2$, where $d^* = 2^{4n}(r|A|)^{4(r+1)}$. The computing time for VIERED is the most significant one in step 4. VIERED is called no more than k^2 times during the $k^{\underline{th}}$ iteration. So $t_{4,k} \preceq k^2(m+n)\{L(d^*)\}^2$ by Theorems 3.23 and 5.6. Thus, $t_3 + t_4 = \sum_{k=1}^{n-1}(t_{3,k} + t_{4,k}) \preceq \sum_{k=1}^{n-1}\{k(m+n) + k^2(m+n)\}\{L(d^*)\}^2 \sim n^3(m+n)\{L(d^*)\}^2$. t_5 is codominant with the computing time for the for-loop and the

repeat-loop in step 5. By Theorems 3.23 and 5.6 and the corollary of Theorem 5.7, $t_5 \preceq r(m+n)\ L(d^*)\ L(e^*) + (m+n)$, where $e^* = 2^{2n+1}\ |b|\ (r|A|)^{r+1}$. Obviously, $t_6 \sim m(n-r) + 1$ and $t_7 \sim 1$. Therefore, $t_{LDSMKB}(A,b) \preceq n^3(m+n)\ \{L(d^*)\}^2 + r(m+n)\ L(d^*)\ L(e^*)$. Since $L(d^*) \sim n + r\ L(r|A|)$ and $L(e^*) \sim n + L(|b|) + r\ L(r|A|) \sim L(d^*) + L(|b|)$, $t_{LDSMKB}(A,b) \preceq n^3(m+n)\ \{L(d^*)\}^2 + r(m+n)\ \{L(d^*)\}^2 + r(m+n)\ L(d^*)\ L(|b|) \sim n^3(m+n)\ \{n + r\ L(r|A|)\}^2 + r(m+n)\ L(|b|) \cdot \{n + r\ L(r|A|)\}$. □

CHAPTER 6. EMPIRICAL RESULTS

First we will show a typical example of coefficient growth in the algorithms LDSSBR and LDSMKB. Consider

$$A = \begin{bmatrix} -7 & -18 & -1 & -8 & 4 \\ -20 & -11 & 7 & 29 & -5 \\ -15 & 19 & -27 & -17 & 21 \\ -4 & 14 & 16 & -11 & -18 \end{bmatrix} \quad \text{and} \quad b = \begin{bmatrix} -24 \\ -6 \\ 21 \\ 8 \end{bmatrix}$$

as a randomly generated 4 by 5 matrix and 4-vector whose entries are five bits or less in length.

The matrix C and the vector B at the end of the $k^{\underline{th}}$ iteration of the major loop of LDSSBR are shown below. The first five columns are the matrix C . The last column is the vector B .

k = 0

-7	-18	-1	-8	4	24
-20	-11	7	29	-5	6
-15	19	-27	-17	21	-21
-4	14	16	-11	-18	-8
1	0	0	0	0	0
0	1	0	0	0	0
0	0	1	0	0	0
0	0	0	1	0	0
0	0	0	0	1	0

$k = 1$

1	0	0	0	0	0
-44	83	42	-19	14	-69
47	-107	25	-25	-20	30
58	-4	-23	47	-98	60
-1	0	-1	0	2	0
1	-1	0	0	-1	2
0	2	-1	0	0	0
-2	2	1	-1	2	-2
-1	0	0	-2	3	-1

$k = 2$

1	0	0	0	0	0
-44	1	0	0	0	0
47	72	-653	2	83	-57
58	-313	239	-293	564	154
-1	5	7	6	-13	-2
1	-2	-4	-3	6	2
0	-3	15	-4	3	2
-2	5	-1	5	-10	-2
-1	9	-4	6	-15	-4

k = 3

1	0	0	0	0	0
-44	1	0	0	0	0
47	72	1	0	0	0
58	-313	38030	139981	-108167	23616
-1	5	-775	-2852	2204	-481
1	-2	352	1293	-1000	219
0	-3	317	1154	-896	193
-2	5	-648	-2385	1843	-402
-1	9	-989	-3654	2819	-618

k = 4

1	0	0	0	0	0
-44	1	0	0	0	0
47	72	1	0	0	0
58	-313	38030	1	0	0
-1	5	-775	-8612	25840	-6129
1	-2	352	40350	-121069	28718
0	-3	317	199388	-598258	141903
-2	5	-648	-2229	6688	-1586
-1	9	-989	211893	-635779	150803

$\{ (25840, -121069, -598258, 6688, -635779)^T \}$ is the basis of the solution module of $Ax = 0$. $(-6129, 28718, 141903, -1586, 150803)^T$ is the particular solution of $Ax = b$.

The matrix C and the vector B at the end of the $k^{\underline{th}}$ iteration of the major loop of LDSMKB are shown below.

k = 0

-7	-18	-1	-8	4	24
-20	-11	7	29	-5	6
-15	19	-27	-17	21	-21
-4	14	16	-11	-18	-8
1	0	0	0	0	0
0	1	0	0	0	0
0	0	1	0	0	0
0	0	0	1	0	0
0	0	0	0	1	0

k = 1

1	0	-1	-8	4	24
-78	-283	7	29	-5	6
-113	-403	-27	-17	21	-21
-48	-170	16	-11	-18	-8
5	18	0	0	0	0
-2	-7	0	0	0	0
0	0	1	0	0	0
0	0	0	1	0	0
0	0	0	0	1	0

$k = 2$

1	0	0	-8	4	24
0	1	0	29	-5	6
1126	157	11007	-17	21	-21
-310	-42	-3014	-11	-18	-8
-14	-2	-137	0	0	0
7	1	69	0	0	0
-29	-4	-283	0	0	0
0	0	0	1	0	0
0	0	0	0	1	0

$k = 3$

1	0	0	0	4	24
0	1	0	0	-5	6
0	0	1	0	21	-21
457006	3871	-133660	-635779	-18	-8
-9796	-83	2865	13628	0	0
6493	55	-1899	-9033	0	0
14993	127	-4385	-20858	0	0
-7912	-67	2314	11007	0	0
0	0	0	0	1	0

k = 4

1	0	0	0	0	24
0	1	0	0	0	6
0	0	1	0	0	-21
0	0	0	1	0	-8
9636	3369	-8415	-8612	25840	0
-45148	-15785	39427	40350	-121069	0
-223097	-78001	194827	199388	-598258	0
2494	872	-2178	-2229	6688	0
-237089	-82893	207046	211893	-635779	0

The basis of the solution module of $Ax = 0$ is $\{ (25840, -121069, -598258, 6688, -635779)^T \}$. The particular solution $(-497089, 2329029, 11508805, -128658, 12230604)^T$ is obtained by eliminating the first four elements of B.

One may observe that the first k diagonal elements of C after the $k^{\underline{th}}$ iteration are small integers. In this example they are all equal to one. An explanation is given below.

Theorem 6.1. (Cauchy-Binet Theorem) Let H and K be k by n and n by k matrices respectively. If $G = HK$, then $\det(G) = \sum_{1 \le i_1 < \ldots < i_k \le n} H\begin{pmatrix} 1, \ldots, k \\ i_1, \ldots, i_k \end{pmatrix} K\begin{pmatrix} i_1, \ldots, i_k \\ 1, \ldots, k \end{pmatrix}$.

Proof. See [KNU73], p. 37, or [GAN59], p. 9, for a proof. □

Let $Q_{k,n}$, with $1 \le k \le n$, be the set of all k-tuples $(i_1, \ldots, i_k)$ of integers such that $1 \le i_1 < \ldots < i_k \le n$. Let

$S \in Z(k,n)$ with $k \leq n$. If $S\begin{pmatrix} 1,\ldots,k \\ i_1,\ldots,i_k \end{pmatrix} = 0$ for all $\{ i_1,\ldots,i_k \} \in Q_{k,n}$, then let $d_k(S) = 0$; otherwise, let $d_k(S)$ be the greatest common divisor of all $S\begin{pmatrix} 1,\ldots,k \\ i_1,\ldots,i_k \end{pmatrix}$ such that $(i_1,\ldots,i_k) \in Q_{k,n}$. $d_k(S)$ is called the <u>k^{th} determinantal divisor</u> of S.

<u>Theorem 6.2.</u> Let $S \in Z(k,n)$. If $S' = SU$ and U is a unimodular matrix, then $d_k(S') = d_k(S)$.

<u>Proof.</u> Since $d_k(S) = 0$ iff $\text{rank}(S) < k$ and $\text{rank}(S') = \text{rank}(S)$, $d_k(S') = 0$ iff $d_k(S) = 0$. Let us assume $d_k(S) \neq 0$. For any $(j_1,\ldots,j_k) \in Q_{k,n}$, let $G = S'\begin{bmatrix} 1,\ldots,k \\ j_1,\ldots,j_k \end{bmatrix}$, let $H = S\begin{bmatrix} 1,\ldots,k \\ 1,\ldots,n \end{bmatrix}$, and let $K = U\begin{bmatrix} 1,\ldots,n \\ j_1,\ldots,j_k \end{bmatrix}$. Then, by Theorem 6.1, $S'\begin{pmatrix} 1,\ldots,k \\ j_1,\ldots,j_k \end{pmatrix} = \sum_{(i_1,\ldots,i_k)\in Q_{k,n}} S\begin{pmatrix} 1,\ldots,k \\ i_1,\ldots,i_k \end{pmatrix} U\begin{pmatrix} i_1,\ldots,i_k \\ j_1,\ldots,j_k \end{pmatrix}$. Since $d_k(S) \mid S\begin{pmatrix} 1,\ldots,k \\ i_1,\ldots,i_k \end{pmatrix}$ for any $(i_1,\ldots,i_k) \in Q_{k,n}$, $d_k(S) \mid S'\begin{pmatrix} 1,\ldots,k \\ j_1,\ldots,j_k \end{pmatrix}$ for any $(j_1,\ldots,j_k) \in Q_{k,n}$, and hence, $d_k(S) \mid d_k(S')$. Similarly, $d_k(S') \mid d_k(S)$, since $S = S'U^{-1}$ and U^{-1} is unimodular. Since $d_k(S)$ and $d_k(S')$ are positive, $d_k(S) = d_k(S')$. □

Suppose S' is lower-triangular and of full row rank. Then $S'\begin{bmatrix} 1,\ldots,k \\ 1,\ldots,k \end{bmatrix}$ is the only k by k minor of S' with non-zero determinant. Let $s_1,\ldots,s_k$ be the diagonal elements of S'. Then $\prod_{i=1}^{k} |s_i| = |S'\begin{pmatrix} 1,\ldots,k \\ 1,\ldots,k \end{pmatrix}| = d_k(S') = d_k(S)$ by Theorem 6.2. If $k < n$ then $d_k(S)$ is the greatest common divisor of the determinants

of all the k by k minors of S. Suppose S is a random matrix. Then $S\begin{pmatrix}1,\ldots\ldots,k\\1,\ldots,k-1,h\end{pmatrix}$ for $h = k, k+1, \ldots, n$ are random integers whose greatest common divisor, say $d_k'(S)$, in general will be small (see [KNU69], p. 301). Since $d_k(S) \leq d_k'(S)$, the s_i's in general are small.

To the algorithm LDSSBR, let S be the matrix $C^0\begin{bmatrix}i_1,\ldots,i_k\\1,\ldots\ldots,n\end{bmatrix}$ for the $k^{\underline{th}}$ iteration, where $C^0 = \begin{bmatrix}A\\I\end{bmatrix}$ and i_h is the $h^{\underline{th}}$ element of the row-sequence of A. To the algorithm LDSMKB, let S be the matrix $C^0\begin{bmatrix}i_1,\ldots,i_k\\1,\ldots,k+1\end{bmatrix}$. Then our observation follows from the above argument.

Finally, we will present several tables of empirical results which indicate some aspects of the performances of the algorithms LDSSBR and LDSMKB.

Symbols in these tables have the following meanings:

n - number of variables in the system,

m - number of equations in the system,

r - rank of the coefficient matrix,

d_0 - length, in bits, of the norm of the coefficient matrix,

e_0 - length, in bits, of the norm of the right-hand-side,

d - length, in bits, of the longest integer occurring in the matrix C during the computation,

e - length, in bits, of the longest integer occurring in the vector B during the computation,

$\bar{d}$ - length, in bits, of the longest integer in the basis obtained,

$\bar{e}$ - length, in bits, of the longest integer in the particular solution obtained,

t - UNIVAC 1110 execution time, in seconds,

T - VAX execution time, in seconds.

These tables are obtained by applying LDSSBR and LDSMKB to sets of randomly generated systems, that is, coefficients in these systems are randomly chosen from pre-specified intervals. For any given n, m, d_0 and e_0 in these tables, random systems were generated until a consistent one was found. About 40 percent of the random systems generated were inconsistent.

By comparing Tables 6.1 and 6.4, the ratio of VAX time to UNIVAC 1110 time is about 2.7 on average. This is probably, at least in part, because several basic SAC-2 subroutines have been written in assembly language for the UNIVAC but not for the VAX. By examining the tables we find that the ratio of the computing time of LDSSBR to that of LDSMKB ranged from 0.8 to 3.47. Ratios obtained from Tables 6.4 to 6.7 are displayed in Table 6.8. In all cases LDSSBR found smaller solutions than LDSMKB did, that is, the norms of the particular solutions obtained by LDSSBR were smaller than the norms of those obtained by LDSMKB. In all cases where $n - r > 1$ LDSSBR also obtained smaller solution module bases in the same sense.

Table 6.1

n	LDSSBR					LDSMKB				
	d	e	$\bar{d}$	$\bar{e}$	t	d	e	$\bar{d}$	$\bar{e}$	t
5	39	36	39	36	.33	81	48	39	46	.28
6	50	48	50	46	.69	101	60	50	59	.60
7	61	61	61	61	1.15	127	71	61	68	1.00
8	69	67	69	67	1.71	145	79	69	79	1.58
9	83	80	83	80	2.64	174	93	83	90	2.49
10	92	92	92	92	3.60	191	103	92	102	3.68
11	105	103	105	103	5.62	215	114	105	114	5.25
12	114	113	114	113	6.55	230	124	114	121	6.97
13	124	121	124	121	8.58	250	134	124	133	9.23
14	135	131	135	131	11.52	271	145	135	141	12.30
15	147	145	147	145	14.64	301	158	147	158	16.13
16	160	157	160	157	18.94	326	170	160	170	20.48
17	168	166	168	166	23.12	344	179	168	178	25.16
18	181	180	181	179	28.87	365	190	181	189	32.02
19	194	192	194	192	35.74	396	205	194	205	38.53
20	202	200	202	200	42.39	407	214	202	214	47.74

Notes: $m = r = n-1$ and $d_0 = e_0 = 10$.

Table 6.2

d_0	LDSSBR					LSDMKB				
	d	e	$\bar{d}$	$\bar{e}$	t	d	e	$\bar{d}$	$\bar{e}$	t
10	102	100	102	100	5.11	208	111	102	110	4.94
15	154	153	154	153	9.17	320	169	154	169	6.80
20	206	205	206	205	15.50	429	227	206	226	9.24
25	254	253	254	253	18.91	529	280	254	280	10.84
30	303	300	303	300	27.99	632	333	303	333	14.50
35	354	351	354	351	33.99	738	389	354	389	17.88
40	404	402	404	402	43.76	846	444	404	444	21.22
45	454	448	454	446	49.04	951	498	454	498	24.91
50	506	505	506	505	58.27	1058	555	506	553	29.51
55	554	553	554	553	67.46	1154	608	554	608	33.21
60	604	603	604	603	79.90	1260	665	604	665	38.39

Notes: $n = 11$, $m = r = 10$ and $e_0 = d_0$.

Table 6.3

	LDSSBR					LDSMKB				
m	d	e	$\bar{d}$	$\bar{e}$	t	d	e	$\bar{d}$	$\bar{e}$	t
1	60	57	14	9	1.68	179	117	60	117	1.95
2	74	72	28	26	3.96	298	177	120	174	3.43
3	85	85	37	33	6.62	416	230	172	230	6.01
4	102	102	60	59	12.50	539	300	241	300	9.28
5	122	121	82	79	18.75	658	355	296	347	12.78
6	143	142	119	118	26.56	776	418	358	416	16.93
7	176	176	166	165	35.84	899	479	419	478	22.59
8	212	211	208	207	46.03	1018	538	479	537	28.28
9	332	330	332	330	62.76	1144	602	541	602	34.27
10	604	603	604	603	79.96	1260	665	604	665	38.42

Notes: $n = 11$, $r = m$ and $d_0 = e_0 = 60$.

Table 6.4

n	LDSSBR					LDSMKB				
	d	e	$\bar{d}$	$\bar{e}$	T	d	e	$\bar{d}$	$\bar{e}$	T
5	41	35	41	35	.73	87	50	41	49	.86
6	50	50	50	50	1.56	107	60	50	58	1.56
7	60	59	60	59	2.45	128	71	60	71	2.87
8	71	66	71	66	3.89	150	81	71	81	4.50
9	82	82	82	82	6.02	170	92	82	91	6.64
10	91	89	91	89	8.83	180	101	91	100	9.48
11	100	97	100	97	12.24	204	110	100	110	13.45
12	115	115	115	114	15.64	236	126	115	126	19.46
13	122	119	122	119	21.82	248	132	122	132	24.69
14	136	134	136	134	28.34	277	148	136	147	34.77
15	149	149	149	149	38.69	306	160	149	160	45.54
16	159	158	159	158	46.79	320	170	159	170	57.75
17	170	166	170	166	61.86	343	181	170	179	73.87
18	177	173	177	173	79.20	359	188	177	187	91.17
19	191	191	191	191	102.81	386	202	191	202	115.25
20	206	205	206	205	125.21	420	217	206	216	141.43

Notes: $m = r = n-1$ and $d_0 = e_0 = 10$.

Table 6.5

n	LDSSBR					LDSMKB				
	d	e	$\bar{d}$	$\bar{e}$	T	d	e	$\bar{d}$	$\bar{e}$	T
5	77	76	77	76	1.97	164	96	77	96	1.35
6	100	99	100	99	4.88	216	121	100	121	2.73
7	119	118	119	118	7.56	260	138	119	137	4.69
8	139	136	139	136	12.41	288	159	139	159	7.61
9	161	159	161	159	19.61	329	180	161	178	12.14
10	184	181	184	181	29.59	380	204	184	204	17.77
11	202	201	202	200	38.26	416	222	202	222	25.01
12	224	222	224	222	48.46	465	245	224	245	36.83
13	247	246	247	246	69.39	512	268	247	267	52.95
14	263	261	263	261	84.72	541	284	263	282	69.46
15	288	285	288	285	119.13	590	308	288	305	96.30

Notes: $m = r = n-1$ and $d_0 = e_0 = 20$.

Table 6.6

n	LDSSBR					LDSMKB				
	d	e	$\bar{d}$	$\bar{e}$	T	d	e	$\bar{d}$	$\bar{e}$	T
5	161	160	161	160	7.89	343	201	161	201	2.60
6	199	196	199	196	16.43	432	239	199	239	5.24
7	239	238	239	238	25.74	514	278	239	278	10.46
8	280	279	280	279	39.10	599	320	280	320	16.83
9	320	317	320	317	58.64	670	360	320	355	26.84
10	361	360	361	360	85.93	761	401	361	401	43.70
11	403	403	403	403	120.43	832	443	403	441	65.24
12	444	443	444	443	159.74	918	482	444	481	96.20
13	485	484	485	484	206.63	999	525	485	524	136.48
14	526	524	526	523	281.03	1090	567	526	567	189.77
15	567	566	567	566	359.11	1169	607	567	605	261.65

Notes: $m = r = n-1$ and $d_0 = e_0 = 40$.

Table 6.7

n	LDSSBR					LDSMKB				
	d	e	$\bar{d}$	$\bar{e}$	T	d	e	$\bar{d}$	$\bar{e}$	T
5	239	238	239	238	15.89	531	299	239	298	4.58
6	298	297	298	297	29.76	652	358	298	358	9.13
7	358	356	358	356	49.15	771	418	358	418	17.60
8	417	415	417	413	76.44	886	477	417	477	31.31
9	483	480	483	480	118.03	1023	542	483	541	50.10
10	542	537	542	537	165.78	1143	601	542	601	80.51
11	601	599	601	599	229.39	1260	661	601	661	120.78
12	664	662	664	662	309.58	1387	723	664	717	181.54
13	723	722	723	722	411.54	1491	783	723	783	258.27
14	786	784	786	784	571.67	1623	846	786	843	364.17
15	849	847	849	847	761.56	1752	908	849	908	519.64

Notes: $m = r = n-1$ and $d_0 = e_0 = 60$.

Table 6.8

n \ d_0	10	20	40	60
5	0.85	1.46	3.03	3.47
6	1.00	1.79	3.14	3.26
7	0.85	1.61	2.46	2.79
8	0.86	1.63	2.32	2.44
9	0.91	1.62	2.18	2.36
10	0.93	1.67	1.97	2.06
11	0.91	1.53	1.85	1.90
12	0.80	1.32	1.66	1.71
13	0.88	1.31	1.51	1.59
14	0.82	1.22	1.48	1.57
15	0.85	1.24	1.37	1.47

CHAPTER 7. FUTURE STUDIES

The algorithm LDSSBR mainly depends on Rosser's algorithm for solving the linear Diophantine equation $a_1x_1 + \ldots + a_nx_n = c$ as described in Section 4.1. The extended Euclidean algorithm is a special case of Rosser's algorithm with $n = 2$ and $c = \gcd(a_1,\ldots,a_n)$. Detailed analyses of the Euclidean algorithm are found in [COL74] and [KNU69]. No analyses of the general case with $n > 2$ have been found. Studies should be done on the general case in order to get a better understanding of the complexity of the algorithm LDSSBR.

As observed in Chapter 6, LDSSBR in general obtained particular solutions with smaller norms than LDSMKB did. Given a basis and a particular solution, an optimal particular solution with respect to a definition of the norm of a vector can be obtained by adding an integral linear combination of the basis vectors to the particular solution. When the basis consists of only one vector, the problem is simple, since there is only one multiplier to be determined. However, when the basis consists of more than one basis vector, the problem becomes much more difficult. Algorithms for computing an optimal particular solution and the complexities of such algorithms can be investigated. If no polynomial time bounded algorithms can be found, algorithms for computing nearly optimal particular solutions could be sought.

REFERENCES

[BLA66] Blankinship, W. A., "Algorithm 288, solution of simultaneous linear Diophantine equations [F4]," CACM, v. 9, 1966, p. 514.

[BRA71a] Bradley, G. H., "Algorithms for Hermite and Smith normal matrices and linear Diophantine equations," Mathematics of Computation, v. 25, 1971, p. 897-907.

[BRA71b] Bradley, G. H., "Equivalent integer programs and canonical problems," Management Sciences, v. 17, 1971, p. 354-366.

[CAS59] Cassels, J. W. S., An Introduction to the Geometry of Numbers, Springer-Verlag, Reading, Berlin, 1959.

[CHG75] Charnes, A. and Granot, E., "Existence and representation of Diophantine and mixed Diophantine solutions to linear equations and inequalities," Discrete Mathematics, v. 11, 1975, p. 233-248.

[COL71] Collins, G. E., "The calculation of multivariate polynomial resultants," JACM, v. 18, 1971, p. 515-532.

[COL74] Collins, G. E., "The computing time of the Euclidean algorithm," SIAM Journal on Computing, v. 3, 1974, p. 1-10.

[COL79a] Collins, G. E., Algebraic Algorithms, to be published by Prentice-Hall.

[COL79b] Collins, G. E., The SAC-2 Manual, version 1, to appear.

[COS76a] Collins, G. E. and Schaller, S. C., "SAC-1 user's guide," Technical Report No. 269, Computer Sciences Department, University of Wisconsin, Madison, Wisconsin, 1976.

[COS76b] Collins, G. E. and Schaller, S. C., "SAC-1 implementation guide," Technical Report No. 268, Computer Sciences Department, University of Wisconsin, Madison, Wisconsin, 1976.

[FOR64] "A programming language for information processing on automatic data processing systems," CACM, v. 7, 1964, p. 591-625.

[FRU76] Frumkin, M. A., "An application of modular arithmetic to the construction of algorithms for solving systems of linear equations," Soviet Math. Dokl., v. 17, 1976, p. 1165-1168.

[GAN59] Gantmacher, F. R., The Theory of Matrices, Vol. 1, Chelsea Publishing Co., Reading, New York, 1959.

[HER51] Hermite, C., "Sur l'introduction des variables continues dans la théorie des nombres," J. Reine und Angewandte Mathematik, v. 41, 1851, p. 191-216.

[HOK71] Hoffman, K. and Kunze, R., Linear Algebra, Prentice-Hall, Reading, New Jersey, 1971.

[HUW70] Hurt, M. F. and Waid, C., "A generalized inverse which gives all the integral solutions to a system of linear equations," SIAM Journal on Applied Mathematics, v. 19, 1970, p. 547-550.

[KAB78] Kannan, R. and Bachem, A., "Polynomial algorithms for computing the Smith and Hermite normal forms of an integer matrix," SIAM Journal on Computing, to appear.

[KNU69] Knuth, D. E., The Art of Computer Programming, Vol. 2: Seminumerical Algorithms, Addison-Wesley, Reading, Massachusetts, 1969.

[KNU73] Knuth, D. E., The Art of Computer Programming, Vol. 1: Fundamental Algorithms, Addison-Wesley, Reading, Massachusetts, 1973.

[LOO76] Loos, R. G. K., "The algorithm description language ALDES (report)," SIGSAM Bulletin, v. 10, 1976, p. 14-38.

[MAR69] Marden, M., The Geometry of the Zeros of a Polynomial in a Complex Variable, 2nd ed., American Mathematical Society, Reading, New York, 1969.

[MCC73] McClellan, M. T., "The exact solution of systems of linear equations with polynomial coefficients," JACM, v. 20, 1973, p. 563-588.

[NEW72] Newman, M., Integral Matrices, Academic Press, Reading, New York and London, 1972.

[ROS41] Rosser, J. B., "A note on the linear Diophantine equation," American Mathematical Monthly, v. 48, 1941, p. 662-666.

[ROS52] Rosser, J. B., "A method of computing exact inverses of matrices with integer coefficients," J. Res. Nat. Bur. Standards, v. 49, 1952, p. 349-358.

[SMI61] Smith, H. J. S., "On systems of linear indeterminate equations and congruences," Philosophical Transactions, v. 151, 1861, p. 293-326.

[ZAP69] Zadeh, L. A. and Polak, E., System Theory, McGraw-Hill, Reading, New York, 1969.

INDEX OF ALGORITHMS