Lecture X
Linear Systems

Determinants and solving linear systems of equations are fundamental in algebraic computation. The basic facts of this topic are well-known when the underlying algebraic structure R is a field. Gaussian elimination is the main method in this case. But our main interest is when R is a domain D. Of course, we can still embed a determinant computation in the quotient field Q_D of D. But this method turns out to be inefficient, for reasons similar to those that argue against computing the GCD over $D[X]$ via the Euclidean algorithm for GCD over $Q_D[X]$. In this lecture, special techniques will be described for three problems:

(i) Computing determinants. The method to be described has similarities to the subresultant PRS algorithm, and in the modern form, is due to Bareiss [10, 11]. Bareiss noted that the method is known to Jordan. This method seems quite effective when D is the integers or univariate polynomials [51] (see also Sasaki and Murao [175]).

(ii)&(iii) Computing Hermite and Smith normal forms of integer matrices. These have applications to lattice-theoretic questions, solving linear Diophantine equations, and finitely generated Abelian groups.

The results for computing determinants apply to any domain D. For the Hermite and Smith normal forms, we describe the results for $D = \mathbb{Z}$ although the basic method could be extended to any UFD.

The set of $m \times n$ matrices with entries in D is denoted $D^{m\times n}$. The (i,j)th entry of a matrix M is denoted $(M)_{i,j}$ or $(M)_{ij}$.

§1. Sylvester's Identity

Bareiss' algorithm for computing determinants of matrices over D is based on a determinantal identity which we derive in this section. When applied to $D = \mathbb{Z}$, Bareiss' algorithm is polynomial-time.

It is instructive to understand why the usual Gaussian elimination is inadequate. Using fairly standard notations, suppose we initially have a matrix $M^{(1)}$ with L-bit integers. In the kth stage, we transform $M^{(k)}$ to $M^{(k+1)}$. The transformation applies to all entries of $M^{(k)}$ with index (i,j) where $i,j \geq k$. If the (i,j) entry of $M^{(k)}$ is $x_{ij}^{(k)}$, we have

$$x_{ij}^{(k+1)} \leftarrow x_{ij}^{(k)} - x_{kj}^{(k)} \frac{x_{i1}^{(k)}}{x_{k1}^{(k)}}.$$

The entries in $M^{(k+1)}$ are rational with bit size up to 4 times the bit sizes of entries in $M^{(k)}$. Thus after m steps, the entries can have bit size up to $4^m b$. It is an open problem if this exponential upper bound can be attained. Pivoting does not seem to help (see Exercise). Hence new methods are needed. Edmonds [62] appears to be the first to give a polynomial time solution for this problem.

Let $M \in D^{n\times n}$ with $(M)_{i,j} = x_{i,j}$. Note that x_{ij} may also be an indeterminate if D is suitably defined. The (i,j)-*cofactor* of M is denoted $[M]_{ij}$ and defined thus:

$$[M]_{ij} := (-1)^{i+j} \det M[i;j] \qquad (1)$$

where $M[i;j]$ is the matrix obtained by deleting the ith row and jth column of M. The *adjoint* of M is the matrix $\texttt{adj}(M)$ whose (i,j)th entry is the (j,i)-cofactor of M (note the transposed subscripts). For any $i,j = 1,\ldots,n$, it is easy to see that the sum

$$x_{i1}[M]_{j1} + x_{i2}[M]_{j2} + \cdots + x_{in}[M]_{jn}$$

is equal to $\det M$ if $i = j$, and equal to 0 if $i \neq j$. This immediately yields the following fundamental identity:

$$M \cdot \texttt{adj}(M) = \det(M) \cdot I. \tag{2}$$

Taking determinants on both sides, it follows that $\det(\texttt{adj} M) = \det(M)^{n-1}$. If $\det(M)$ is an invertible element of D, we infer that the inverse of M exists and is given by

$$M^{-1} = (\det(M))^{-1}\texttt{adj}(M). \tag{3}$$

Let ABC denote a triple matrix product of shape $m \times n \times n \times p$ (so B is an n-square matrix). For any b from domain D, we have the identity

$$bABC = A(bB)C. \tag{4}$$

This follows by looking at the typical element of $bABC$:

$$\begin{aligned}(bABC)_{rs} &= b\sum_{i=1}^{n} a_{ri}(BC)_{is} \\ &= b\sum_{i=1}^{n} a_{ri}\sum_{j=1}^{n}(B)_{ij}c_{js} \\ &= \sum_{i=1}^{n} a_{ri}\sum_{j=1}^{n}(bB)_{ij}c_{js} \\ &= \sum_{i=1}^{n} a_{ri}(bBC)_{is} \\ &= (A(bB)C)_{rs}.\end{aligned}$$

Next we express the matrix M in the form

$$M = \begin{bmatrix} A & B \\ C & X \end{bmatrix}$$

where A is $(k-1)$-square and X is $(n-k+1)$-square. For the following derivation, assume

$$\delta = \det A \neq 0.$$

Lemma 1

$$\delta^{n-k} \det M = \det(\delta X - C \cdot \texttt{adj}(A)B).$$

Proof. If we express

$$M = \begin{bmatrix} A & 0 \\ C & I \end{bmatrix} \begin{bmatrix} I & A^{-1}B \\ 0 & X - CA^{-1}B \end{bmatrix} \tag{5}$$

then we see that

$$\begin{aligned}\det M &= \delta \det(X - CA^{-1}B) \\ \delta^{n-k}\det M &= \det(\delta X - \delta CA^{-1}B) \\ &= \det(\delta X - C \cdot \texttt{adj}(A)B)\end{aligned}$$

where the last step exploits equations (3) and (4). **Q.E.D.**

But what is $\delta X - C\cdot \mathtt{adj}(A)B$? To see this, introduce the "(r,s)-bordered matrix of order k" defined to be

$$M_{r,s}^{(k)} := \left[\begin{array}{cccc|c} x_{1,1} & x_{1,2} & \cdots & x_{1,k-1} & x_{1,s} \\ x_{2,1} & x_{2,2} & \cdots & x_{2,k-1} & x_{2,s} \\ \cdots & & \cdots & & \cdots \\ x_{k-1,1} & x_{k-1,2} & \cdots & x_{k-1,k-1} & x_{k-1,s} \\ \hline x_{r,1} & x_{r,2} & \cdots & x_{r,k-1} & x_{r,s} \end{array}\right]$$

for $k \le \min\{r,s\}$. By definition, $M_{r,s}^{(1)}$ is the 1×1 matrix $[x_{r,s}]$. Also, define

$$x_{rs}^{(k)} := \det M_{rs}^{(k)}. \tag{6}$$

For instance, $M = M_{nn}^{(n)}$ and $\det M = x_{nn}^{(n)}$. Now let us look at a typical element of $\delta X - C\cdot\mathtt{adj}(A)B$: for $r \ge k$ and $s \ge k$, we have

$$\begin{aligned} (\delta X - C\cdot\mathtt{adj}(A)B)_{r-k+1,s-k+1} &= \delta x_{r,s} - (C\cdot\mathtt{adj}(A)B)_{r-k+1,s-k+1} \\ &= \delta x_{r,s} - \sum_{i=1}^{k-1} C_{r-k+1,i}\sum_{j=1}^{k-1}(\mathtt{adj}(A))_{ij}B_{j,s-k+1} \\ &= \delta x_{r,s} - \sum_{i=1}^{k-1} x_{r,i}\sum_{j=1}^{k-1}[A]_{ji}x_{j,s} \end{aligned}$$

where $[A]_{ji}$ is the (j,i)-cofactor of A. But the last expression can be seen to be equal to the determinant of the bordered matrix $M_{rs}^{(k)}$ (cf. exercise below giving the cofactors of $x_{r,i}x_{j,s}$). This proves:

Lemma 2

$$(\delta X - C\cdot\mathtt{adj}(A)B)_{rs} = x_{rs}^{(k)}.$$

Note that $\delta = x_{k-1,k-1}^{(k-1)}$. Combining the last two lemmas:

Lemma 3 (Sylvester's identity)

$$(x_{k-1,k-1}^{(k-1)})^{n-k}\det M = \det\begin{bmatrix} x_{k,k}^{(k)} & x_{k,k+1}^{(k)} & \cdots & x_{k,n}^{(k)} \\ x_{k+1,k}^{(k)} & x_{k+1,k+1}^{(k)} & \cdots & x_{k+1,n}^{(k)} \\ \vdots & & \ddots & \vdots \\ x_{n,k}^{(k)} & x_{n,k+1}^{(k)} & \cdots & x_{n,n}^{(k)} \end{bmatrix}.$$

Exercises

Exercise 1.1: (i) (Wilkinson 1961) With the notations for Gaussian elimination in the introduction, let us now assume total pivoting. Show that

$$|a_{ij}^{(k)}| \le k^{1/2}\left(2\cdot 3^{1/2}\cdot 4^{1/3}\cdot\cdots\cdot k^{1/(k-1)}\right)^{1/2}\|A\|_\infty.$$

NOTE: a rough estimate is that $\log\left(2\cdot 3^{1/2}\cdot 4^{1/3}\cdot\cdots\cdot k^{1/(k-1)}\right)$ is $O(\log^2 k)$. Thus the magnitude of the entries are polynomial.
(ii) Why is it not obvious that this leads to a polynomial time solution for integer matrices?
(iii) (Open) Construct examples with exponential bit sizes in the intermediate entries. (You may assume that no pivoting is needed.) □

Exercise 1.2: (Bareiss) Show that Sylvester's identity holds even when $x^{(k-1)}_{k-1,k-1} = 0$. HINT: perturb the singular submatrix. □

§2. Fraction-free Determinant Computation

Lemma 3 with $n-k=1$ is called the "first order" Sylvester identity. With the notations of the previous section, this identity for the matrix $M^{(k+1)}_{i,j}$ amounts to

$$x^{(k-1)}_{k-1,k-1}\det M^{(k+1)}_{ij} \quad=\quad \det\begin{bmatrix} x^{(k)}_{k,k} & x^{(k)}_{k,j} \\ x^{(k)}_{i,k} & x^{(k)}_{i,j}\end{bmatrix}.$$

Hence,

$$x^{(k+1)}_{ij} = \frac{x^{(k)}_{k,k}x^{(k)}_{i,j} - x^{(k)}_{i,k}x^{(k)}_{k,j}}{x^{(k-1)}_{k-1,k-1}}. \tag{7}$$

The important point is that the division by $x^{(k-1)}_{k-1,k-1}$ in this equation is exact (*i.e.*, with no remainder). Equation (7) is the defining step of the fraction-free Gaussian elimination algorithm of Bareiss (and Jordan):

```
Bareiss Algorithm
Input: $M$ an $n$-square matrix,
        assuming its principal minors $x^{(k)}_{kk}$ are all non-zero.
Output: The matrix entry $(M)_{n,n}$ contains the determinant of $M$.
        In general, for $i \ge k$, we have
        $(M)_{ik} = x^{(k)}_{ik}$,    $(M)_{ki} = x^{(k)}_{ki}$.

1.  $(M)_{0,0} \leftarrow 1$;   {Note: $(M)_{0,0}$ is a special variable.}
2.  for $k = 1,\ldots,n-1$ do
3.      for $i = k+1,\ldots,n$ do
4.          for $j = k+1,\ldots,n$ do
5.              $(M)_{ij} \leftarrow \frac{(M)_{ij}(M)_{kk}-(M)_{ik}(M)_{kj}}{(M)_{k-1,k-1}}$
```

The program structure of this algorithm amounts to a simple triple-loop, as in the standard Gaussian elimination. Its correctness is easily shown by induction on k, and by appeal to equation (7) (we leave this as an exercise).

In case the assumption about principal minors turns out to be false, this is easily detected and the algorithm may be aborted. Alternatively, it is not hard to add the code (between lines 2 and 3) to perform some kind of pivoting: say, if $(M)_{k-1,k-1} = 0$ and some $(M)_{k-1,i} \neq 0$ ($i = k,\ldots,n$) then we can exchange the ith column with the $k-1$st column. But we defer a more complete discussion of this to an extension of Bareiss' algorithm below.

The division in line 5 is exact since $(M)_{k-1,k-1} = x^{(k-1)}_{k-1,k-1}$. Hence all computed values remain inside the domain D.

This is an "in-place" algorithm that destroys the contents of the original matrix. But we can easily preserve the original matrix is desired. The output M has the following shape:

$$M = \begin{bmatrix} x^{(1)}_{1,1} & \cdots & & & & x^{(1)}_{1,n} \\ \vdots & \ddots & & & & \\ & & x^{(k)}_{kk} & x^{(k)}_{k,k+1} & \cdots & x^{(k)}_{k,n} \\ & & x^{(k)}_{k+1,k} & \ddots & & \\ & & \vdots & \ddots & & \\ x^{(1)}_{n,1} & & x^{(k)}_{n,k} & & & x^{(n)}_{n,n} \end{bmatrix}$$

In other words, for each $k = 1, \ldots, n$, there is an (rotated) L-shaped band in M that contains determinants of order k bordered matrices, as indicated.

In view of the definition of $x^{(k)}_{ij}$ (equation (6)) as subdeterminants of M, we obtain at once:

Lemma 4 *The intermediate values encountered in the algorithm have absolute values at most $n^n 2^{Ln}$ where 2^L bounds the absolute values of the entries of M.*

Since the algorithm takes $O(n^3)$ arithmetic steps, and each entry has at most $n(\log n + L)$ bits, we conclude that the bit-complexity of this method is

$$O(n^3 \mathrm{M}_B(n(\log n + L)))$$

where $\mathrm{M}_B(s)$ is the bit-complexity of multiplying two s-bit integers.

One can exploit the higher order Sylvester identities to obtain analogous algorithms. We will not explore this but see Bareiss [10] for an analysis of an algorithm exploiting the second order identity. We will instead describe two other extensions.

Extended Bareiss Algorithm. We extend the algorithm to matrices of arbitrary $m \times n$ shape and of general rank ρ. This can be formulated as follows:

> (*) *Given a matrix $A \in \mathbb{Z}^{m\times n}$, we want to compute its rank ρ, a permutation P of its rows, a permutation Q of its columns, and non-zero values $d_1, \ldots, d_\rho$, such that each d_i is the ith principal minor of PAQ.*

This will be needed for the Hermite normal form algorithm in §7.

Partially Processed Matrices. To describe the solution, it is useful to introduce some terminology. Let $A, M \in \mathbb{Z}^{m\times n}$. We say that M is a *partially processed version* of A if each (i,j)th entry

$(M)_{i,j}$ is an (i,j)-bordered determinant of A of some order $k = k(i,j)$. Clearly we have

$$1 \le k \le \min\{i,j\}.$$

If $k(i,j) = \min\{i,j\}$ (respectively, $k(i,j) = 1$), we say the (i,j)th entry is *fixed* (resp., *original*). We call $k(i,j)$ the *fixing order* of the (i,j)th entry. For instance, A is a partially processed version of itself, with every entry original. If every entry of M is fixed (for A), then M is said to be *completely processed* (for A). In this terminology, we can view Bareiss' algorithm on input A as trying to fix every entry of A. An operation of the form

$$(M)_{i,j} \leftarrow \frac{(M)_{k,k}(M)_{i,j} - (M)_{i,k}(M)_{k,j}}{(M)_{k-1,k-1}} \tag{8}$$

is called a *fixing step* for the (i,j)th entry. The fixing step is *valid* provided the fixing order of $(M)_{k,k}$, $(M)_{i,m}$, $(M)_{i,k}$ and $(M)_{k,m}$ is k and the fixing order of $(M)_{k-1,k-1}$ is $k-1$. If M is partially processed before a valid step for the (i,j)th entry, it remains partially processed after the step, with the fixing order of the (i,j) entry equal to $k+1$ (cf. equation (10)).

Let us return to problem (*). Suppose M is initialized to A and the row and column permutations P, Q are initially identities. Inductively, assume M is a partially processed version of PAQ. We proceed in stages. In stage s $(s = 1, \ldots, \rho)$ our goal is to ensure the following properties:
(i) Every entry in the sth principal submatrix of M is fixed.
(ii) The first s diagonal entries of M are non-zero.
(iii) To maintain an index $m_0 \ge s$ such that each row $i > m_0$ is known to be dependent on the first $s-1$ rows. Moreover, the entries in rows $s+1$ to m_0 are original. Initially, $m_0 = m$.

For $s \ge 1$, suppose that stage $s-1$ is completed. [For $s = 1$, the properties (i)-(iii) are vacuously true.] Observe that we can fix the first $s-1$ entries of the sth row of C in $O(s^2)$ fixing steps. Applying this observation once more, we can fix the first s entries in the sth column in $O(s^2)$ fixing steps. Thus goal (i) is attained. The problem is that the (s,s)th entry may be zero. We may go on to fix the first s entries of column j for $j = s+1, s+2, \ldots, n$. There are two cases:
(A) We find a column j which can be exchanged with column s to satisfy goal (ii). Then we exchange column j and column s, and update permutation Q. If $s = m_0$, we halt, else go to stage $s+1$.
(B) No such column exists. In this case we conclude that row s is dependent on the first $s-1$ rows. If $m_0 = s$, we stop, since $\rho = s-1$. Otherwise, we exchange row m_0 with row s, decrement m_0, update permutation P and continue with stage s.

This completes the description of our solution to problem (*). For reference, call this the *Extended Bareiss Algorithm.* It's correctness is clear. For its complexity, first note that each matrix entry is a minor of A and hence has bit-size at most $L' := \rho(\lg \rho + L)$ where $L = \lg \|A\|_\infty$. [Recall (§0.9) that $\lg = \log_2$.] Second, we use $O(mn\rho)$ arithmetic operations on integers of bit-size $O(\rho(\lg \rho + L))$. This is because the fixing step for each entry is applied at most $\rho + 1$ times. To see this, we verify two facts: (a) In case (A), when we exchange columns s and j, note that the first s entries in the columns between s and j have been fixed. This information is still valid after the column exchange. So we just need a Boolean flag at each entry to indicate whether it is fixed or not. (b) In case (B), note that after the exchange between rows s and row m_0, the entire row m_0 is already fixed and row s is original. Thus we have:

Theorem 5 *On input $A \in \mathbb{Z}^{m\times n}$ with rank $\rho \ge 1$, the Extended Bareiss Algorithm has bit complexity $O(mn\rho \mathrm{M}_B(L'))$ where $L' = \rho(\lg \rho + L)$ and $L = \lg \|A\|$.*

A Preprocessing Problem. We note another useful extension to Bareiss' algorithm. If $N \in \mathbb{Z}^{m\times(m-1)}$, let N_i $(i = 1, \ldots, m)$ be the submatrix of N with the ith row deleted. Consider this problem:

(**) *Given $N \in \mathbb{Z}^{m\times(m-1)}$, compute the m subdeterminants*

$$\det N_1, \ldots, \det N_m. \tag{9}$$

This is a "preprocessing problem" in the sense that once the $\det N_i$'s are available, for any given column $a = (a_1, \ldots, a_m)^T \in \mathbb{Z}^m$, if A is obtained by appending a to N, we can quickly compute $\det A$ as

$$\det A = \sum_{i=1}^{m} a_i(-1)^{i+m+1} \det N_i.$$

The solution is simple: let M be the $m \times m$ matrix obtained from N by appending a column

$$Z = (Z_1, \ldots, Z_m)^T$$

where the Z_i's are indeterminates. Let $X = (x_{i,j})_{i,j=1}^{m,m}$ be the output matrix when M is input to Bareiss' algorithm. The entries $x_{i,j}$ are computed as expected. But the entry $x_{i,m}$ in the last column is a linear function in $Z_1, \ldots, Z_i$, which we denote by $f_i(Z_1, \ldots, Z_i)$.

In the notations of §1, let $x_{i,j}^{(k)}$ denote the (i,j)-bordered determinant of M of order k $(i, j \geq k)$. Thus each output entry $x_{i,j}$ is equal $x_{i,j}^{(k)}$ where $k = \min\{i, j\}$. Let $\delta_k = x_{k,k}^{(k)}$. For example, the reader may verify that

$$\begin{aligned} f_1 &= Z_1, \\ f_2 &= \delta_1 Z_2 - x_{2,1} Z_1, \\ f_3 &= \delta_2 Z_3 - x_{3,2} Z_2 - \left(\frac{\delta_2 x_{3,1} - x_{3,2} x_{2,1}}{\delta_1}\right) Z_1. \end{aligned}$$

For any $(a_1, \ldots, a_m) \in \mathbb{Z}^m$, the value of $f_m(a_1, \ldots, a_m)$ yields the determinant of the matrix M where each Z_i is replaced by a_i. Thus, up to signs, the desired subdeterminants (9) may be read off from the coefficients of f_m.

What is the complexity of this procedure? The bit-sizes of entries in the first $m-1$ columns of M and the time to compute them are exactly as in Bareiss' algorithm. Consider the mth column. In stage $k+1$ $(k = 1, \ldots, m-1)$ of the outermost for-loop in Bareiss' algorithm, the entries of the column m that are computed are $x_{i,m}^{(k+1)}$ $(i = k+1, \ldots, m)$. We have

$$\begin{aligned} x_{i,m}^{(k+1)} &= \frac{x_{k,k}^{(k)} x_{i,m}^{(k)} - x_{i,k}^{(k)} x_{k,m}^{(k)}}{x_{k-1,k-1}^{(k-1)}} && (10) \\ &= \frac{\delta_k x_{i,m}^{(k)}(Z_1, \ldots, Z_{k-1}, Z_i) - x_{i,k}^{(k)} f_k(Z_1, \ldots, Z_k)}{\delta_{k-1}}. && (11) \end{aligned}$$

Each of the linear functions $x_{i,m}^{(k)}(Z_1, \ldots, Z_{k-1}, Z_i)$ and $f_k(Z_1, \ldots, Z_k)$ has k coefficients that are minors of N of order $k-1$, and hence has bit-size at most $m(L + \lg m)$. Hence it takes takes $O(k)$ arithmetic operations to compute $x_{i,m}^{(k+1)}$. Summing over the cost for computing the entries of column m, we again have $O(m^3)$ arithmetic operations on integers of bit size $O(m(L + \lg m))$. So the overall complexity is exactly as in the original Bareiss' algorithm.

Exact Division. Exact division turns out to be slightly more efficient than division-with-remainder (by a small constant factor). We briefly describe the method (see Jebelean [92] for more details). Suppose $C = A \cdot B$ is an integer product. Consider the problem of computing B given

C, A where integers are in base β. Let $0 \le \ell(A) < \beta$ denote the least significant digit (LSD) of A. Then it is easy to see that $\ell(C) \equiv \ell(A)\ell(B)(\bmod \beta)$. Thus

$$\ell(B) = (\ell(C) \cdot (\ell(A)^{-1}) \,\mathbf{mod}\, \beta)$$

provided $\ell(A)$ is invertible mod β. For simplicity, assume β is prime so that this is always possible. This immediately leads to an exact division algorithm that produces the digits of B sequentially, beginning with the LSD of B. Clearly, this is the opposite of the classical division algorithm [105], and avoids the "guess-and-correct" step of the classical method:

```
EXACT DIVISION
      Input: $A, C \in \mathbb{Z}$ in a prime base $\beta$, $A|C$.
      Output: $B$ such that $AB = C$.
NORMALIZATION:
1.        while $\beta|A$ do
2.              $A \leftarrow A/\beta$; $C \leftarrow C/\beta$.
MAIN LOOP:
3.        while $C > 0$ do
4.              $b \leftarrow \ell(C)/\ell(A) \,\mathbf{mod}\, \beta$;
5.              Output $b$;
6.              $C \leftarrow (C - b \cdot A)/\beta$.
```

Let $\texttt{len}(A)$ denote the number of digits in A. Step 6 is considered the inner loop. To speed up this step, observe that only the lowest $\texttt{len}(B)$ digits of C are involved in the inner loop. Hence the main loop can be improved as follows:

```
          ...
MAIN LOOP:
3.        for $k \leftarrow (\texttt{len}(C) - \texttt{len}(A) + 1)$ downto 1
4.              $b \leftarrow \ell(C)/\ell(A) \,\mathbf{mod}\, \beta$;
5.              Output $b$;
6.              $C \leftarrow ((C - b \cdot A) \,\mathbf{mod}\, \beta^k)/\beta$.
```

If β is a power of 2, then $\ell(A)$ would be invertible if A is odd. We achieve this by a simple modification to the normalization stage, namely, by inserting steps 2.1 and 2.2 below:

```
NORMALIZATION:
1.        while $\beta|A$ do
2.              $A \leftarrow A/\beta$; $C \leftarrow C/\beta$.
2.1       while $2|A$ do
2.2             $A \leftarrow A/2$; $C \leftarrow C/2$.
MAIN LOOP:
          ...
```

At the end of normalization, A is odd, and hence $\ell(A)$ is odd. This guarantees that $\ell(A)$ is invertible. The bit analysis of this algorithm is left to an exercise.

Exercises

Exercise 2.1: M is a square matrix with $(M)_{ij} = x_{ij}$ for all i, j. The "cofactor" of $x_{ij}x_{i'j'}$ is defined to be the expression E that is multiplied by $x_{ij}x_{i'j'}$ when we collect terms in the determinant of M that involve both x_{ij} and $x_{i'j'}$. E.g., if M is 2×2, the cofactor of $x_{12}x_{21}$ is -1 and the cofactor of $x_{11}x_{22}$ is 1. If $i = i'$ or $j = j'$ then $E = 0$; otherwise, show that

$$E = (-1)^{i+i'+j+j'+\delta} \det M[i, i'; j, j']$$

where

$$\delta = \begin{cases} 1 & \text{if } (i > i') \oplus (j > j'), \qquad (\oplus \text{ is exclusive-or}) \\ 0 & \text{else,} \end{cases}$$

and $M[i, i'; j, j']$ is the submatrix of M obtained by deleting rows i and i' and deleting columns j and j'. □

Exercise 2.2: Show that $\mathtt{adj}(\mathtt{adj} A) = \det(A)^n A$. □

Exercise 2.3: What is the 3×3 matrix analogue of equation (5)? □

Exercise 2.4: Carry out the dth order version of Bareiss algorithm, by exploiting the order d Sylvester identity. For instance, for $d = 2$, we must construct $x_{ij}^{(k)}$ for even values of k, evaluating 3×3 determinants. □

Exercise 2.5: Modify Bareiss' algorithm in order to compute the determinant of a matrix with rational entries. Carry out comparison experiments in this setting of rational inputs. HINT: make each row have a common denominator first. □

Exercise 2.6: Suppose M is $n \times m$ where $n \le m$. In Bareiss' algorithm, we replace line 4 with "for $j = k+1, \ldots, m$ do" (*i.e.*, we extend change the limit from n to m). Describe the contents of the entries $(M)_{n,j}$ for $j = n, n+1, \ldots, m$. What is the complexity of this modification? How is this related to the determinantal polynomials (§III.3)? □

Exercise 2.7: In the exact division algorithm, show that when the length m of C is less than twice the length n of A, this method uses no more than half the number of bit-operations of the classical method. Quantify this bit-operation count more generally in terms of m, n. □

§3. Matrix Inversion

Matrix inverse is easily computed using the standard Gaussian elimination procedure. Following [69], let us compute the more general product

$$CA^{-1}B$$

where the triple product CAB has shape $m \times n \times n \times p$. We proceed as follows: apply Gaussian elimination to the $(m + n) \times (n + p)$ matrix

$$M = \begin{bmatrix} A & B \\ -C & \mathbf{0} \end{bmatrix}, \qquad (A \text{ is nonsingular}) \tag{12}$$

and obtain (after operations that zero the first n columns below A)

$$M' = \left[\begin{array}{cc} A' & B' \\ \mathbf{0} & D' \end{array}\right]. \tag{13}$$

Here $\mathbf{0}$ denotes a matrix of zeros. Note that if A is singular, we would have discovered this fact during Gaussian elimination. Henceforth, assume A is non-singular. We claim that D' is precisely what we wanted:

Lemma 6

$$D' = CA^{-1}B.$$

Block Gaussian elimination. This lemma is a slight generalization of lemma 1 to the non-square case. It is instructive to briefly consider Gaussian elimination for block-size elements. Let

$$M = \left[\begin{array}{cccc} A_{11} & A_{12} & \cdots & A_{1n} \\ A_{21} & A_{22} & \cdots & A_{2n} \\ \vdots & & \ddots & \vdots \\ A_{m1} & A_{m2} & \cdots & A_{mn} \end{array}\right]$$

where A_{ij} is a $k_i \times \ell_j$ matrix ("block"). We may say that M has shape $(k_1, \ldots, k_m) \times (\ell_1, \ldots, \ell_n)$. Consider the following transformation of M: for $1 \le r, s \le m$, $r \ne s$, and any $k_r \times k_s$ matrix H, we replace the rth row of M with the sum of the rth row and H right multiplied by the sth row. That is,

$$A_{rt} \leftarrow A_{rt} + HA_{st}, \qquad (\text{for } t = 1, \ldots, n).$$

This is equivalent to left multiplying M by the matrix

$$T_{r,s}(H) := \left[\begin{array}{ccccccc} I_1 & \cdots & \mathbf{0} & \cdots & \mathbf{0} & \cdots & \mathbf{0} \\ \vdots & \ddots & \vdots & & \vdots & & \vdots \\ \mathbf{0} & \cdots & I_r & \cdots & H & \cdots & \mathbf{0} \\ \vdots & & \vdots & \ddots & \vdots & & \vdots \\ \mathbf{0} & \cdots & \mathbf{0} & \cdots & I_s & \cdots & \mathbf{0} \\ \vdots & & \vdots & & \vdots & \ddots & \vdots \\ \mathbf{0} & \cdots & \mathbf{0} & \cdots & \mathbf{0} & \cdots & I_m \end{array}\right]$$

where each I_j is the k_j-square identity matrix, the I_j's occurring along the main diagonal, $\mathbf{0}$-blocks occurring everywhere else except at the (r, s)th position which is occupied by H. Clearly, for any minor Δ of M (now viewed as an ordinary matrix of shape $(\sum_{i=1}^m k_i) \times (\sum_{j=1}^n \ell_j)$) that contains rows in the ith and jth block rows of M, this operation preserves Δ. If $m = n$ and for all i, $k_i = \ell_i$ and $A_{i,i}$ is non-singular, then we can effect Gaussian elimination at this block level: for instance, $T_{i1}(A_{i1}A_{11}^{-1})$ will make the $(i, 1)$th entry zero. In particular, if A is square and non-singular, we may transform the matrix

$$N = \left[\begin{array}{cc} A & B \\ C & D \end{array}\right] \tag{14}$$

to

$$N' = \left[\begin{array}{cc} A' & B' \\ \mathbf{0} & D - CA^{-1}B \end{array}\right].$$

We may conclude:

Lemma 7 *The rank of N is equal to the rank of A iff $D = CA^{-1}B$.*

We complete the proof of lemma 6. Suppose, instead of M in equation (12), we had

$$P = \begin{bmatrix} A & B \\ -C & -CA^{-1}B \end{bmatrix}.$$

Then by lemma 7, the rank of P is equal to the rank of M. Applying Gaussian elimination to P to zero the entries below A, we obtain

$$P' = \begin{bmatrix} A' & B' \\ \mathbf{0} & D' - CA^{-1}B \end{bmatrix}.$$

The matrices A', B', D' here are the same as those in M' (equation (13)) since the same multipliers were used to do the elimination. But the rank of P' equals the rank of P and hence of M. This can only mean that $D' - CA^{-1}B = \mathbf{0}$, as we wanted shown.

Exercises

Exercise 3.1: Let N be square in equation (14).
(i) Show that $\det N = \det(AD - ACA^{-1}B)$ if A^{-1} exists.
(ii) Under what conditions is $\det N = \det(AD - CB)$? $\det N = \det(AD - BC)$?
(iii) Consider the four cases of these identities depending on which of the 4 blocks of N are non-singular.
(iv) Obtain the corresponding formulas using column operations (multiplication from the right). □

Exercise 3.2: Use the method to compute $CA^{-1}B$ to solve the system $Ax = b$ of linear equations. □

Exercise 3.3: For $n > 1$, let U be the n-square matrix all of whose entries are 1 and let $S = U - I$. So S has zeros along its main diagonal. Compute S^{-1} using the above algorithm for small n. NOTE: $S^{-1} = \frac{1}{n-1}U - I$. □

Exercise 3.4: Let $T(n)$ be the algebraic complexity of computing the determinant of an n-square matrix. Show that $T(n) = O(\mathrm{MM}(n))$ where $\mathrm{MM}(n)$ is the complexity of multiplying two n-square matrices. □

§4. Hermite Normal Form

Let $A, B \in \mathbb{Z}^{m\times n}$. In §VIII.1, we viewed the columns of A as a generating set of the lattice $\Lambda(A) \subseteq \mathbb{Z}^m$. Alternatively, we may regard $\Lambda(A)$ as a subgroup of the Abelian group $\mathbb{Z}^m$. We have shown (§VIII.1)

$$\Lambda(A) = \Lambda(B) \text{iff} \quad A = B \cdot U \tag{15}$$

for some integer unimodular matrix U. The original proof requires A and B to be bases but this assumption can be dropped (see below). This raises the question: how can we decide if two matrices A, B generate the same subgroup (or lattice)? The result (15) does not appear to be helpful for this purpose – there is no obvious way to find U even if one is known to exist. In this lecture, and unlike §VIII.1, we will no longer assume that A has rank n, so the columns of A form a generating set but not necessarily a basis of $\Lambda(A)$. The tool for answering such questions about $\Lambda(A)$ is a normal form which we now describe.

By applying the elementary *integer* column operations (§VIII.1) to A, we can transform A to a matrix $H = H(A)$ of the following shape:

1. For some $r = 1, \ldots, n$, the first r columns of H are non-zero and the remaining columns (if any) are zero.
2. For $i = 1, \ldots, r$, let $(H)_{c(i),i}$ be the first non-zero entry of column i. Then
$$1 \le c(1) < c(2) < \cdots < c(r) \le n. \tag{16}$$
3. For $i = 1, \ldots, r$, $(H)_{c(i),i} > 0$.
4. If $1 \le j < i \le r$ then $0 \le (H)_{c(i),j} < (H)_{c(i),i}$.

Such a matrix $H = H(A)$ is said to be in *Hermite normal norm* (HNF), or H is *the HNF of* A. For instance, the following 6×4 matrix H_1 is in HNF:

$$H_1 = \begin{bmatrix} 2 & 0 & 0 & 0 \\ -1 & 0 & 0 & 0 \\ 0 & 3 & 0 & 0 \\ 0 & 1 & 2 & 0 \\ 2 & 10 & -4 & 0 \\ 9 & -3 & 0 & 0 \end{bmatrix}. \tag{17}$$

If only the first two conditions in the definition of HNF hold, H is said to be in *column echelon form* Column echelon form is just a generalization of "lower triangular matrices". If H is a column echelon form of A, then rows $c(1), \ldots, c(r)$ are called the *critical rows* of H. Each of the entries $(H)_{c(i),i}$ is called a *critical entry* of H. The number r of non-zero columns in H is called the *rank* of A. For the matrix H_1 in (17), the critical rows are the 1st, 3rd and 4th, the critical entries are $(1,1), (3,2)$ and $(4,3)$.

Since the elementary column operations preserve the underlying lattice, $\Lambda(A) = \Lambda(H(A))$. It is then easy to see: *row i is critical in $H(A)$ iff there is a vector $\xi \in \Lambda(H(A))$ whose first non-zero entry is in the ith position.* This latter characterization of critical rows depends only on the lattice. This shows that the set of critical rows of $H(A)$ depends only on $\Lambda(A)$.

Theorem 8 *Let A, B have the same shape. $\Lambda(A) = \Lambda(B)$ if and only if A and B have the same Hermite normal form: $H(A) = H(B)$.*

Proof. If $H(A) = H(B)$ then clearly $\Lambda(A) = \Lambda(B)$. In the other direction, let us assume $\Lambda(A) = \Lambda(B)$. We wish to prove $H(A) = H(B)$. First observe that A and B have the same rank, d. This is because A, B have the same set of critical rows, by a preceding observation. But d is the number of critical rows.

We use induction on d. We assume that $d \ge 1$ since $d = 0$ is trivial. Let a, b be the first columns of $H(A)$ and $H(B)$, respectively. Let $c(1)$ and $c'(1)$ denote the first non-zero entry of a and b, respectively. It is easy to see that $c(1) = c'(1)$. Indeed, these two entries must be identical since they each generate all the $c(1)$th entries of vectors in $\Lambda(A) = \Lambda(B)$.

Let $H_1(A)$ and $H_1(B)$ be obtained by deleting a and b, respectively. Notice that $\Lambda(H_1(A))$ generates the subgroup of $\Lambda(A)$ whose first $c(1)$ entries are zero. Similarly $\Lambda(H_1(B))$ generates the subgroup of $\Lambda(B)$ whose first $c(1)$ entries are zero. Since $\Lambda(A) = \Lambda(B)$, we conclude that $\Lambda(H_1(A)) = \Lambda(H_1(B))$. Since $H_1(A)$ and $H_1(B)$ are in HNF, we deduce by induction that $H_1(A) = H_1(B)$.

It remains to prove that $a = b$. Suppose $a - b$ is not identically zero. It follows from the preceding that the first $c(1)$ entries of $a-b$ are zero. If k is the first non-zero column of $a-b$, then $k > c(1)$ and it follows that there exists a column c in $H_1(A)$ whose first non-zero entry is in position k. If $(a)_k$ denotes the kth component of a, then by the definition of HNF, $(c)_k > (a)_k \geq 0$ and $(c)_k > (b)_k \geq 0$. Hence $(a-b)_k$ has absolute value less than $(c)_k$. This is impossible since $a - b \in \Lambda(H_1(A))$ means that $(a-b)_k$ must be a multiple of $(c)_k$. **Q.E.D.**

As corollary, we also see that (15) holds without the restriction that the columns of A, B (respectively) are linearly independent. It suffices to show that if $\Lambda(A) = \Lambda(B)$ then there is a unimodular matrix U such that $AU = B$. But there are unimodular matrices U_A, U_B such that $AU_A = H(A) = H(B) = BU_B$. Take $U = U_A U_B^{-1}$.

It follows that our basic question of deciding if $\Lambda(A) = \Lambda(B)$ is reduced to computing and comparing their HNF's. Other questions such as checking whether a given vector ξ belongs to $\Lambda(A)$ can similarly be answered (Exercise). We next address the computation of HNF.

Generic HNF algorithm. Assume the input is the $m \times n$ matrix A. By the "subrow" at an entry $(A)_{i,j}$ we mean the set of entries of the form $(A)_{i,k}$ for $k = j, j+1, \ldots, n$. The best way to understand this algorithm is to imagine that our task (the main loop below) is to determine the critical rows of A.

```
GENERIC HNF ALGORITHM:
    Input: an $m \times n$ integer matrix $A$.
    Output: the Hermite normal norm of $A$.
MAIN LOOP:
1.      Initialize $i \leftarrow 1$ and $j \leftarrow 1$.
2.      while $i \leq m$ and $j \leq n$ do:
3.          While the subrow at $(A)_{ij}$ has only zero entries, increment $i$.
4.          Now assume the subrow at $(A)_{ij}$ has non-zero entries.
5.          By adding multiples of one column to another, eliminate all but
                one non-zero element in the subrow at $(A)_{ij}$.
6.          By a column-exchange, bring this sole non-zero entry to
                position $(i, j)$ and increment $j$.
        end{while}
CLEAN UP:
7.      At this point, the matrix is in column-echelon form.
8.      By adding multiples of one column to another, achieve the remaining
            two conditions for the definition of HNF.
```

Exercise 4.1: (i) There are several ways to fill in details in the generic algorithm. Describe a reasonable choice.
(ii) For your version of the generic algorithm, analyze the number of arithmetic operations for a $m \times n$ input matrix whose entries are L-bit integers.
(iii) Bound the bit complexity of your algorithm. □

Although part (i) of the exercise should give a polynomial bound in m, n, L, the bit complexity in part (ii) should be exponential in $\min\{m, n\}$. Note that these exponential bounds arise from potential sizes of intermediate matrix entries; the final entries in the HNF can be shown to be polynomially bounded (below). Nevertheless, it is an open problem to construct examples that actually exhibit exponential behavior. In random examples, huge intermediate entries routinely appear. For instance, Hafner and McCurley [77] reported that for random 20×20 matrices with entries between 0 and 10, most gave rise to an entry exceeding 10^{500}. One example has an entry exceeding 10^{5011}. We will develop a modular technique to achieve polynomial bit-complexity bounds.

Invariants of the Hermite normal form. Let $A \in \mathbb{Z}^{m\times n}$ and $1 \le k \le \min\{m, n\}$. For $1 \le i_1 < i_2 < \cdots < i_k \le m$, let $A(i_1, i_2, \ldots, i_k)$ denote the submatrix of A formed by rows $i_1, \ldots, i_k$. Let $\gamma(A; i_1, \ldots, i_k)$ denote the GCD of all the $k \times k$ subdeterminants (*i.e.*, order k minors) of $A(i_1, i_2, \ldots, i_k)$. Note that $\gamma(A; i_1, \ldots, i_k) = 0$ iff every order k minor of $A(i_1, i_2, \ldots, i_k)$ is zero.

Lemma 9 *The elementary integer column operations on A preserve $\gamma(A; i_1, i_2, \ldots, i_k)$.*

Proof. The column operations that interchange two columns or multiply a column by -1 do not change the GCD (since GCD is defined up to associates and we always pick the positive member.) Suppose $c \in \mathbb{Z}$ times the ith column of A is added to the jth column. Certain of the $k \times k$ subdeterminants of $A(i_1, i_2, \ldots, i_k)$ change. If a subdeterminant value D is changed, say to D', it is easy to see that $D - D' = \pm c \cdot D''$ where D'' is another subdeterminant. Moreover, D'' is among the subdeterminants of $A(i_1, \ldots, i_k)$ that have not changed. To see this, observe that a subdeterminant D is changed iff D involves column j but not column i. Schematically, if the old GCD is $\mathtt{GCD}(\ldots, D, \ldots, D'', \ldots)$ then the new one is

$$\mathtt{GCD}(\ldots, D', \ldots, D'', \ldots) = \mathtt{GCD}(\ldots, D \pm c \cdot D'', \ldots, D'', \ldots).$$

But the later expression is equal to the old GCD. **Q.E.D.**

Let r be the rank of A. For $k = 1, \ldots, r$, we use the shorthand

$$\gamma_k(A) := \gamma(A; c(1), c(2), \ldots, c(k)).$$

We define $\gamma_i(A) = 0$ for $i = r + 1, \ldots, n$.

Corollary 10 *The value $\gamma_k(A)$ is invariant under the elementary column operations on A. If H is the Hermite normal form of A, then the product of the first k critical values of H is equal to $\gamma_k(A)$. In particular, $\gamma_k(A)$ divides $\gamma_{k+1}(A)$ for $k = 1, \ldots, n - 1$.*

Exercises

Exercise 4.2: Describe the HNF of an $m \times 1$ matrix. □

Exercise 4.3: Discuss other strategies for implementing the generic HNF algorithm. □

Exercise 4.4: Solve the following problem efficiently:
(i) Given $x \in \mathbb{Z}^m$ and $A \in \mathbb{Z}^{m\times n}$, decide if $x \in \Lambda(A)$. If $x \in \Lambda(A)$, find the n-vector ξ such that $A\xi = x$.
(ii) Given $A, B \in \mathbb{Z}^{m\times n}$, check whether $\Lambda(A) = \Lambda(B)$. In case they are equal, construct matrices U, V such that $A = BU$ and $B = AV$. [These matrices need not be unimodular unless A, B are bases for $\Lambda(A)$.] $\square$

Exercise 4.5: Suppose A, B are both in column-echelon form, and $\Lambda(A) \subseteq \Lambda(B)$. If the critical entries in A and the critical entries in B are in the same position and corresponding critical entries have the same values, then $\Lambda(A) = \Lambda(B)$ $\square$

Exercise 4.6: (i) Every subgroup of $\mathbb{Z}^n$ is finitely generated.
(ii) Every finitely generated Abelian group G is isomorphic to a subgroup of $\mathbb{Z}^n$ for some n. $\square$

Exercise 4.7: Call H a *generalized HNF* for A if H is in HNF and obtained from A by the usual elementary column operations, but now we allow the permutation of rows as well. How do the various generalized HNF's of A relate to each other? $\square$

Exercise 4.8: (Open) Given a matrix $A \in \mathbb{Z}^{m\times n}$ and its Hermite normal form $H(A)$. Let L bound the bit sizes of entries in A and $H(A)$. What is best upper bound $B(m, n, L)$ such that there exists a sequence of elementary integer column operations from A to $H(A)$ where all intermediate entries have bit size at most $B(m, n, L)$? $\square$

§5. A Multiple GCD Bound and Algorithm

Computing multiple GCD's is a key subproblem in Hermite normal forms. For example, in the generic HNF algorithm (§4), the process of zeroing out all but one entry of a subrow amounts to computing the multiple GCD of the entries in the subrow. Of course, multiple GCD can be reduced to simple GCD, *i.e.*, GCD for two elements. But this is not the most efficient method. In this section, we present an efficient multiple GCD algorithm over integers. This is based on a co-factor bound that we first derive.

In the following, fix

$$(a_1, a_2, \ldots, a_k) \qquad (k \geq 2)$$

such that the a_i's are distinct and positive. Let $d = \mathtt{GCD}(a_1, \ldots, a_k)$. By definition, an integer sequence $(s_1, \ldots, s_k)$ is called a *co-factor* of $(a_1, \ldots, a_k)$ if

$$d = s_1 a_1 + s_2 a_2 + \cdots + s_k a_k.$$

The "co-GCD problem" refers to the problem of computing a co-factor of $(a_1, \ldots, a_k)$. Note that once a co-factor is available, we can easily compute the GCD. Our first goal is to prove the existence of a co-factor with each $|s_i|$ upper bounded by a_1. We use an argument of Hongwei Cheng:

Lemma 11 *If $d = \mathtt{GCD}(a_1, \ldots, a_k)$ then there exists a co-factor $(s_1, \ldots, s_k)$ for $(a_1, \ldots, a_k)$ such that*

$$|s_1| \leq \frac{a_k}{2},$$

$$\begin{aligned} |s_i| &< \frac{a_{i-1}+a_k}{2} \qquad i=2,\ldots,k-1,\\ |s_k| &\le \frac{d}{a_k}+\frac{a_{k-1}}{2}. \end{aligned}$$

Proof. Suppose $(t_1,\ldots,t_k)$ is any co-factor for $(a_1,\ldots,a_k)$. Define

$$s_i := \begin{cases} t_i - q_i a_k & i=1,\ldots,k-1\\ t_k + \sum_{j=1}^{k-1} q_j a_j & i=k, \end{cases}$$

where $q_1,\ldots,q_k \in \mathbb{Z}$ are to be specified. It is not hard to check that $(s_1,\ldots,s_k)$ is also a co-factor for $(a_1,\ldots,a_k)$. We now define the q_i's inductively. Pick q_1 to be the symmetric quotient (§II.3) of t_1 divided by a_k. Then $|s_1| \le a_k/2$, as desired. Now consider the partial sums

$$S_i = \sum_{j=1}^{i} s_j a_j.$$

Thus $S_1 = a_1 s_1$ and $|S_1| \le a_1 a_k/2$. Inductively, assume S_{i-1} has been defined so that $|S_{i-1}| \le a_{i-1}a_k/2$. For $i=2,\ldots,k-1$, define q_i so that $|S_i| \le a_i a_k/2$. This is clearly possible since $S_i = S_{i-1} + a_i s_i = S_{i-1} + a_i(t_i - q_i a_k)$. It follows that

$$\begin{aligned} |a_i s_i| &= |S_i - S_{i-1}| \le \frac{a_i a_k}{2} + \frac{a_{i-1}a_k}{2}\\ |s_i| &< \frac{a_k}{2} + \frac{a_{i-1}}{2}, \end{aligned}$$

as desired. Finally, we bound $|s_k|$. By definition of S_k, we have $S_k = d$, the GCD of $a_1,\ldots,a_k$. So $|s_k a_k| = |S_k - S_{k-1}| \le d + \frac{a_{k-1}a_k}{2}$ or $|s_k| \le \frac{d}{a_k} + \frac{a_{k-1}}{2}$. **Q.E.D.**

Note that for $k=2$, this lemma gives a well-known bound

$$|s_1| \le \frac{a_2}{2}, \qquad |s_2| \le 1 + \frac{a_1}{2}.$$

Suppose we further assume that

$$a_1 > a_2 > \cdots > a_k.$$

Then we may further infer the bounds $|s_i| \le a_{i-1} - (k-i+1)/2$ for $i=2,\ldots,k$ and $|s_1| \le a_k/2$. This yields:

Corollary 12 *For all $a_1 > a_2 > \cdots > a_k \ge 2$, there exists a co-factor $(s_1,\ldots,s_k)$ for $(a_1,\ldots,a_k)$ such that*

$$\prod_{i=1}^{k} |s_i| < \frac{a_k}{2} \prod_{i=1}^{k-1}\left(a_i - \frac{k-i+1}{2}\right) \le \prod_{i=1}^{k}(a_i - 1).$$

This says that the output size of the co-GCD algorithm need not be larger than the input size.

We now address the question of computing a co-factor $(s_1,\ldots,s_k)$ satisfying the lemma. We will use a divide and conquer approach. We split $a_1,\ldots,a_k$ into two subsets according to the parity of their subscripts, and assume inductively that we have computed $c, c', t_1,\ldots,t_k$ such that

$$\begin{aligned} c &:= \mathtt{GCD}(a_2, a_4, \ldots, a_{2\lfloor k/2\rfloor}) = \sum_{i=1}^{\lfloor k/2\rfloor} t_{2i}a_{2i},\\ c' &:= \mathtt{GCD}(a_1, a_3, \ldots, a_{2\lfloor (k-1)/2\rfloor+1}) = \sum_{i=1}^{\lfloor (k-1)/2\rfloor} t_{2i+1}a_{2i+1}. \end{aligned}$$

By a call to the simple extended GCD on c, c', we obtain d, t, t' such that

$$d = tc + t'c'.$$

By induction, we may assume that $|t_i|$ is bounded according to lemma 11. In particular, $|t_i| < a_1$ for $i = 1, \ldots, k$. Similarly, $|t| < a_1$ and $|t'| < a_1$. Define

$$s'_1 = \begin{cases} t_i t & \text{if } i = \text{even}, \\ t_i t' & \text{if } i = \text{odd}. \end{cases}$$

Thus $\sum_{i=1}^{k} s'_i a_i = d$ and $|s'_i| < a_1^2$ for all i. Following the proof of lemma 11, we may now reduce $s'_1, \ldots, s'_k$ to $s_1, \ldots, s_k$:

```
Reduction Step:
1.  $s_1 \leftarrow s'_1 \,\mathbf{mod}\, a_k$ and $S_1 \leftarrow s_1 a_1$.
2.  for $i = 2, \ldots, k-1$ do
        $S_i \leftarrow (S_{i-1} + a_i s'_i) \,\mathbf{mod}\, a_i a_k$;
        $s_i \leftarrow (S_i - S_{i-1})/a_i$;
        $q_i \leftarrow (s'_i - s_i)/a_k$.
3.  $s_k \leftarrow s'_k + \sum_{i=1}^{k-1} q_i a_i$.
```

The **mod** operator here is the symmetric version (§II.3). Note that both divisions in step 2 are exact.

Let

$$L = \lg \max\{a_1, \ldots, a_k\}. \tag{18}$$

The bit complexity of this reduction step is $O(k\mathrm{M}_B(L))$. Let $T(k, L)$ be the bit complexity of the overall recursive procedure. Clearly

$$T(k, L) = 2T(k/2, L) + O(k\mathrm{M}_B(L) + \mathrm{M}_B(L)\log L) \tag{19}$$

where '$k\mathrm{M}_B(L)$' comes from the reduction step and '$\mathrm{M}_B(L)\log L$' comes from the simple co-GCD computation for c, c' (Lecture II). The solution is easily seen to be $T(k, L) = O(k(\log k + \log L)\mathrm{M}_B(L))$. Thus we have:

Theorem 13 *There is a multiple co-GCD algorithm with bit complexity*

$$T(k, L) = O(k(\log k + \log L)\mathrm{M}_B(L)).$$

On input $(a_1, \ldots, a_k)$, the output co-factor $(s_1, \ldots, s_k)$ satisfy the bounds of lemma 11.

Application to multiple LCM. There are many applications of multiple GCD. An obvious application is for computing the primitive factorization (§III.1) of an integer polynomial. Bareiss [10] states that "there is no question that for maximum efficiency in any integer-preserving elimination code, the elements of all the rows and columns respectively should be made relative prime to each other before starting the elimination process". Here we consider another application: the computation of multiple LCM. Simple GCD and simple LCM are closely related: $\mathtt{LCM}(a, b) = ab/\mathtt{GCD}(a, b)$. It is slightly more involved to relate multiple GCD with multiple LCM. Our multiple GCD algorithm computes intermediate information that can be used to obtain the multiple LCM. Specifically, on input $(a_1, \ldots, a_k)$ the information can be organized as a binary tree T of height $\lceil \lg k \rceil$ with k leaves, each associated with an a_i. At each internal node u of T, let $S_u \subseteq \{a_1, \ldots, a_k\}$ denote the a_i's associated to the leaves of the subtree rooted at u. Assume we store at u the value $\mathtt{GCD}(S_u)$. It is then simple to extend the multiple GCD algorithm so that we recursively compute at node u the LCM of S_u. If v, w are the children of u, we use the formula

$$\mathtt{LCM}(S_u) = \mathtt{LCM}(\mathtt{LCM}(S_v), \mathtt{LCM}(S_w)) = \frac{\mathtt{LCM}(S_v) \cdot \mathtt{LCM}(S_w)}{\mathtt{GCD}(S_u)}.$$

Let $L' = \lg \mathtt{LCM}(a_1, \ldots, a_k)$. With L as in (18), we see that $L' \le kL$. The cost of additional computation at u is $O(\mathrm{M}_B(L'))$. Overall, the additional cost is $O(k\mathrm{M}_B(L'))$. Combined with theorem 13, we conclude:

Theorem 14 *There is an algorithm to compute both the GCD and LCM of $a_1, \ldots, a_k$ with bit complexity $O(k(\log k \cdot \mathrm{M}_B(L) + \log L \cdot \mathrm{M}_B(L) + \mathrm{M}_B(L')))$ which is*

$$O(k(\log L \cdot \mathrm{M}_B(L) + k\mathrm{M}_B(L))).$$

Remarks. Iliopoulos [89] obtains the co-factor bound $|s_i| = O(a_i^k)$ by using a balanced binary tree and co-factor bounds for simple GCD's. Lüneburg [121] gives the bound

$$|s_i| \le a_1/2$$

for all i except when $i = i_0$ (for some i_0). Moreover, $|s_{i_0}| \le (1 + a_1(k-1))/2$. The dependence on k in these bounds is somewhat unsatisfactory. Hongwei Cheng[1] shows the uniform bound $|s_i| < a_1$ for all i; our lemma 11 follows his argument.

Exercises

Exercise 5.1: Suppose we apply lemma 11 with

$$a_{k-1} > a_{k-2} > \cdots > a_2 > a_1 > a_k.$$

How does this compare to the bound in the corollary?

□

Exercise 5.2: Verify the solution to recurrence (19) by an appropriate induction. □

Exercise 5.3: Suppose $a_1, \ldots, a_k \in \mathbb{Z}[X]$. Give an efficient method for computing the extended GCD of $a_1, \ldots, a_k$. □

Exercise 5.4: (Open: Odlyzko, Sims) Is there a function $f(k) > 0$ that goes to infinity as $k \to \infty$ such that for all $a_1 > a_2 > \cdots > a_k \ge 2$, a cofactor $(s_1, \ldots, s_k)$ exists where $|s_i| \le |a_1|/f(k)$? [Can we take $f(k) = \lg k$?] □

§6. Hermite Reduction Step

[1]Private communication, 1991.

Let $a \in \mathbb{Z}^{1\times n}$ be a non-zero row n-vector, $n \geq 2$. The goal in this section is to construct a unimodular matrix $U \in \mathbb{Z}^{n\times n}$ such that $a \cdot U$ is zero except for its first entry. Since U is invertible, the non-zero entry of $a \cdot U$ must be equal to $\texttt{GCD}(a)$. We will call the transformation

$$a \mapsto U \tag{20}$$

a "Hermite reduction step problem". The reason for this name is clear. For, if a is the first row in a $m \times n$ matrix, then $A \cdot U$ is essentially the first step of the generic HNF algorithm. Repeating this process suitably, we finally obtain a column echelon form of A, which is easily converted into the Hermite normal form.

First let us illustrate our approach for the case $n = 2$. Suppose $a = (a_1, a_2)$ and we wish to find a 2×2 unimodular U such that $aU = (g, 0)$ where $g = \texttt{GCD}(a_1, a_2)$. So there exist integers s, t such that $g = sa_1 + ta_2$. Moreover, s, t are relatively prime. Hence there exist integers s', t' such that $ss' + tt' = 1$. In fact, we may choose $s' = a_1/g$ and $t' = a_2/g$. Then it is easy to see that

$$(a_1, a_2)\begin{bmatrix} s & -t' \\ t & s' \end{bmatrix} = (g, g') \tag{21}$$

for some g'. But g divides g' (since g divides a_1 and a_2). If $e = g'/g$ then we see that the desired U can be taken to be $P \cdot P^*$ where

$$P = \begin{bmatrix} s & -t' \\ t & s' \end{bmatrix}, \qquad P^* = \begin{bmatrix} 1 & -e \\ 0 & 1 \end{bmatrix}.$$

It turns out that U can always be written as the product of matrices P and P^*. In fact, we will see that it is more useful to represent U as the pair (P, P^*). We begin with a key lemma which shows how to construct P (see [86, p. 375] or [145]).

Lemma 15 *Let $u = (u_1, \ldots, u_n)^T$ be a column of integers with $d = \texttt{GCD}(u)$. There exists an $n \times n$ integer matrix P with the following properties:*

(i) *The first column of P equals u.*

(ii) $\det P = d$.

(iii) $\|P\|_\infty < \|u\|_\infty$.

Proof. We use induction on n. The result is trivial for $n = 1$ so let $n \geq 2$. Let $u' = (u_1, \ldots, u_{n-1})^T$ with $d' = \texttt{GCD}(u')$. By induction, there is a matrix P' with first column u' and $\det P' = d'$. So there are integers s, t such that

$$sd' + tu_n = d. \tag{22}$$

We may assume that $|s| < |u_n|$ and $|t| < |d'|$. We claim (§5) that the desired matrix can be taken to be

$$P = \left[\begin{array}{cccc|c} & & & & \frac{u_1}{r} \\ & P' & & & \frac{u_2}{r} \\ & & & & \vdots \\ & & & & \frac{u_{n-1}}{r} \\ \hline u_n & 0 & \cdots & 0 & s \end{array}\right],$$

where $r \in \mathbb{Q}$ is to be determined. Part (i) is clearly satisfied. By expanding the determinant along the last row, we have

$$\det P = s \det P' + u_n \det P'',$$

where P'' is obtained by omitting the first column and last row of P. We want to choose r so that $\det P'' = t$ (so that (ii) is satisfied). We observe that P'' is rather similar to P': the last column of P'' is just $1/r$ times the first column of P'. In fact,

$$P'' = P' \cdot C = P' \cdot \begin{bmatrix} 0 & 0 & 0 & \cdots & 0 & \frac{1}{r} \\ 1 & 0 & 0 & & 0 & 0 \\ 0 & 1 & 0 & & 0 & 0 \\ \vdots & & & \ddots & & \vdots \\ 0 & 0 & 0 & & 0 & 0 \\ 0 & 0 & 0 & \cdots & 1 & 0 \end{bmatrix}.$$

Here C has a subdiagonal of 1's and a top-right entry of $1/r$. Note that C simply circulates the columns of P', moving the first column (multiplied by $1/r$) to the last place and for the other columns, one place to the left. When $n = 2$, C is simply the matrix $[\frac{1}{r}]$. Then $\det P'' = \det C \det P' = (-1)^n \frac{1}{r} d'$. Thus part (ii) is satisfied with $r = (-1)^n d'/t$. To see that P is correct, note that the entries of P are integers since $u_i/r = u_i t/d' \in \mathbb{Z}$ for $i = 1, \ldots, n-1$.

Finally, part (iii) claims that each entry of P is bounded by $\|u\|_\infty$. This is true of the entries of P' (by induction) and also true of the non-zero entries in the rest of P, namely, s, u_n and u_i/r (in the last case, we use the fact that $|t| < |d'|$). **Q.E.D.**

Note that P is not unique since s, t are not unique. In our application, the vector $u = (u_1, \ldots, u_n)^T$ satisfies $\mathtt{GCD}(u) = 1$ so that the matrix P is unimodular.

Now we implement the above lemma.

Lemma 16 *Let $a = (a_1, \ldots, a_n) \neq 0$ be an integer row vector and $\log \|a\|_\infty \le L$.*

(i) *We can construct the unimodular matrix P of the previous lemma 15 in bit complexity*

$$O(n(n + \log L)\mathrm{M}_B(L)). \tag{23}$$

(ii) *Suppose $b = (b_1, \ldots, b_n) \in \mathbb{Z}^n$ satisfies $\log \|b\|_\infty \le L$ then we can compute $b \cdot P$ from b, P in bit complexity*

$$O(n\mathrm{M}_B(L + \log n)). \tag{24}$$

Proof. (i) Let $u = (u_1, \ldots, u_n)^T$ such that $\sum_{i=1}^n u_i a_i = \mathtt{GCD}(a_1, \ldots, a_n) = d$. By theorem 13, we can compute u in time

$$O(n(\log n + \log L)\mathrm{M}_B(L)). \tag{25}$$

Note that $\mathtt{GCD}(u_1, \ldots, u_n) = 1$. As in the proof of lemma 15, we recursively compute the $(n-1) \times (n-1)$ matrix P' with $\det P' = d'$ with first column $(u_1, \ldots, u_{n-1})^T$. Then we compute s, t with $sd' + tu_n = d$, in time $\mathrm{M}_B(L)\log L$. Finally, we compute the last row of P in time $O(n\mathrm{M}_B(L))$. Hence if $T(n, L)$ is the complexity of computing P

$$T(n, L) = T(n-1, L) + O((n + \log L)\mathrm{M}_B(L)).$$

Hence $T(n, L) = O(n(n + \log L)\mathrm{M}_B(L))$, and this dominates the complexity in (25).

(ii) Given a and P, a straightforward multiplication gives bP in time $O(n^2\mathrm{M}_B(L))$. But a better bound is needed. For $i = 2, \ldots, n$, the ith column of P has the form

$$p_i = (\frac{u_1}{r_i}, \frac{u_2}{r_i}, \ldots, \frac{u_{i-1}}{r_i}, s_i, 0, \ldots, 0)^T$$

where r_i, s_i are the elements described in the proof of lemma 15. So

$$bP = (\langle b, p_1 \rangle, \langle b, p_2 \rangle, \ldots, \langle b, p_n \rangle)$$

where $\langle b, p_i \rangle$ indicates scalar product. We will compute the entries of bP in a particular order. Notice that

$$\log |\langle b, p_i \rangle| = O(L + \log n). \tag{26}$$

Clearly $\langle b, p_2 \rangle$ can be computed in time $O(\mathrm{M}_B(L))$. For $i = 2, \ldots, n-1$, $\langle b, p_{i+1} \rangle$ can be obtained from $\langle b, p_i \rangle$ in time $O(\mathrm{M}_B(L + \log n))$ using the formula

$$\langle b, p_{i+1} \rangle = \frac{(\langle b, p_i \rangle - b_i s_i) r_i + b_i u_i}{r_{i+1}} + b_{i+1} s_{i+1}.$$

Finally, the first entry $\langle b, p_1 \rangle$ in bP can be computed from the last entry in time $O(\mathrm{M}_B(L + \log n))$. The entire computation costs $O(n\mathrm{M}_B(L + \log n))$ as claimed. **Q.E.D.**

Continuing in our pursuit of the matrix U, we now need to compute the matrix P^* such that $(aP)P^*$ will have zero everywhere except the first entry. Clearly, if $aP = (g_1, g_2, g_3, \ldots, g_n)$ then $g_1 = d$ and

$$P^* = \begin{bmatrix} 1 & -g_2/d & -g_3/d & \cdots & -g_{n-1}/d & -g_n/d \\ 0 & 1 & 0 & \cdots & 0 & 0 \\ \vdots & & & \ddots & & \vdots \\ 0 & 0 & 0 & \cdots & 1 & 0 \\ 0 & 0 & 0 & \cdots & 0 & 1 \end{bmatrix}.$$

Note that P^* is integer unimodular since d divides each g_i. The entries in the first row of P^* have bit-size of $O(L + \log n)$. We return to the main problem (20):

Lemma 17 (Hermite Reduction Step) *Let $a \neq 0$ be the first row of $A \in \mathbb{Z}^{m \times n}$ and $L = \lg \|A\|_\infty$. Define the matrices P, P^*, U relative to a, as before.*

(i) *We can compute the matrices P, P^*, U from a in time $O(n(n + \log L)\mathrm{M}_B(L + \log n))$.*

(ii) *If b is any row of A, we can compute bU from b, P, P^* in time $O(n\mathrm{M}_B(L + \log n))$.*

(iii) *We can compute AU from A, P, P^* in time $O(mn\mathrm{M}_B(L + \log n))$.*

(iv) *We can compute P, P^*, AU from A in time $O(n(n + m + \log L)\mathrm{M}_B(L + \log m))$.*

Proof. (i) We use the previous lemma to compute P in time (23). Similarly, we can compute $aP = (g_1, \ldots, g_n)$ in time (24). Assuming that we only represent the first row of P^* explicitly, we can also construct P^* in time (24). Next, each entry of $U = PP^*$ can be computed in time $O(\mathrm{M}_B(L + \log n))$ or

$$O(n^2\mathrm{M}_B(L + \log n))$$

for the entire matrix U. Thus the three matrices P, P^*, U can be computed in time

$$O(n(n + \log L)\mathrm{M}_B(L)) + n\mathrm{M}_B(L + \log n) + O(n^2\mathrm{M}_B(L + \log n)) = O(n(n + \log L)\mathrm{M}_B(L + \log n)).$$

(ii) To compute bU, we first compute bP in time (24). Then compute $(bP)P^*$ within the same time bound (24).
(iii) This amounts to repeating part (ii) m times.
(iv) This just adds up parts (i) and (iii). **Q.E.D.**

Application. Although we could use this lemma repeatedly to compute the Hermite normal form, it seems that the best bounds for bit sizes of intermediate values are exponential in $\min\{m, n\}$. So this application is only useful for $m \gg n$ or $n \gg m$. We describe another application here. Suppose the columns of $A = [a_1, \ldots, a_n] \in \mathbb{Z}^{m\times n}$ are not linearly independent. Consider the problem of computing $B = [b_1, \ldots, b_{n-1}] \in \mathbb{Z}^{m\times(n-1)}$ such that $\Lambda(A) = \Lambda(B)$. For instance, this is useful in preprocessing a basis before calling the LLL algorithm, since the LLL algorithm requires the input columns to be linearly independent.

Note that there exists $x \in \mathbb{Z}^{n\times 1}$ such that $Ax = 0$. Finding such an x from A is easily reduced to Gram-Schmidt orthogonalization. In the notations of §IX.1, if $A^* = [a_1^*, \ldots, a_n^*]$ is the Gram-Schmidt version of A, then $a_i^* = 0$ for some i. This means

$$a_i^* = 0 = a_i - \sum_{j=1}^{i-1} \mu_{ij} a_j^*.$$

Recall that μ_{ij} are rational numbers. It is then easy to find integers $t_1, \ldots, t_i$ such that

$$0 = t_i a_i + \sum_{j=1}^{i-1} t_j a_j. \tag{27}$$

We may set $x = (t_1, \ldots, t_i, 0, \ldots, 0)^T$. Clearly, we may assume that $\mathtt{GCD}(x) = 1$. To find B, we can use the Hermite reduction step to give us a unimodular matrix $U \in \mathbb{Z}^{n\times n}$ such that $Ux = (1, 0, \ldots, 0)^T$. Since $(AU^{-1})(Ux) = \mathbf{0}$, we conclude that the first column of AU^{-1} is zero. The desired matrix B may be taken to comprise all but the first column of AU^{-1}.

Algebraic complexity of the Hermite Reduction Step.

We have given an upper bound on the bit complexity of the Hermite Reduction Step (20). But suppose we want its complexity in an algebraic model of computation (§0.6). It is clear from the preceding that problem (20) over $\mathbb{Z}$ can be solved in polynomial time in an algebraic model M if the elementary operations or *basis* of M includes the following:

$$+, \quad -, \quad , \times, \quad \mathtt{cGCD}(x, y). \tag{28}$$

Here cGCD denotes the co-GCD primitive (§II.2) that returns a co-factor (s, t) of an input pair $(x, y) \in \mathbb{Z}^2$: $sx + ty = \mathtt{GCD}(x, y)$. Hafner and McCurley [77, p. 1075] suggested that there may be no solution in case our basis comprises only the ring operations $(+, -, \times)$, *i.e.*, if we omit cGCD. Let us show this. If (20) can be solved in the algebraic model M then $\mathtt{GCD}(x, y)$ can be solved in M in constant time by reduction to the $n = 2$ case. This assumes (as we may) that the ring operations are available in M. Suppose $\pi(x, y)$ is a branching program in M for computing the $\mathtt{GCD}(x, y)$. The inputs x, y and constants used in π are all integers, and at each decision node, there is an integer polynomial $f(X, Y)$ whose sign at the input (x, y) determines the flow of computation. The finiteness of π means that there are finitely many leaf nodes in the branching program. At each leaf ℓ, there is an integer polynomial $P_\ell(X, Y)$ such that if input (x, y) terminates at ℓ the $P_\ell(x, y) = \mathtt{GCD}(x, y)$. Let S_ℓ denote the set of all $(x, y) \in \mathbb{R}^2$ that terminate at ℓ. Note that it makes sense to feed pairs (x, y) of real numbers to π and hence the set S_ℓ is well-defined. Clearly, S_ℓ is a semi-algebraic set (*i.e.*, defined by a finite set of polynomial inequalities). By basic properties of semi-algebraic sets, there is some leaf ℓ such that S_ℓ has the following properties: S_ℓ contains an infinite cone C which in turn contains infinitely many vertical rays of the form $R_i = \{(x_i, y) : y \geq c_i\}$ where $x_i \in \mathbb{Z}$ is prime and c_i is arbitrary, for $i = 0, 1, 2, \ldots$. Focus on the output polynomial $P_\ell(X, Y)$ at such an ℓ. We may pick a ray R_i such that such that none of the non-zero coefficients of the Y's in $P_\ell(X, Y)$ vanish when X is replaced by x_i. Since there are infinitely many prime y's such that $(x_i, y) \in R_i \subseteq S_\ell$, we conclude that $P_\ell(x_i, Y)$ is the constant 1 and $P_\ell(X, Y)$ does not depend on Y. Next suppose

P_ℓ has X-degree $< d$. Pick a disc $D \subseteq S_\ell$ large enough so that there are prime numbers y_0 and u_i $(i = 1, \ldots, d)$ such that D contains (u_i, y_0) for $i = 1, \ldots, d$ and none of the non-zero coefficients of X in $P_\ell(X, Y)$ vanishes when y_0 replaces Y. Again, we argue that $P_\ell(X, y_0)$ is the constant 1 and $P_\ell(X, Y)$ does not depend on X. Thus $P_\ell(X, Y)$ must be the constant 1. But, clearly S_ℓ contains a pair (a, b) of integers such that $\texttt{GCD}(a, b) > 1$. This is a contradiction since $1 = P_\ell(a, b) = \texttt{GCD}(a, b) > 1$.

Exercises

Exercise 6.1: (i) Show that the matrix P in lemma 15 is a product of elementary integer unimodular matrices (§VIII.1). HINT: use induction on n.
(ii) Do the same for P^*. Hence conclude that U in (20) is a product of elementary integer unimodular matrices. □

Exercise 6.2: (i) What is the algebraic complexity of the Hermite reduction step assuming the basis (28)?
(ii) Show that if 2×2 Smith normal form (see §8) can be solved in constant time in an algebraic model M then the general Smith normal form problem can be solved in polynomial time in M.
(iii) Assume an algebraic computation model M whose basis is given by (28). Show that the 2×2 Smith normal form problem is equivalent to the following: given $a, b, c \in \mathbb{Z}$ find $U, V, W, Z \in \mathbb{Z}$ such that $aUV + bVW + cWZ = \texttt{GCD}(a, b, c)$. HINT: first reduce the 2×2 matrix to Hermite normal form.
(iv) (Open) Prove that the problem in (iii) cannot be solved in constant time in model M. □

Exercise 6.3: In the above application, work out efficient algorithms for:
(i) Computing the integers $t_1, \ldots, t_n$ in (27) from μ_{ij}'s.
(ii) Computing the inverse U^{-1} from U. □

Exercise 6.4: Suppose $a = (a_1, \ldots, a_n)^T$ and $b = (b_1, \ldots, b_n)^T$ $(n \geq 2)$ be two columns. Under what circumstances is there a unimodular matrix U whose first and second columns are a and b? A necessary condition is that $\texttt{GCD}(a) = \texttt{GCD}(b) = 1$ and the GCD of all minors of order 2 of $[a, b]$ is 1. □

Exercise 6.5: (Bass) Let $M = [a_{ij}]$ be an $n \times n$ matrix over a ring R. Say M is invertible iff its determinant is invertible in R. But $\det M = a_{11}A_{11} + ... + a_{1n}A_{1n}$ where A_{ij} is the co-factor of a_{ij}. Write a_i for a_{1i}. Then invertibility of M implies $R = \texttt{Ideal}(a_1, \ldots, a_n)$. We also call $(a_1, \ldots, a_n)$ a unimodular row in this case. The converse asks: is every unimodular row the row of an invertible matrix? This is related to a conjecture of Serre's which was answered affirmatively by Quillen and Suslin. Verify the counterexample: $R = \mathbb{R}[X, Y, Z]/(X^2 + Y^2 + Z^2 = 1)$. The unimodular row $(\overline{X}, \overline{Y}, \overline{Z})$ is not the first row of any invertible matrix. Here $\overline{u}$ denotes the image of u in the canonical map from $\mathbb{R}[X, Y, Z]$ to R. □

§7. Bachem-Kannan Algorithm

We present a polynomial time algorithm for Hermite normal form. It is essentially that of Bachem and Kannan [7], extended to non-square matrices of arbitrary rank. *We assume that the input matrix*

$A \in \mathbb{Z}^{m\times n}$ has been preprocessed so that the ith principal minor is non-zero for $i = 1, \ldots, \rho$ where $\rho > 0$ is the rank of A. The Extended Bareiss Algorithm (§2) can convert any matrix A into this form. Moreover, the complexity of this conversion is dominated by the subsequent computation.

Algorithm. The algorithm is relatively straightforward to describe. For any matrix M, let

$$\langle M\rangle_i := M(1,2,\ldots,i;1,2,\ldots,i)$$

using the general matrix notations in §0.9. So $\langle M\rangle_i$ is just the ith principal submatrix of M. On input A, the algorithm has n stages where in the sth ($s = 1, \ldots, n$) stage, we seek to put $\langle A\rangle_s$ into the Hermite normal form. Stage 1 requires no action. Suppose that we have just completed the $(s-1)$st stage. The sth stage has two *phases*. *Elimination phase:* we eliminate (*i.e.*, zero out) the first $s-1$ entries in the sth column. This will make $\langle A\rangle_s$ a lower triangular matrix. *Reduction phase:* we then reduce the off-diagonal entries of $\langle A\rangle_s$ so that each such entry is non-negative and less than the corresponding diagonal entry in its row. This completes the sth stage.

```
Bachem-Kannan Algorithm
    Input:   $A \in \mathbb{Z}^{m\times n}$ and $\rho \ge 1$ the rank of $A$.
                 Assume the $i$th principal minor is non-zero for $i = 1, \ldots, \rho$.
    Output:  $H \in \mathbb{Z}^{m\times n}$, the HNF of $A$, and $U \in \mathbb{Z}^{n\times n}$, a unimodular matrix
                 such that $H = AU$.
INITIALIZATION:
    1.  $H := A$; $U := I$, the identity matrix.
MAIN LOOP:
    2.  for $s \leftarrow 2$ to $n$ do
                ELIMINATION PHASE:
    3.          for $i \leftarrow 1$ to $\min\{s-1, \rho\}$ do
    4.              By postmultiplying $H$ with a suitable unimodular matrix $K$,
                        eliminate the $(i,s)$th entry of $H$; Update $U \leftarrow UK$.
                REDUCTION PHASE:
    5.          if $s > \rho$, skip this phase; else continue.
    6.          for $j \leftarrow s-1$ downto 1 do
    7.              for $i \leftarrow j+1$ to $s$ do
    8.                  By postmultiplying $H$ with a suitable unimodular matrix $K$,
                            reduce the $(i,j)$th entry of $H$; Update $U \leftarrow UK$.
```

Call step 4 an "elimination step" and step 8 a "reduction step". Note that when $s > \rho$, the reduction phase is omitted since column s would be zero after the elimination phase. The order of reduction represented by the double for-loop (steps 6 and 7) is important for the analysis: this is an improvement from Chou and Collins [42]. The reduction step is rather obvious: the entry x to be reduced is replaced by $x \,\mathbf{mod}\, d$ where d is the diagonal element in the same row. The rest of the column of x is modified accordingly. We now illustrate the elimination step: it is basically the 2×2 version of the Hermite reduction step (§6). For instance, suppose $H \in \mathbb{Z}^{5\times 6}$ already has its 3rd principal submatrix in HNF. The following shows the first elimination step of stage 4:

$$H = [h_1, h_2, h_3, h_4, h_5, h_6] \quad\to\quad [h_1', h_2, h_3, h_4', h_5, h_6] = H',$$

$$\begin{bmatrix} a_{1,1} & 0 & 0 & a_{1,4} & * & * \\ a_{2,1} & a_{2,2} & 0 & a_{2,4} & * & * \\ a_{3,1} & a_{3,2} & a_{3,3} & a_{3,4} & * & * \\ a_{4,1} & a_{4,2} & a_{4,3} & a_{4,4} & * & * \\ * & * & * & * & * & * \end{bmatrix} \to \begin{bmatrix} a_{1,1}' & 0 & 0 & 0 & * & * \\ a_{2,1}' & a_{2,2} & 0 & a_{2,4}' & * & * \\ a_{3,1}' & a_{3,2} & a_{3,3} & a_{2,4}' & * & * \\ a_{4,1}' & a_{4,2} & a_{4,3} & a_{2,4}' & * & * \\ * & * & * & * & * & * \end{bmatrix}.$$

As in (21) (§6), this amounts to replacing columns h_1, h_4 (respectively) by h_1', h_4' which is defined as follows:

$$[h_1', h_4'] \leftarrow [h_1, h_4] \begin{bmatrix} s & a_{1,4}/g \\ t & -a_{1,1}/g \end{bmatrix}$$

where $g = \texttt{GCD}(a_{1,1}, a_{1,4}) = s \cdot a_{1,1} + t \cdot a_{1,4}$. We may assume that (see §5)

$$|s| < |a_{1,4}|, \qquad |t| \le |a_{1,1}|. \tag{29}$$

We may write this elimination step as $H' = HK$ where

$$K = \begin{bmatrix} s & 0 & 0 & a_{1,4}/g & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ t & 0 & 0 & -a_{1,1}/g & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}. \tag{30}$$

If U is the unimodular matrix such that $H = AU$ then we update U via $U \leftarrow UK$.

Analysis. There are subtleties in proving a polynomial running time. In this analysis, we simply write $\|M\|$ instead of $\|M\|_\infty$ for the ∞-norm of a matrix M. With

$$\lambda_0 := \|A\|, \quad L := \lg \lambda_0,$$

the analysis amounts to showing that throughout the algorithm, $\lg \|H\|$ and $\lg \|U\|$ are polynomially-bounded in terms of m, n and L.

Bound between stages. Let $H^{(s)}$ denote the H-matrix just after the sth stage. Thus $H^{(1)}$ is equal to the initial matrix A. Also let the unimodular matrix that transforms A to $H^{(s)}$ be denoted $U^{(s)}$:

$$H^{(s)} = AU^{(s)}.$$

So $U^{(1)}$ is the identity matrix. Letting

$$\lambda_1 := (\rho\lambda_0)^\rho, \tag{31}$$

we see that every minor of A is bounded by λ_1. From this we infer

$$\|\langle H^{(s)}\rangle_s\| \le \lambda_1. \tag{32}$$

This is because each step (elimination or reduction) in the first s stages does not change the sth principal minor of H, and H is initially equal to A. Since $\langle H^{(s)}\rangle_s$ is lower triangular, this means that each diagonal entry of $\langle H^{(s)}\rangle_s$ is bounded by λ_1. But the off-diagonal entries are also bounded by λ_1, since $\langle H^{(s)}\rangle_s$ is in HNF. Thus (32) is verified.

First assume $s \le \rho$. Note that $U^{(s)}$ has the form

$$U^{(s)} = \begin{bmatrix} \langle U^{(s)}\rangle_s & \mathbf{0} \\ \mathbf{0} & I \end{bmatrix}. \tag{33}$$

So

$$\langle U^{(s)}\rangle_s = \langle A\rangle_s^{-1}\langle H^{(s)}\rangle_s. \tag{34}$$

Each entry of $\langle A\rangle_s^{-1}$ is bounded by $\lambda_1/\det(\langle A\rangle_s)$ (see §1). So a typical element $U_{ij}^{(s)}$ of $\langle U^{(s)}\rangle_s$ is bounded by

$$|U_{ij}^{(s)}| \le \sum_{k=1}^{s} \frac{\lambda_1}{|\det\langle A\rangle_s|}|H_{kj}^{(s)}| \le \lambda_1,$$

since

$$\begin{aligned}\sum_{k=1}^{s}|H_{kj}^{(s)}| &\le 1+\sum_{k=j}^{s}(|H_{kk}^{(s)}|-1) &(35)\\ &\le \prod_{k=1}^{s}|H_{kk}^{(s)}| &(36)\\ &= |\det\langle A\rangle_s|.\end{aligned}$$

Inequality (35) is a consequence of Hermite normal form of $\langle H^{(s)}\rangle_s$. Inequality (36) exploits the elementary bound

$$1+\sum_{k=1}^{\ell}(x_k-1)\le\prod_{k=1}^{\ell}x_k \tag{37}$$

which holds for any $\ell\ge 1$ positive integers $x_1,\ldots,x_\ell$ (see Exercise). From (33), we infer that

$$\|U^{(s)}\|\le\|\langle U^{(s)}\rangle_s\|\le\lambda_1. \tag{38}$$

From $H^{(s)}=AU^{(s)}$ and the special form (33), we conclude

$$\|H^{(s)}\|\le\lambda_2 \tag{39}$$

where

$$\lambda_2:=\rho\lambda_0\lambda_1=(\rho\lambda_0)^{1+\rho}. \tag{40}$$

We have therefore given a bound on $\|U^{(s)}\|$ and $\|H^{(s)}\|$ for $s\le\rho$.

Now let $s>\rho$. Clearly $H^{(s)}$ still satisfies the bound (39) since in stage s we eliminated column s while the remaining columns are unchanged from $H^{(s-1)}$. We conclude that $\|H\|\le\lambda_2$ holds in *the transition* between any two successive stages.

What about $U^{(s)}$? Eliminating the (r,s)th entry amounts to adding some multiple $c_{r,s}$ of column r ($r\le\rho$) to column s. The multiple $c_{r,s}$ is bounded by λ_0. Therefore, it increases $\|U^{(s)}\|$ by a factor of (λ_0+1). We perform ρ such elimination steps to entirely eliminate column s. Thus

$$\|U^{(s)}\|\le(\lambda_0+1)^\rho\|U^{(s-1)}\|.$$

The maximum size bound is when $s=n$:

$$\|U^{(n)}\|\le(\lambda_0+1)^{\rho(n-\rho)}\|U^{(\rho)}\|\le\lambda_3, \tag{41}$$

where

$$\lambda_3:=(\lambda_0+1)^{\rho(n-\rho)}\lambda_1. \tag{42}$$

Bounds on H during a stage. What remains is to bound $\|H\|$ and $\|U\|$ during a stage. In other words, although the entries in H and U are nicely bounded between two successive stages, we must ensure that they do not swell up excessively within a stage. In our analysis, we shall use the observation that once the (s,s)th element is "fixed" at the end of the sth stage, it is hereafter bounded by λ_1. If it changes at all, it can only become smaller (replaced by a divisor). In fact, the product of all the "fixed" diagonal elements is bounded by λ_1. Let us now focus on stage s for some $s=1,\ldots,n$. Of course, the columns of interest are the first $\min\{s-1,\rho\}$ columns and column s.

ELIMINATION PHASE: let $H^{(r,s)}$ be the H matrix just before the (r,s)th entry is eliminated ($r=1,\ldots,\min\{s-1,\rho\}$). Let $h_j^{(r,s)}$ be the jth column of $H^{(r,s)}$ and $H_{i,j}^{(r,s)}$ be the (i,j)th entry of $H^{(r,s)}$. Initially, we have

$$\|h_j^{(1,s)}\|\le\begin{cases}\lambda_2, & \text{for } j=1,\ldots,\min\{s-1,\rho\},\\ \lambda_0, & \text{for } j=s.\end{cases}$$

When we eliminate the (r,s)th entry, column s is transformed (cf. equation (30)) according to the rule

$$h_s^{(r+1,s)} \leftarrow \frac{H_{r,s}^{(r,s)} h_r^{(r,s)} - H_{r,r}^{(r,s)} h_s^{(r,s)}}{g} \tag{43}$$

where $g = \mathtt{GCD}(H_{r,r}^{(r,s)}, H_{r,s}^{(r,s)})$. At the same time, column r is transformed according to the rule

$$h_r^{(r+1,s)} \leftarrow c \cdot h_r^{(r,s)} - c' \cdot h_s^{(r,s)} \tag{44}$$

where $|c| < |H_{r,s}^{(r,s)}|$ and $|c'| \le |H_{r,r}^{(r,s)}|$ (cf. equation (29)). Define

$$\beta_r := \lambda_2(\lambda_1 + \lambda_2)^{r-1}.$$

Inductively assume that column s is bounded as follows:

$$\|h_s^{(r,s)}\| \le \beta_r. \tag{45}$$

This is true for $r = 1$. From (43), we extend the inductive hypothesis to $r+1$:

$$\begin{aligned} \|h_s^{(r+1,s)}\| &\le |H_{r,s}^{(r,s)}| \cdot \|h_r^{(r,s)}\| + |H_{r,r}^{(r,s)}| \cdot \|h_s^{(r,s)}\| \\ &\le \beta_r \cdot \lambda_2 + \lambda_1 \cdot \beta_r \\ &= \beta_r(\lambda_1 + \lambda_2) = \beta_{r+1}. \end{aligned}$$

From (44), we similarly obtain a bound on column s:

$$\|h_r^{(r,s)}\| \le \beta_r. \tag{46}$$

Let $H^{(s,s)}$ be the H matrix just *after* the $(s-1,s)$th entry of H is eliminated. Then the bounds (45) and (46) extend to

$$\|h_{s-1}^{(s,s)}\| \le \beta_s, \qquad \|h_s^{(s,s)}\| \le \beta_s.$$

We conclude that throughout an elimination phase, each entry of H is bounded by

$$\beta_s = \lambda_2(\lambda_1 + \lambda_2)^{s-1} < \lambda_4$$

where

$$\lambda_4 := (2\lambda_2)^{\rho} = 2^{\rho}(\rho\lambda_0)^{\rho(1+\rho)}. \tag{47}$$

REDUCTION PHASE: So $H^{(s,s)}$ is the H matrix just before the start of the reduction phase. Note that we may assume $s \le \rho$. Let $h_j^{(s)}$ be the jth column of $H^{(s,s)}$. Also let $\widehat{h}_j^{(s)}$ be the jth column at the *end* of the reduction phase: these are called the "reduced vectors". Note that the reduced vectors are columns of $H^{(s)}$ and hence satisfy

$$\|\widehat{h}_j^{(s)}\| \le \lambda_2. \tag{48}$$

Exploiting the special sequencing of reduction steps in the algorithm (following Chou-Collins), we see that

$$\widehat{h}_j^{(s)} = h_j^{(s)} - \sum_{r=j+1}^{s} b_{r,j,s} \widehat{h}_r^{(s)} \tag{49}$$

for suitable constants $b_{r,j,s}$. The point is that reduced vectors appear on the right-hand side of (49). The entries of column j in H are bounded by λ_4 at the start of the reduction phase. To reduce column j ($j = 1, \ldots, s-1$), we first reduce its $j+1$st entry by subtracting the column $b_{j+1,j,s}\widehat{h}_{j+1}^{(s)}$ (cf. equation (49)). Clearly $b_{j+1,j,s} \le \lambda_4$ so that entries of column j are bounded by $\lambda_4(1+\lambda_2)$ after this reduction step. Inductively, it is easy to see that after the $(j+k)$th ($k = 1, 2, \ldots$) entry of column j is reduced, the entries of column j are bounded by

$$\lambda_4(1+\lambda_2)^k.$$

So just after the $s-1$st entry is reduced, its entries are bounded by

$$\lambda_4(1+\lambda_2)^{s-1-j} < \lambda_4(2\lambda_2)^{\rho} = \lambda_4^2.$$

Finally, we reduce the sth entry of column j. But the result is a reduced column bounded as in (48). It follows that the bound

$$\|H\| \le \lambda_4^2$$

holds throughout the stage.

Bounds on U during a stage. First assume $s \le \rho$. It suffices to use the bound

$$\|U\| \le n\|A^{-1}\| \cdot \|H\| \le n\lambda_1\lambda_4^2$$

since the relation $U = A^{-1}H$ holds throughout the stage. Suppose $s > \rho$. There is no reduction phase and the argument showing $\|U^{(s)}\| \le \lambda_3$ in (41) actually shows that $\|U\| \le \lambda_3$ *throughout* stage s. This concludes our analysis.

We summarize the foregoing analysis: the entries of H and U matrices are uniformly bounded by

$$\lambda_5 := \lambda_3 + \lambda_4 < \lambda_1((2\lambda_0)^{\rho(n-\rho)} + n\lambda_4^2)$$

throughout the algorithm. Since $L = \lg \lambda_0$, we get

$$L' := \lg \lambda_5 = O(\rho n L + \rho^2 \lg \rho)$$

as a bound on the bit-size of entries. There are $O(\rho^2 n)$ column operations and $O(\rho^2)$ co-GCD operations on matrix entries. Each column operation takes $O(m)$ ring operations on matrix entries. Hence the cost of co-GCD operations is dominated by the cost of column operations, which amounts to:

Theorem 18 *With $L = \lg \lambda_0$, the matrix entries have bit-size that are uniformly bounded by*

$$L' = O(\rho[nL + \rho \lg \rho)])$$

in the Bachem-Kannan algorithm. The bit-complexity of the algorithm is $O(mn\rho^2 \mathrm{M}_B(L'))$.

Remarks. Our complexity analysis is somewhat more involved than that in Bachem and Kannan [7], in part because the input matrix is non-square and may have dependent rows and columns. For instance, if $n = \rho$ then $\lg(\lambda_3) = O(n(L + \lg n)$ and not $O(n^2 L)$.

Suppose we want to compute the HNF $H(A)$ of an arbitrary matrix A. Accordingly, we submit A to the Extended Bareiss algorithm (§2) to obtain $B = PAQ$ where P and Q are permutation matrices. Now we submit B to the Bachem-Kannan algorithm which outputs $H = H(B)$, the HNF of B. We leave it as an exercise to show:

$$H(A) = P^{-1}H. \tag{50}$$

Exercises

Exercise 7.1: (Chou-Collins) Verify inequality (37) □

Exercise 7.2: Show equation (50). HINT: this depends on the particular structure of our Extended Bareiss algorithm. □

Exercise 7.3: Analyze the algorithm assuming A is square and non-singular. □

§8. Smith Normal Form

H. J. S. Smith (1861) introduced the normal form bearing his name. Let $A \in \mathbb{Z}^{m\times n}$. We say A is in *Smith normal form* (SNF) if it is diagonal, that is, $(A)_{i,j} = 0$ for $i \neq j$, and its diagonal entries are non-negative satisfying

$$(A)_{i-1,i-1} | (A)_{i,i}, \qquad \text{for } \; i = 2, \ldots, \min\{m, n\}. \tag{51}$$

Since every number divides 0, but 0 divides only itself, we conclude from (51) that if $(A)_{i,i} = 0$ for some i, then $(A)_{j,j} = 0$ for all $j > i$. The multi-set of non-zero diagonal entries of a Smith normal form matrix A,

$$\{(A)_{1,1}, (A)_{2,2}, \ldots, (A)_{r,r}\}$$

where $(A)_{r,r} > 0$ and $(A)_{r+1,r+1} = 0$, is called the *set of Smith invariants* of A. We also call $(A)_{i,i}$ the ith Smith invariant ($i = 1, \ldots, r$). [In the literature, the Smith invariants of A are also called "invariant factors" of A.]

By *elementary operations* in this section, we mean elementary integer row or column operations. We say that two matrices are *equivalent* if they are inter-transformable by elementary operations.

Lemma 19 *Every integer matrix A can be brought into a Smith normal form S by a sequence of elementary operations.*

We leave the proof to an Exercise. The algorithm below implies this, of course, but it is instructive for the student to give a direct proof. We will show that S is unique for A, and so S is *the* Smith normal form of A, usually denoted $S(A)$. For $k = 1, \ldots, \min\{m, n\}$, let $\delta_k(A)$ denote the GCD of all the order k minors of A. In particular, $\delta_1(A)$ is the GCD of all the entries of A.

Lemma 20

(i) *The elementary operations on a matrix A preserve $\delta_k(A)$.*

(ii) *The set of Smith invariants of A is unique.*

(iii) *The Smith normal form of A is unique.*

Proof. (i) This is immediate from our treatment of the γ-invariants for the HNF.
(ii) Let the rank of A be r. Then $\delta_k(A) \neq 0$ iff $k = 1, \ldots, r$. From the definition of the δ's, it is clear that

$$\delta_k(A) | \delta_{k+1}(A)$$

for $k = 1, \ldots, r-1$. Let S be a Smith normal form of A. Since A and S have the same rank, we conclude that $(S)_{k,k} \neq 0$ iff $k = 1, \ldots, r$. Note that $\delta_k(S) = (S)_{1,1}(S)_{2,2}\cdots(S)_{k,k}$. In view of part (i), we conclude

$$(S)_{1,1}(S)_{2,2}\cdots(S)_{k,k} = \delta_k(A).$$

It follows that $(S)_{1,1} = \delta_1(A)$ and $(S)_{k+1,k+1} = \delta_{k+1}(A)/\delta_k(A)$ $(k = 2, \ldots, r-1)$.
(iii) This follows from (ii) since a Smith normal form is determined by the set of its Smith invariants.
Q.E.D.

Polynomial-time Algorithm. Computing the Smith normal form of a matrix A is equivalent to computing the set of Smith invariants. For some applications, it is desirable to also know the unimodular matrices U, V such that

$$S(A) = UAV.$$

For instance, it is easy to compute the 1st Smith invariant – it is just the GCD of all the matrix entries. But computing U and V such that $(UAV)_{1,1}$ has this invariant is non-trivial. As usual, the difficulty in giving a polynomial time algorithm is the possibility of exponential size intermediate entries. We describe an algorithm from Bachem-Kannan [7], based on a Hermite normal form algorithm.

It suffices to show how to perform a *Smith Reduction Step* (in analogy to the Hermite Reduction Step of §6): given a matrix $A \in \mathbb{Z}^{m\times n}$, compute two unimodular matrices $U \in \mathbb{Z}^{m\times m}, V \in \mathbb{Z}^{n\times n}$ such that UAV is "Smith-reduced". In general, a matrix M is said to be *Smith-reduced* if:
(i) The first row and first column of M are zero except for the top-left corner entry $(M)_{1,1}$.
(ii) $(M)_{1,1}$ divides all the remaining entries of M.

Our Hermite normal form is based on column operations. We now need the "row version" of the normal form: a matrix A is in *row Hermite normal form* (abbr. RHNF) if its transpose A^T is in Hermite normal form. Using elementary row operations, or multiplication by unimodular matrices on the left, we can transform any matrix into its RHNF. Algorithms for RHNF are trivially obtained from HNF algorithms, simply by interchanging the roles of rows and columns.

```
Smith Reduction Step
    Input:   $A \in \mathbb{Z}^{m\times n}$. Assume $(A)_{1,1} \neq 0$.
    Output:  Unimodular matrices $U \in \mathbb{Z}^{m\times m}$ and $V \in \mathbb{Z}^{n\times n}$, and matrix $S$
                 such that $S$ is Smith-reduced and HNF, and $S = UAV$.
INITIALIZATION:
    1.  $S \leftarrow A$.
    2.  $U \leftarrow I_m$; $V \leftarrow I_n$ (identity matrices).
MAIN LOOP:
        {Loop Invariant: $S = UAV$}
    3.  while $S$ is not Smith-reduced do
    4.      if $(S)_{1,1}$ is the only non-zero entry in the first row
                and first column, then $(S)_{1,1}$ does not divide some $(S)_{i,j}$.
    5.          In this case, add column $j$ to column 1, and update $V$.
    6.      Apply a RHNF algorithm to $S$ and update $U$ accordingly.
    7.      Apply a HNF algorithm to $S$ and update $V$ accordingly.
```

Analysis. The bit-sizes of entries remain polynomially bounded between successive while-iterations: this is because S is in HNF on exit from an iteration, and so the largest entry lies on the diagonal. But the diagonal entries of S are bounded by the determinant of the input matrix A, since the elementary operations preserve $\delta_k(A)$ for each k. Since the RHNF and HNF algorithms are polynomial time, we conclude that each while-iteration is polynomial-time.

It remains to show that the number of iterations is polynomial. Note that whenever $(S)_{1,1}$ changes after an iteration, it is being replaced by a factor of itself. We claim that $(S)_{1,1}$ must change at least in every other iteration. To see this, consider the two possibilities: either step 5 (adding column j to column 1) is executed or it is not. *Step 5 is executed*: then column 1 contains an entry not divisible by $(S)_{1,1}$ before the RHNF algorithm. After the RHNF algorithm, $(S)_{1,1}$ will contain the the GCD of entries in column 1, and this will be a proper factor of its previous value. *Step 5 is not executed*: then row 1 or column 1 must have at least some other non-zero entry. Again there are two cases. (i) If all of these non-zero entries are divisible by $(S)_{1,1}$ then after this iteration, $(S)_{1,1}$ is the only non-zero entry in row 1 and column 1. If this is not the last iteration, then we already saw that $(S)_{1,1}$ will change in the next iteration. (ii) If there is an entry that is not divisible by $(S)_{1,1}$ then either the RHNF or HNF algorithm will change $(S)_{1,1}$ to a smaller value. Hence the number of iterations is at most $1 + 2\lg\|A\|$.

One other remark: the output matrices U, V are of polynomial bit-size. This is because each matrix is the product of $O(\lg\|A\|)$ many component matrices (produced by the call to HNF or RHNF or by Step 5). But each component matrix is of polynomial bit-size. This concludes our analysis of the Smith Reduction Step algorithm. Clearly the Smith normal form of an $m \times n$ matrix can be reduced to $\min\{m, n\}$ Smith Reduction steps. Moreover, the result of Smith Reduction Step is an HNF, and hence polynomially bounded in terms of the original input. This proves:

Theorem 21 *There is a polynomial-time algorithm to compute the Smith normal form $S(A)$ of a matrix A. This algorithm simultaneously computes two unimodular matrices U, V such that $S(A) = UAV$.*

Exercises

Exercise 8.1: Show lemma 19 by a direct argument (*i.e.*, without reference to the existence of the SNF algorithm). □

Exercise 8.2: Let d_i $(i = 1, \ldots, r)$ be the ith Smith invariant of A. Write

$$d_i = p_1^{e_{i,1}} p_2^{e_{i,2}}, \ldots, p_{\ell_i}^{e_{i,\ell_i}}, \qquad (\ell_i \geq 1)$$

where p_i is the ith prime number. Call the prime power $p_j^{e_{i,j}}$ an *elementary divisor* of A. Show that two matrices are equivalent iff they have the same rank and the same set of elementary divisors. □

Exercise 8.3: Analyze the complexity of the Smith Reduction Step and the associated SNF algorithm. □

Exercise 8.4: Let $A, B, C \in \mathbb{Z}^{n\times n}$. We say A is *irreducible* if, whenever $A = BC$ then either B or C is unimodular. Otherwise A is *reducible*. If $A = BC$, we call C a *right divisor* of A or C *right-divides* A. Write $C|A$ in this case. Similarly, there is a notion of *left divisor*.
(i) An irreducible matrix is equivalent to $\mathtt{diag}(1, \ldots, 1, p)$, the diagonal matrix whose main diagonal has all ones except the last entry, which is a prime p.
(ii) A necessary and sufficient condition for a square matrix to be irreducible is that its determinant is prime.
(iii) A reducible matrix can be factored into a finite product of irreducible matrices. Formulate a uniqueness property for this factorization. □

Exercise 8.5: Let $A, B, C, D \in \mathbb{Z}^{n\times n}$. Assume A and B are not both zero. Call D a (right) *greatest common divisor* (GCD) of A, B if $D|A$ and $D|B$ and for any other C that divides both A and B, then $C|D$. (See definitions in previous exercise.)
(i) Show that GCDs of A, B exist. Moreover, if D is a GCD then $D = PA + QB$ for some P, Q. HINT: consider the SNF of $[A|B]$.
(ii) If D, D' are two GCD's of A then $D = UD'$ for some unimodular matrix U. □

§9. Further Applications

Linear Diophantine equations. Let $A \in \mathbb{Z}^{m\times n}$ and $b = (b_1, \ldots, b_n) \in \mathbb{Z}^{1\times n}$. Consider the problem of solving the linear system

$$x \cdot A = b \tag{52}$$

for an unknown $x = (x_1, \ldots, x_m) \in \mathbb{Z}^{1\times m}$. Such a system is also called a *Diophantine linear system.* For simplicity, assume $m \geq n$; otherwise, the n equations in (52) are not independent and some may be omitted. Let $S = UAV$ be the Smith normal form of A and let the diagonal elements of S be $d_1, \ldots, d_n$. Then (52) implies

$$\begin{aligned}(xU^{-1})(UAV) &= bV,\\ \widehat{x}S &= \widehat{b}.\end{aligned}$$

where $\widehat{x} = (\widehat{x}_1, \ldots, \widehat{x}_m) = xU^{-1}$ and $\widehat{b} = (\widehat{b}_1, \ldots, \widehat{b}_n) = bV$. The last equation amounts to

$$\widehat{x}_i d_i = \widehat{b}_i, \qquad i = 1, \ldots, n. \tag{53}$$

Suppose $d_1, \ldots, d_r$ are non-zero and $d_{r+1} = \cdots = d_n = 0$. Then the system (52) has solution iff
(i) $d_i|\widehat{b}_i$ for $i = 1, \ldots, r$ and
(ii) $\widehat{b}_i = 0$ for $i = r + 1, \ldots, n$.
If these conditions are satisfied then a *general solution* to (53) is given by setting $\widehat{x}_i = \widehat{b}_i/d_i$ for $i = 1, \ldots, r$ and arbitrary assignments to $\widehat{x}_i$ for $i = r + 1, \ldots, m$. For instance, we may choose

$$\widehat{x} = (\widehat{b}_1/d_1, \ldots, \widehat{b}_r/d_r, 0, \ldots, 0).$$

From any such *particular solution* we obtain a solution $x = \widehat{x}U$ to the original system (52).

Homogeneous Case. The important special case of (52) where $b = \mathbf{0}$ is said to be *homogeneous.* A solution to this homogeneous system $xA = \mathbf{0}$ is called a *null-vector* of A. The set $N(A) \subseteq \mathbb{Z}^m$ of these null-vectors forms a $\mathbb{Z}$-module. By the Hilbert basis theorem for modules (see Lecture XI), $N(A)$ has a finite basis. Now $e_{r+1}, \ldots, e_m$ is a basis for the set of solutions to (53) in the homogeneous case. Here e_i denotes the m-vector with 1 in the ith position and 0 everywhere else. We conclude that the set

$$e_{r+1}U, e_{r+2}U, \ldots, e_mU$$

is a basis for $N(A)$. Therefore, we may consider a *complete solution* of (52) to have the form $(\widehat{x}U, e_{r+1}U, \ldots, e_mU)$ where $\widehat{x}$ is any particular solution to (53). Notice that $N(A)$ is a lattice, which we may regard as the "dual" of the lattice $\Lambda(A)$ (§VIII.1).

If $A \in \mathbb{Z}^{m\times n}$ and $m > n$ then we have just shown that the homogeneous Diophantine system $x \cdot A = \mathbf{0}$ has non-trivial solutions. An interesting question is whether there exist small integer solutions. Siegel (1929) shows: *there is an $x \in N(A)$ satisfying*

$$\|x\| < 1 + (m\|A\|)^{n/(m-n)}. \tag{54}$$

We leave the demonstration to an exercise.

Finitely-generated Abelian groups. Smith normal forms are intimately related to the theory of finitely generated Abelian groups. Let G be a *finitely-presented* Abelian group, that is, G is represented by n *generator* $x_1, \ldots, x_n$ and m *relations* of the form

$$\sum_{j=1}^{n} a_{ij} x_j = 0, \qquad (a_{ij} \in \mathbb{Z})$$

for $i = 1, \ldots, m$. (Note the convention of writing the operations of an Abelian group "additively".) The corresponding *relations matrix* is an $m \times n$ integer matrix A where $a_{ij} = (A)_{i,j}$. We may rewrite the relations in the form $Ax = \mathbf{0}$ where $x = (x_1, \ldots, x_n)^T$. In the special case where $m = 0$ (alternatively, the matrix A is all zero) deserves mention: the group G in this case is called the *free Abelian group of rank n*. Clearly, $G \approx \mathbb{Z}^n$ where $\approx$ indicates group isomorphism.

Let the Smith normal form of A be $S = S(A) = UAV$ for some unimodular matrices U, V. This amounts to transforming the generators of G to $(y_1, \ldots, y_n)^T = V^{-1}(x_1, \ldots, x_n)^T$. Then $S \cdot (y_1, \ldots, y_n)^T = \mathbf{0}$. If S has rank r and the diagonal elements of S are $d_1, \ldots, d_{\min(m,n)}$ then we see that $d_i y_i = 0$ for $i = 1, \ldots, r$ and $y_{r+1}, \ldots, y_n$ satisfy no relations whatsoever. Each y_i generates the subgroup $G_i = \mathbb{Z} y_i$ of G. Moreover, $G_i \approx \mathbb{Z}_{d_i}$ for $i = 1, \ldots, r$ and $G_i \approx \mathbb{Z}$ for $i = r+1, \ldots, n$. Clearly G is a direct sum of the G_i's: $G = \oplus_{i=1}^n G_i$. There are three kinds of subgroups:
$d_i = 0$: these correspond to torsion-free subgroups $G_i \approx \mathbb{Z}$. The number β of these subgroups is called the *Betti number* of G.
$d_i = 1$: these are trivial subgroups, and may be omitted in the direct sum expression.
$d_i \geq 2$: these give rise to finite cyclic groups G_i. These d_i's are called *torsion coefficients* of G.

We have just proven the "fundamental theorem of finitely generated Abelian groups": *every finitely presented Abelian group G on n generators can be written as a direct sum $G = \oplus_{i=0}^r H_i$ where H_0 is a free Abelian group of rank β, and each H_i $(i = 1, \ldots, r)$ is a finite cyclic group of order $d_i \geq 2$ satisfying such that $d_1 | d_2 | \cdots | d_r$. The numbers $\beta, d_1, \ldots, d_r$ are uniquely determined by G.*

It follows that a polynomial time algorithm for SNF implies that we can check for isomorphism between two finitely generated Abelian groups in polynomial time. A slightly different group isomorphism problem arises if we assume that finite groups are represented by their multiplication tables instead of by a set of relations. An observation of Tarjan implies that we can check isomorphism of two such groups in $O(n^{\log n + O(1)})$ time. For the Abelian case, Vikas [212] has shown an $O(n \log n)$ isomorphism algorithm.

Exercises

Exercise 9.1: Let $a, b, c \in \mathbb{Z}$ where a, b are relatively prime. Suppose $sa + tb = 1$. Show that the general solution of the Diophantine equation $ax + by = c$ is $(x, y) = (sc + nb, tc - na)$ where n is any parameter. □

Exercise 9.2: Consider $N(A)$ in case $n = 1$. Say $A = (a_1, \ldots, a_m)^T$.
(i) Let $s = (s_1, \ldots, s_m)$ be a co-factor of $(a_1, \ldots, a_m)$. Show that $\mathbb{Z} \cdot N(A) + \mathbb{Z} \cdot s$ is the unit lattice $\mathbb{Z}^m$.
(ii) For $1 \leq i < j \leq m$ let $T(i, j)$ be the m-vector that is zero everywhere except $a_i/\mathtt{GCD}(a_i, a_j)$ at the jth position and $-a_j/gcd(a_i, a_j)$ at the ith position. The set of these $T(i, j)$'s generates $N(A)$. □

Exercise 9.3: (i) Show the bound of Siegel (54). HINT: for H a parameter to be chosen, let $C = C_H$ be the cube comprising points $x \in \mathbb{R}^m$ where $\|x\| \leq H$. Let $\alpha : \mathbb{R}^m \to \mathbb{R}^n$ be the

linear map given by $\alpha(x) = x \cdot A$. Give a cube C' in $\mathbb{R}^n$ the contains $\alpha(C)$. Use a pigeon hole argument to show that α is not $1-1$ on the integer points of C. See [179] for several versions of Siegel's bound.
(ii) Show that the exponent $n/(m-n)$ cannot be improved. □

Exercise 9.4: Show:
(i) If d_1, d_2 are co-prime then $\mathbb{Z}_{d_1} \oplus \mathbb{Z}_{d_2} \approx \mathbb{Z}_{d_1 d_2}$.
(ii) Every finite cyclic group is a direct sum of cyclic groups of prime power.
(iii) Every finitely generated Abelian group written as a direct sum $G = \oplus_{i=0}^{\ell} H_i$ where H_0 is a free Abelian group of rank β, and $H_i \approx \mathbb{Z}_{q_i}$ where q_i is a prime power ($i = 1, \ldots, \ell$). Moreover, the numbers $\beta, q_1, \ldots, q_\ell$ are uniquely determined by G. (These q_i's are called the *invariant factors* of G.) □

References

[1] W. W. Adams and P. Loustaunau. *An Introduction to Gröbner Bases.* Graduate Studies in Mathematics, Vol. 3. American Mathematical Society, Providence, R.I., 1994.

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

[3] S. Akbulut and H. King. *Topology of Real Algebraic Sets.* Mathematical Sciences Research Institute Publications. Springer-Verlag, Berlin, 1992.

[4] E. Artin. *Modern Higher Algebra (Galois Theory).* Courant Institute of Mathematical Sciences, New York University, New York, 1947. (Notes by Albert A. Blank).

[5] E. Artin. *Elements of algebraic geometry.* Courant Institute of Mathematical Sciences, New York University, New York, 1955. (Lectures. Notes by G. Bachman).

[6] M. Artin. *Algebra.* Prentice Hall, Englewood Cliffs, NJ, 1991.

[7] A. Bachem and R. Kannan. Polynomial algorithms for computing the Smith and Hermite normal forms of an integer matrix. *SIAM J. Computing*, 8:499–507, 1979.

[8] C. Bajaj. Algorithmic implicitization of algebraic curves and surfaces. Technical Report CSD-TR-681, Computer Science Department, Purdue University, November, 1988.

[9] C. Bajaj, T. Garrity, and J. Warren. On the applications of the multi-equational resultants. Technical Report CSD-TR-826, Computer Science Department, Purdue University, November, 1988.

[10] E. F. Bareiss. Sylvester's identity and multistep integer-preserving Gaussian elimination. *Math. Comp.*, 103:565–578, 1968.

[11] E. F. Bareiss. Computational solutions of matrix problems over an integral domain. *J. Inst. Math. Appl.*, 10:68–104, 1972.

[12] D. Bayer and M. Stillman. A theorem on refining division orders by the reverse lexicographic order. *Duke Math. J.*, 55(2):321–328, 1987.

[13] D. Bayer and M. Stillman. On the complexity of computing syzygies. *J. of Symbolic Computation*, 6:135–147, 1988.

[14] D. Bayer and M. Stillman. Computation of Hilbert functions. *J. of Symbolic Computation*, 14(1):31–50, 1992.

[15] A. F. Beardon. *The Geometry of Discrete Groups.* Springer-Verlag, New York, 1983.

[16] B. Beauzamy. Products of polynomials and a priori estimates for coefficients in polynomial decompositions: a sharp result. *J. of Symbolic Computation*, 13:463–472, 1992.

[17] T. Becker and V. Weispfenning. *Gröbner bases : a Computational Approach to Commutative Algebra.* Springer-Verlag, New York, 1993. (written in cooperation with Heinz Kredel).

[18] M. Beeler, R. W. Gosper, and R. Schroepppel. HAKMEM. A. I. Memo 239, M.I.T., February 1972.

[19] M. Ben-Or, D. Kozen, and J. Reif. The complexity of elementary algebra and geometry. *J. of Computer and System Sciences*, 32:251–264, 1986.

[20] R. Benedetti and J.-J. Risler. *Real Algebraic and Semi-Algebraic Sets.* Actualités Mathématiques. Hermann, Paris, 1990.

[21] S. J. Berkowitz. On computing the determinant in small parallel time using a small number of processors. *Info. Processing Letters*, 18:147–150, 1984.

[22] E. R. Berlekamp. *Algebraic Coding Theory.* McGraw-Hill Book Company, New York, 1968.

[23] J. Bochnak, M. Coste, and M.-F. Roy. *Geometrie algebrique reelle.* Springer-Verlag, Berlin, 1987.

[24] A. Borodin and I. Munro. *The Computational Complexity of Algebraic and Numeric Problems.* American Elsevier Publishing Company, Inc., New York, 1975.

[25] D. W. Boyd. Two sharp inequalities for the norm of a factor of a polynomial. *Mathematika*, 39:341–349, 1992.

[26] R. P. Brent, F. G. Gustavson, and D. Y. Y. Yun. Fast solution of Toeplitz systems of equations and computation of Padé approximants. *J. Algorithms*, 1:259–295, 1980.

[27] J. W. Brewer and M. K. Smith, editors. *Emmy Noether: a Tribute to Her Life and Work.* Marcel Dekker, Inc, New York and Basel, 1981.

[28] C. Brezinski. *History of Continued Fractions and Padé Approximants.* Springer Series in Computational Mathematics, vol.12. Springer-Verlag, 1991.

[29] E. Brieskorn and H. Knörrer. *Plane Algebraic Curves.* Birkhäuser Verlag, Berlin, 1986.

[30] W. S. Brown. The subresultant PRS algorithm. *ACM Trans. on Math. Software*, 4:237–249, 1978.

[31] W. D. Brownawell. Bounds for the degrees in Nullstellensatz. *Ann. of Math.*, 126:577–592, 1987.

[32] B. Buchberger. Gröbner bases: An algorithmic method in polynomial ideal theory. In N. K. Bose, editor, *Multidimensional Systems Theory*, Mathematics and its Applications, chapter 6, pages 184–229. D. Reidel Pub. Co., Boston, 1985.

[33] B. Buchberger, G. E. Collins, and R. L. (eds.). *Computer Algebra.* Springer-Verlag, Berlin, 2nd edition, 1983.

[34] D. A. Buell. *Binary Quadratic Forms: classical theory and modern computations.* Springer-Verlag, 1989.

[35] W. S. Burnside and A. W. Panton. *The Theory of Equations*, volume 1. Dover Publications, New York, 1912.

[36] J. F. Canny. *The complexity of robot motion planning.* ACM Doctoral Dissertion Award Series. The MIT Press, Cambridge, MA, 1988. PhD thesis, M.I.T.

[37] J. F. Canny. Generalized characteristic polynomials. *J. of Symbolic Computation*, 9:241–250, 1990.

[38] D. G. Cantor, P. H. Galyean, and H. G. Zimmer. A continued fraction algorithm for real algebraic numbers. *Math. of Computation*, 26(119):785–791, 1972.

[39] J. W. S. Cassels. *An Introduction to Diophantine Approximation.* Cambridge University Press, Cambridge, 1957.

[40] J. W. S. Cassels. *An Introduction to the Geometry of Numbers.* Springer-Verlag, Berlin, 1971.

[41] J. W. S. Cassels. *Rational Quadratic Forms.* Academic Press, New York, 1978.

[42] T. J. Chou and G. E. Collins. Algorithms for the solution of linear Diophantine equations. *SIAM J. Computing*, 11:687–708, 1982.

[43] H. Cohen. *A Course in Computational Algebraic Number Theory.* Springer-Verlag, 1993.

[44] G. E. Collins. Subresultants and reduced polynomial remainder sequences. *J. of the ACM*, 14:128–142, 1967.

[45] G. E. Collins. Computer algebra of polynomials and rational functions. *Amer. Math. Monthly*, 80:725–755, 1975.

[46] G. E. Collins. Infallible calculation of polynomial zeros to specified precision. In J. R. Rice, editor, *Mathematical Software III*, pages 35–68. Academic Press, New York, 1977.

[47] J. W. Cooley and J. W. Tukey. An algorithm for the machine calculation of complex Fourier series. *Math. Comp.*, 19:297–301, 1965.

[48] D. Coppersmith and S. Winograd. Matrix multiplication via arithmetic progressions. *J. of Symbolic Computation*, 9:251–280, 1990. Extended Abstract: ACM Symp. on Theory of Computing, Vol.19, 1987, pp.1-6.

[49] M. Coste and M. F. Roy. Thom's lemma, the coding of real algebraic numbers and the computation of the topology of semi-algebraic sets. *J. of Symbolic Computation*, 5:121–130, 1988.

[50] D. Cox, J. Little, and D. O'Shea. *Ideals, Varieties and Algorithms: An Introduction to Computational Algebraic Geometry and Commutative Algebra.* Springer-Verlag, New York, 1992.

[51] J. H. Davenport, Y. Siret, and E. Tournier. *Computer Algebra: Systems and Algorithms for Algebraic Computation.* Academic Press, New York, 1988.

[52] M. Davis. *Computability and Unsolvability.* Dover Publications, Inc., New York, 1982.

[53] M. Davis, H. Putnam, and J. Robinson. The decision problem for exponential Diophantine equations. *Annals of Mathematics, 2nd Series*, 74(3):425–436, 1962.

[54] J. Dieudonné. *History of Algebraic Geometry.* Wadsworth Advanced Books & Software, Monterey, CA, 1985. Trans. from French by Judith D. Sally.

[55] L. E. Dixon. Finiteness of the odd perfect and primitive abundant numbers with n distinct prime factors. *Amer. J. of Math.*, 35:413–426, 1913.

[56] T. Dubé, B. Mishra, and C. K. Yap. Admissible orderings and bounds for Gröbner bases normal form algorithm. Report 88, Courant Institute of Mathematical Sciences, Robotics Laboratory, New York University, 1986.

[57] T. Dubé and C. K. Yap. A basis for implementing exact geometric algorithms (extended abstract), September, 1993. Paper from URL `http://cs.nyu.edu/cs/faculty/yap`.

[58] T. W. Dubé. *Quantitative analysis of problems in computer algebra: Gröbner bases and the Nullstellensatz.* PhD thesis, Courant Institute, N.Y.U., 1989.

[59] T. W. Dubé. The structure of polynomial ideals and Gröbner bases. *SIAM J. Computing*, 19(4):750–773, 1990.

[60] T. W. Dubé. A combinatorial proof of the effective Nullstellensatz. *J. of Symbolic Computation*, 15:277–296, 1993.

[61] R. L. Duncan. Some inequalities for polynomials. *Amer. Math. Monthly*, 73:58–59, 1966.

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

[63] H. M. Edwards. *Divisor Theory.* Birkhauser, Boston, 1990.

[64] I. Z. Emiris. *Sparse Elimination and Applications in Kinematics.* PhD thesis, Department of Computer Science, University of California, Berkeley, 1989.

[65] W. Ewald. *From Kant to Hilbert: a Source Book in the Foundations of Mathematics.* Clarendon Press, Oxford, 1996. In 3 Volumes.

[66] B. J. Fino and V. R. Algazi. A unified treatment of discrete fast unitary transforms. *SIAM J. Computing*, 6(4):700–717, 1977.

[67] E. Frank. Continued fractions, lectures by Dr. E. Frank. Technical report, Numerical Analysis Research, University of California, Los Angeles, August 23, 1957.

[68] J. Friedman. On the convergence of Newton's method. *Journal of Complexity*, 5:12–33, 1989.

[69] F. R. Gantmacher. *The Theory of Matrices, volume 1.* Chelsea Publishing Co., New York, 1959.

[70] I. M. Gelfand, M. M. Kapranov, and A. V. Zelevinsky. *Discriminants, Resultants and Multidimensional Determinants.* Birkhäuser, Boston, 1994.

[71] M. Giusti. Some effectivity problems in polynomial ideal theory. In *Lecture Notes in Computer Science*, volume 174, pages 159–171, Berlin, 1984. Springer-Verlag.

[72] A. J. Goldstein and R. L. Graham. A Hadamard-type bound on the coefficients of a determinant of polynomials. *SIAM Review*, 16:394–395, 1974.

[73] H. H. Goldstine. *A History of Numerical Analysis from the 16th through the 19th Century.* Springer-Verlag, New York, 1977.

[74] W. Gröbner. *Moderne Algebraische Geometrie.* Springer-Verlag, Vienna, 1949.

[75] M. Grötschel, L. Lovász, and A. Schrijver. *Geometric Algorithms and Combinatorial Optimization.* Springer-Verlag, Berlin, 1988.

[76] W. Habicht. Eine Verallgemeinerung des Sturmschen Wurzelzählverfahrens. *Comm. Math. Helvetici*, 21:99–116, 1948.

[77] J. L. Hafner and K. S. McCurley. Asymptotically fast triangularization of matrices over rings. *SIAM J. Computing*, 20:1068–1083, 1991.

[78] G. H. Hardy and E. M. Wright. *An Introduction to the Theory of Numbers.* Oxford University Press, New York, 1959. 4th Edition.

[79] P. Henrici. *Elements of Numerical Analysis.* John Wiley, New York, 1964.

[80] G. Hermann. Die Frage der endlich vielen Schritte in der Theorie der Polynomideale. *Math. Ann.*, 95:736–788, 1926.

[81] N. J. Higham. *Accuracy and stability of numerical algorithms.* Society for Industrial and Applied Mathematics, Philadelphia, 1996.

[82] C. Ho. Fast parallel gcd algorithms for several polynomials over integral domain. Technical Report 142, Courant Institute of Mathematical Sciences, Robotics Laboratory, New York University, 1988.

[83] C. Ho. *Topics in algebraic computing: subresultants, GCD, factoring and primary ideal decomposition.* PhD thesis, Courant Institute, New York University, June 1989.

[84] C. Ho and C. K. Yap. The Habicht approach to subresultants. *J. of Symbolic Computation*, 21:1–14, 1996.

[85] A. S. Householder. *Principles of Numerical Analysis.* McGraw-Hill, New York, 1953.

[86] L. K. Hua. *Introduction to Number Theory.* Springer-Verlag, Berlin, 1982.

[87] A. Hurwitz. Über die Trägheitsformem eines algebraischen Moduls. *Ann. Mat. Pura Appl.*, 3(20):113–151, 1913.

[88] D. T. Huynh. A superexponential lower bound for Gröbner bases and Church-Rosser commutative Thue systems. *Info. and Computation*, 68:196–206, 1986.

[89] C. S. Iliopoulous. Worst-case complexity bounds on algorithms for computing the canonical structure of finite Abelian groups and Hermite and Smith normal form of an integer matrix. *SIAM J. Computing*, 18:658–669, 1989.

[90] N. Jacobson. *Lectures in Abstract Algebra, Volume 3.* Van Nostrand, New York, 1951.

[91] N. Jacobson. *Basic Algebra 1.* W. H. Freeman, San Francisco, 1974.

[92] T. Jebelean. An algorithm for exact division. *J. of Symbolic Computation*, 15(2):169–180, 1993.

[93] M. A. Jenkins and J. F. Traub. Principles for testing polynomial zerofinding programs. *ACM Trans. on Math. Software*, 1:26–34, 1975.

[94] W. B. Jones and W. J. Thron. *Continued Fractions: Analytic Theory and Applications.* vol. 11, Encyclopedia of Mathematics and its Applications. Addison-Wesley, 1981.

[95] E. Kaltofen. Effective Hilbert irreducibility. *Information and Control*, 66(3):123–137, 1985.

[96] E. Kaltofen. Polynomial-time reductions from multivariate to bi- and univariate integral polynomial factorization. *SIAM J. Computing*, 12:469–489, 1985.

[97] E. Kaltofen. Polynomial factorization 1982-1986. Dept. of Comp. Sci. Report 86-19, Rensselaer Polytechnic Institute, Troy, NY, September 1986.

[98] E. Kaltofen and H. Rolletschek. Computing greatest common divisors and factorizations in quadratic number fields. *Math. Comp.*, 52:697–720, 1989.

[99] R. Kannan, A. K. Lenstra, and L. Lovász. Polynomial factorization and nonrandomness of bits of algebraic and some transcendental numbers. *Math. Comp.*, 50:235–250, 1988.

[100] H. Kapferer. Über Resultanten und Resultanten-Systeme. *Sitzungsber. Bayer. Akad. München*, pages 179–200, 1929.

[101] A. N. Khovanskii. *The Application of Continued Fractions and their Generalizations to Problems in Approximation Theory.* P. Noordhoff N. V., Groningen, the Netherlands, 1963.

[102] A. G. Khovanskiĭ. *Fewnomials*, volume 88 of *Translations of Mathematical Monographs.* American Mathematical Society, Providence, RI, 1991. tr. from Russian by Smilka Zdravkovska.

[103] M. Kline. *Mathematical Thought from Ancient to Modern Times*, volume 3. Oxford University Press, New York and Oxford, 1972.

[104] D. E. Knuth. The analysis of algorithms. In *Actes du Congrés International des Mathématiciens*, pages 269–274, Nice, France, 1970. Gauthier-Villars.

[105] D. E. Knuth. *The Art of Computer Programming: Seminumerical Algorithms*, volume 2. Addison-Wesley, Boston, 2nd edition edition, 1981.

[106] J. Kollár. Sharp effective Nullstellensatz. *J. American Math. Soc.*, 1(4):963–975, 1988.

[107] E. Kunz. *Introduction to Commutative Algebra and Algebraic Geometry.* Birkhäuser, Boston, 1985.

[108] J. C. Lagarias. Worst-case complexity bounds for algorithms in the theory of integral quadratic forms. *J. of Algorithms*, 1:184–186, 1980.

[109] S. Landau. Factoring polynomials over algebraic number fields. *SIAM J. Computing*, 14:184–195, 1985.

[110] S. Landau and G. L. Miller. Solvability by radicals in polynomial time. *J. of Computer and System Sciences*, 30:179–208, 1985.

[111] S. Lang. *Algebra.* Addison-Wesley, Boston, 3rd edition, 1971.

[112] L. Langemyr. *Computing the GCD of two polynomials over an algebraic number field.* PhD thesis, The Royal Institute of Technology, Stockholm, Sweden, January 1989. Technical Report TRITA-NA-8804.

[113] D. Lazard. Résolution des systémes d'équations algébriques. *Theor. Computer Science*, 15:146–156, 1981.

[114] D. Lazard. A note on upper bounds for ideal theoretic problems. *J. of Symbolic Computation*, 13:231–233, 1992.

[115] A. K. Lenstra. Factoring multivariate integral polynomials. *Theor. Computer Science*, 34:207–213, 1984.

[116] A. K. Lenstra. Factoring multivariate polynomials over algebraic number fields. *SIAM J. Computing*, 16:591–598, 1987.

[117] A. K. Lenstra, H. W. Lenstra, and L. Lovász. Factoring polynomials with rational coefficients. *Math. Ann.*, 261:515–534, 1982.

[118] W. Li. Degree bounds of Gröbner bases. In C. L. Bajaj, editor, *Algebraic Geometry and its Applications*, chapter 30, pages 477–490. Springer-Verlag, Berlin, 1994.

[119] R. Loos. Generalized polynomial remainder sequences. In B. Buchberger, G. E. Collins, and R. Loos, editors, *Computer Algebra*, pages 115–138. Springer-Verlag, Berlin, 2nd edition, 1983.

[120] L. Lorentzen and H. Waadeland. *Continued Fractions with Applications.* Studies in Computational Mathematics 3. North-Holland, Amsterdam, 1992.

[121] H. Lüneburg. On the computation of the Smith Normal Form. Preprint 117, Universität Kaiserslautern, Fachbereich Mathematik, Erwin-Schrödinger-Straße, D-67653 Kaiserslautern, Germany, March 1987.

[122] F. S. Macaulay. Some formulae in elimination. *Proc. London Math. Soc.*, 35(1):3–27, 1903.

[123] F. S. Macaulay. *The Algebraic Theory of Modular Systems.* Cambridge University Press, Cambridge, 1916.

[124] F. S. Macaulay. Note on the resultant of a number of polynomials of the same degree. *Proc. London Math. Soc*, pages 14–21, 1921.

[125] K. Mahler. An application of Jensen's formula to polynomials. *Mathematika*, 7:98–100, 1960.

[126] K. Mahler. On some inequalities for polynomials in several variables. *J. London Math. Soc.*, 37:341–344, 1962.

[127] M. Marden. *The Geometry of Zeros of a Polynomial in a Complex Variable.* Math. Surveys. American Math. Soc., New York, 1949.

[128] Y. V. Matiyasevich. *Hilbert's Tenth Problem.* The MIT Press, Cambridge, Massachusetts, 1994.

[129] E. W. Mayr and A. R. Meyer. The complexity of the word problems for commutative semigroups and polynomial ideals. *Adv. Math.*, 46:305–329, 1982.

[130] F. Mertens. Zur Eliminationstheorie. *Sitzungsber. K. Akad. Wiss. Wien, Math. Naturw. Kl. 108*, pages 1178–1228, 1244–1386, 1899.

[131] M. Mignotte. *Mathematics for Computer Algebra.* Springer-Verlag, Berlin, 1992.

[132] M. Mignotte. On the product of the largest roots of a polynomial. *J. of Symbolic Computation*, 13:605–611, 1992.

[133] W. Miller. Computational complexity and numerical stability. *SIAM J. Computing*, 4(2):97–107, 1975.

[134] P. S. Milne. On the solutions of a set of polynomial equations. In B. R. Donald, D. Kapur, and J. L. Mundy, editors, *Symbolic and Numerical Computation for Artificial Intelligence*, pages 89–102. Academic Press, London, 1992.

[135] G. V. Milovanović, D. S. Mitrinović, and T. M. Rassias. *Topics in Polynomials: Extremal Problems, Inequalities, Zeros.* World Scientific, Singapore, 1994.

[136] B. Mishra. Lecture Notes on Lattices, Bases and the Reduction Problem. Technical Report 300, Courant Institute of Mathematical Sciences, Robotics Laboratory, New York University, June 1987.

[137] B. Mishra. *Algorithmic Algebra.* Springer-Verlag, New York, 1993. Texts and Monographs in Computer Science Series.

[138] B. Mishra. Computational real algebraic geometry. In J. O'Rourke and J. Goodman, editors, *CRC Handbook of Discrete and Comp. Geom.* CRC Press, Boca Raton, FL, 1997.

[139] B. Mishra and P. Pedersen. Arithmetic of real algebraic numbers is in *NC*. Technical Report 220, Courant Institute of Mathematical Sciences, Robotics Laboratory, New York University, Jan 1990.

[140] B. Mishra and C. K. Yap. Notes on Gröbner bases. *Information Sciences*, 48:219–252, 1989.

[141] R. Moenck. Fast computations of GCD's. *Proc. ACM Symp. on Theory of Computation*, 5:142–171, 1973.

[142] H. M. Möller and F. Mora. Upper and lower bounds for the degree of Gröbner bases. In *Lecture Notes in Computer Science*, volume 174, pages 172–183, 1984. (Eurosam 84).

[143] D. Mumford. *Algebraic Geometry, I. Complex Projective Varieties.* Springer-Verlag, Berlin, 1976.

[144] C. A. Neff. Specified precision polynomial root isolation is in *NC*. *J. of Computer and System Sciences*, 48(3):429–463, 1994.

[145] M. Newman. *Integral Matrices.* Pure and Applied Mathematics Series, vol. 45. Academic Press, New York, 1972.

[146] L. Nový. *Origins of modern algebra.* Academia, Prague, 1973. Czech to English Transl., Jaroslav Tauer.

[147] N. Obreschkoff. *Verteilung and Berechnung der Nullstellen reeller Polynome.* VEB Deutscher Verlag der Wissenschaften, Berlin, German Democratic Republic, 1963.

[148] C. Ó'Dúnlaing and C. Yap. Generic transformation of data structures. *IEEE Foundations of Computer Science*, 23:186–195, 1982.

[149] C. Ó'Dúnlaing and C. Yap. Counting digraphs and hypergraphs. *Bulletin of EATCS*, 24, October 1984.

[150] C. D. Olds. *Continued Fractions.* Random House, New York, NY, 1963.

[151] A. M. Ostrowski. *Solution of Equations and Systems of Equations.* Academic Press, New York, 1960.

[152] V. Y. Pan. Algebraic complexity of computing polynomial zeros. *Comput. Math. Applic.*, 14:285–304, 1987.

[153] V. Y. Pan. Solving a polynomial equation: some history and recent progress. *SIAM Review*, 39(2):187–220, 1997.

[154] P. Pedersen. Counting real zeroes. Technical Report 243, Courant Institute of Mathematical Sciences, Robotics Laboratory, New York University, 1990. PhD Thesis, Courant Institute, New York University.

[155] O. Perron. *Die Lehre von den Kettenbrüchen.* Teubner, Leipzig, 2nd edition, 1929.

[156] O. Perron. *Algebra*, volume 1. de Gruyter, Berlin, 3rd edition, 1951.

[157] O. Perron. *Die Lehre von den Kettenbrüchen.* Teubner, Stuttgart, 1954. Volumes 1 & 2.

[158] J. R. Pinkert. An exact method for finding the roots of a complex polynomial. *ACM Trans. on Math. Software*, 2:351–363, 1976.

[159] D. A. Plaisted. New *NP*-hard and *NP*-complete polynomial and integer divisibility problems. *Theor. Computer Science*, 31:125–138, 1984.

[160] D. A. Plaisted. Complete divisibility problems for slowly utilized oracles. *Theor. Computer Science*, 35:245–260, 1985.

[161] E. L. Post. Recursive unsolvability of a problem of Thue. *J. of Symbolic Logic*, 12:1–11, 1947.

[162] A. Pringsheim. Irrationalzahlen und Konvergenz unendlicher Prozesse. In *Enzyklopädie der Mathematischen Wissenschaften, Vol. I*, pages 47–146, 1899.

[163] M. O. Rabin. Probabilistic algorithms for finite fields. *SIAM J. Computing*, 9(2):273–280, 1980.

[164] A. R. Rajwade. *Squares.* London Math. Society, Lecture Note Series 171. Cambridge University Press, Cambridge, 1993.

[165] C. Reid. *Hilbert.* Springer-Verlag, Berlin, 1970.

[166] J. Renegar. On the worst-case arithmetic complexity of approximating zeros of polynomials. *Journal of Complexity*, 3:90–113, 1987.

[167] J. Renegar. On the Computational Complexity and Geometry of the First-Order Theory of the Reals, Part I: Introduction. Preliminaries. The Geometry of Semi-Algebraic Sets. The Decision Problem for the Existential Theory of the Reals. *J. of Symbolic Computation*, 13(3):255–300, March 1992.

[168] L. Robbiano. Term orderings on the polynomial ring. In *Lecture Notes in Computer Science*, volume 204, pages 513–517. Springer-Verlag, 1985. Proceed. EUROCAL '85.

[169] L. Robbiano. On the theory of graded structures. *J. of Symbolic Computation*, 2:139–170, 1986.

[170] L. Robbiano, editor. *Computational Aspects of Commutative Algebra.* Academic Press, London, 1989.

[171] J. B. Rosser and L. Schoenfeld. Approximate formulas for some functions of prime numbers. *Illinois J. Math.*, 6:64–94, 1962.

[172] S. Rump. On the sign of a real algebraic number. *Proceedings of 1976 ACM Symp. on Symbolic and Algebraic Computation (SYMSAC 76)*, pages 238–241, 1976. Yorktown Heights, New York.

[173] S. M. Rump. Polynomial minimum root separation. *Math. Comp.*, 33:327–336, 1979.

[174] P. Samuel. About Euclidean rings. *J. Algebra*, 19:282–301, 1971.

[175] T. Sasaki and H. Murao. Efficient Gaussian elimination method for symbolic determinants and linear systems. *ACM Trans. on Math. Software*, 8:277–289, 1982.

[176] W. Scharlau. *Quadratic and Hermitian Forms.* Grundlehren der mathematischen Wissenschaften. Springer-Verlag, Berlin, 1985.

[177] W. Scharlau and H. Opolka. *From Fermat to Minkowski: Lectures on the Theory of Numbers and its Historical Development.* Undergraduate Texts in Mathematics. Springer-Verlag, New York, 1985.

[178] A. Schinzel. *Selected Topics on Polynomials.* The University of Michigan Press, Ann Arbor, 1982.

[179] W. M. Schmidt. *Diophantine Approximations and Diophantine Equations.* Lecture Notes in Mathematics, No. 1467. Springer-Verlag, Berlin, 1991.

[180] C. P. Schnorr. A more efficient algorithm for lattice basis reduction. *J. of Algorithms*, 9:47–62, 1988.

[181] A. Schönhage. Schnelle Berechnung von Kettenbruchentwicklungen. *Acta Informatica*, 1:139–144, 1971.

[182] A. Schönhage. Storage modification machines. *SIAM J. Computing*, 9:490–508, 1980.

[183] A. Schönhage. Factorization of univariate integer polynomials by Diophantine approximation and an improved basis reduction algorithm. In *Lecture Notes in Computer Science*, volume 172, pages 436–447. Springer-Verlag, 1984. Proc. 11th ICALP.

[184] A. Schönhage. The fundamental theorem of algebra in terms of computational complexity, 1985. Manuscript, Department of Mathematics, University of Tübingen.

[185] A. Schönhage and V. Strassen. Schnelle Multiplikation großer Zahlen. *Computing*, 7:281–292, 1971.

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

[187] J. T. Schwartz. Polynomial minimum root separation (Note to a paper of S. M. Rump). Technical Report 39, Courant Institute of Mathematical Sciences, Robotics Laboratory, New York University, February 1985.

[188] J. T. Schwartz and M. Sharir. On the piano movers' problem: II. General techniques for computing topological properties of real algebraic manifolds. *Advances in Appl. Math.*, 4:298–351, 1983.

[189] A. Seidenberg. Constructions in algebra. *Trans. Amer. Math. Soc.*, 197:273–313, 1974.

[190] B. Shiffman. Degree bounds for the division problem in polynomial ideals. *Mich. Math. J.*, 36:162–171, 1988.

[191] C. L. Siegel. *Lectures on the Geometry of Numbers.* Springer-Verlag, Berlin, 1988. Notes by B. Friedman, rewritten by K. Chandrasekharan, with assistance of R. Suter.

[192] S. Smale. The fundamental theorem of algebra and complexity theory. *Bulletin (N.S.) of the AMS*, 4(1):1–36, 1981.

[193] S. Smale. On the efficiency of algorithms of analysis. *Bulletin (N.S.) of the AMS*, 13(2):87–121, 1985.

[194] D. E. Smith. *A Source Book in Mathematics.* Dover Publications, New York, 1959. (Volumes 1 and 2. Originally in one volume, published 1929).

[195] V. Strassen. Gaussian elimination is not optimal. *Numerische Mathematik*, 14:354–356, 1969.

[196] V. Strassen. The computational complexity of continued fractions. *SIAM J. Computing*, 12:1–27, 1983.

[197] D. J. Struik, editor. *A Source Book in Mathematics, 1200-1800.* Princeton University Press, Princeton, NJ, 1986.

[198] B. Sturmfels. *Algorithms in Invariant Theory.* Springer-Verlag, Vienna, 1993.

[199] B. Sturmfels. Sparse elimination theory. In D. Eisenbud and L. Robbiano, editors, *Proc. Computational Algebraic Geometry and Commutative Algebra 1991*, pages 377–397. Cambridge Univ. Press, Cambridge, 1993.

[200] J. J. Sylvester. On a remarkable modification of Sturm's theorem. *Philosophical Magazine*, pages 446–456, 1853.

[201] J. J. Sylvester. On a theory of the syzegetic relations of two rational integral functions, comprising an application to the theory of Sturm's functions, and that of the greatest algebraical common measure. *Philosophical Trans.*, 143:407–584, 1853.

[202] J. J. Sylvester. *The Collected Mathematical Papers of James Joseph Sylvester*, volume 1. Cambridge University Press, Cambridge, 1904.

[203] K. Thull. Approximation by continued fraction of a polynomial real root. *Proc. EUROSAM '84*, pages 367–377, 1984. Lecture Notes in Computer Science, No. 174.

[204] K. Thull and C. K. Yap. A unified approach to fast GCD algorithms for polynomials and integers. Technical report, Courant Institute of Mathematical Sciences, Robotics Laboratory, New York University, 1992.

[205] J. V. Uspensky. *Theory of Equations.* McGraw-Hill, New York, 1948.

[206] B. Vallée. Gauss' algorithm revisited. *J. of Algorithms*, 12:556–572, 1991.

[207] B. Vallée and P. Flajolet. The lattice reduction algorithm of Gauss: an average case analysis. *IEEE Foundations of Computer Science*, 31:830–839, 1990.

[208] B. L. van der Waerden. *Modern Algebra*, volume 2. Frederick Ungar Publishing Co., New York, 1950. (Translated by T. J. Benac, from the second revised German edition).

[209] B. L. van der Waerden. *Algebra.* Frederick Ungar Publishing Co., New York, 1970. Volumes 1 & 2.

[210] J. van Hulzen and J. Calmet. Computer algebra systems. In B. Buchberger, G. E. Collins, and R. Loos, editors, *Computer Algebra*, pages 221–244. Springer-Verlag, Berlin, 2nd edition, 1983.

[211] F. Viète. *The Analytic Art.* The Kent State University Press, 1983. Translated by T. Richard Witmer.

[212] N. Vikas. An $O(n)$ algorithm for Abelian p-group isomorphism and an $O(n \log n)$ algorithm for Abelian group isomorphism. *J. of Computer and System Sciences*, 53:1–9, 1996.

[213] J. Vuillemin. Exact real computer arithmetic with continued fractions. *IEEE Trans. on Computers*, 39(5):605–614, 1990. Also, 1988 ACM Conf. on LISP & Functional Programming, Salt Lake City.

[214] H. S. Wall. *Analytic Theory of Continued Fractions.* Chelsea, New York, 1973.

[215] I. Wegener. *The Complexity of Boolean Functions.* B. G. Teubner, Stuttgart, and John Wiley, Chichester, 1987.

[216] W. T. Wu. *Mechanical Theorem Proving in Geometries: Basic Principles.* Springer-Verlag, Berlin, 1994. (Trans. from Chinese by X. Jin and D. Wang).

[217] C. K. Yap. A new lower bound construction for commutative Thue systems with applications. *J. of Symbolic Computation*, 12:1–28, 1991.

[218] C. K. Yap. Fast unimodular reductions: planar integer lattices. *IEEE Foundations of Computer Science*, 33:437–446, 1992.

[219] C. K. Yap. A double exponential lower bound for degree-compatible Gröbner bases. Technical Report B-88-07, Fachbereich Mathematik, Institut für Informatik, Freie Universität Berlin, October 1988.

[220] K. Yokoyama, M. Noro, and T. Takeshima. On determining the solvability of polynomials. In *Proc. ISSAC'90*, pages 127–134. ACM Press, 1990.

[221] O. Zariski and P. Samuel. *Commutative Algebra*, volume 1. Springer-Verlag, New York, 1975.

[222] O. Zariski and P. Samuel. *Commutative Algebra*, volume 2. Springer-Verlag, New York, 1975.

[223] H. G. Zimmer. *Computational Problems, Methods, and Results in Algebraic Number Theory.* Lecture Notes in Mathematics, Volume 262. Springer-Verlag, Berlin, 1972.

[224] R. Zippel. *Effective Polynomial Computation.* Kluwer Academic Publishers, Boston, 1993.

Contents