Solving Sparse Linear Equations Over Finite Fields

DOUGLAS H. WIEDEMANN, MEMBER, IEEE

Abstract—A "coordinate recurrence" method for solving sparse systems of linear equations over finite fields is described. The algorithms discussed all require $O(n_1(\omega + n_1)\log^k n_1)$ field operations, where n_1 is the maximum dimension of the coefficient matrix, ω is approximately the number of field operations required to apply the matrix to a test vector, and the value of k depends on the algorithm. A probabilistic algorithm is shown to exist for finding the determinant of a square matrix. Also, probabilistic algorithms are shown to exist for finding the minimum polynomial and rank with some arbitrarily small possibility of error.

I. Introduction

THIS ARTICLE presents a random method of solving a nondegenerate linear system in n equations and n unknowns over a finite field in $O(n\omega)$ field operations, where ω is the total number of nonzero coefficients in all the equations. The method requires $O(n)$ space in addition to that required to store the coefficient matrix. The problem of solving sparse linear systems has been discussed in application to the discrete logarithm problem [1] and the problem of factoring large integers [2]. Experts have realized that fast algorithms for solving sparse systems bring several algorithms for these two problems to roughly the same computational complexity. Thus the complexity $\exp((1 + o(1))(\log_e n \log_e \log_e n)^{1/2})$ for factoring integers close to n and computing discrete logarithms in finite fields with approximately n elements, for the most general cases of these problems, has become something of an apparent "barrier" [3].

The method for sparse systems described here was developed after the NSF-CBMS Regional Conference in Complexity Theory at Eugene, OR. At this conference methods proposed by Coppersmith, Karmarkar, and Odlyzko were mentioned. These previous methods are finite-field adaptations of two methods for solving sparse linear systems over the reals, the Lanczos, and conjugate gradient algorithms. The method described here was developed directly for finite fields.

The method of this article is based on the fact that, when a square matrix is repeatedly applied to a vector, the resulting vector sequence is linear recursive. The problem is to solve the linear system

$$Ax = b \tag{1}$$

for a column vector x with A a given sparse matrix and b any column vector. The entries will be assumed to lie in some finite field $E = \mathrm{GF}(q)$. The parameter to describe sparsity will be ω, the number of nonzero entries in A. It can be assumed that A is stored in ω memory locations and that A can be applied to a row or column vector in at most 2ω operations. In fact, the methods here will not require representation of A to be manipulated, and they will work when A is any linear operator that requires $O(\omega)$ operations to apply. Various special cases of (1) will be investigated.

The next section gives practical algorithms for the solution of the simplest case. It is expected that typical problems can be made to fit this case. Subsequent sections show that more complex versions of the same method can be used to solve very singular cases we well as to find the determinant, minimum polynomial, and rank, although we only know how to solve the last two problems mentioned up to some arbitrarily small possibility or error. Throughout this paper we used the notation log for natural logarithm, $\log_q$ for logarithm to base q, and lg for $\log_2$.

II. The Nonsingular Square Case

The simplest case is when A is square (say $n \times n$) and known to be nonsingular. In this case, (1) has a unique solution for each b. Let S be the space spanned by $\{A^i b | i = 0, 1, 2, \cdots\}$ with A^0 defined to be the identity matrix. Then A acts in a nonsingular way on S. Let A_S denote the operator A restricted to S, and let the minimum polynomial of A_S be $f(z)$, normalized so that the *trailing* coefficient is one. Because A_S is nonsingular, z is not a factor of $f(z)$. Let $d = \deg f$ and $f[i]$ denote the coefficient of z^i in $f(z)$. Provided f can be found, the solution to (1) can quickly be constructed. In fact, $f(A_S)b = 0$, so $f(A)b = 0$. Then rearranging terms shows that A applied to

$$x = -\sum_{i=1}^{d} f[i]A^{i-1}b \tag{2}$$

gives b. Note $d = \dim S \leq n$, so that (2) can be computed in $2n(\omega + 1)$ field operations. Also, the storage required to do this is no more than a small constant times n in addition to the storage required for A. The first algorithm we give for finding f also has the property that the storage required is small; however, this algorithm is a probabilistic one.

Manuscript received November 30, 1984; revised March 18, 1985.

The author is with the Math CO Department, University of Waterloo, Waterloo, ON, Canada N2L 3G1.

IEEE Log Number 8406092.

0018-9448/86/0100-0054$01.00

Let u be any column vector, and let (,) denote vector dot product. The sequence (u, A^ib) satisfies the linear recurrence whose associated polynomial is f. This may not be the linear recurrence of smallest span of this sequence. Let f_u be the polynomial indicating the smallest span linear recurrence of the sequence $\{(u, A^ib)\}_{i=0}^{\infty}$. Call f_u the minimum polynomial of the sequence, and normalize f_u to have trailing coefficient one. It follows from elementary considerations that $f_u|f$. It is well-known that (1) f_u can be computed from the first $2n$ terms of the sequence (u, A^ib) using the Berlekamp–Massey algorithm [4]–[7] in time $O(n^2)$ field operations and (2) the multiplicative constant is small. It follows from the work in [8] that asymptotically much faster ways exist of finding the minimum polynomial, but using these probably gives no significant improvement in the total time taken by the algorithms described later. A method that solves (1) is to select u repeatedly at random, compute f_u, assume that $f_u = f$, and compute x by (2). With probability one this will eventually achieve a solution to (1); proposition 3 in Section VI shows the expected number of passes is $O(\log n)$.

A better algorithm is to use the factors of f as they are found. Let $b_0 = b$ and $f_1 = f_u$. Then if $b_1 = f_1(A)b_0 \neq 0$, the procedure can be restarted with b_1 in place of b_0. That is, a vector u_2 is selected, and the miniumum polynomial f_2 of the sequence (u_2, A^ib_1) is computed. If $b_2 = f_2(A)b_1 \neq 0$, the procedure continues. Eventually, some $b_k = 0 = f_k \cdots f_1(A)b_0$, in which case $f = f_k \cdots f_1$ can be computed and the solution found by (2). A more efficient version of this is to compute the solution as a byproduct. For any polynomial g, define $g^-(z) = (g(z) - g(0))/z$.

Algorithm 1

1) Set $b_0 = b$, $k = 0$, $y_0 = 0$, and $d_0 = 0$.
2) If $b_k = 0$, then the solution is $x = -y_k$ and stop.
3) Select u_{k+1} at random.
4) Compute the first $2(n - d_k)$ terms of $\{(u_{k+1}, A^ib_k)\}_{i=0}^{\infty}$.
5) Set $f_{k+1}(z)$ to the minimum polynomial of the sequence of step 4.
6) Set $y_{k+1} = y_k + f_{k+1}^-(A)b_k$, $b_{k+1} = b_0 + Ay_{k+1}$, and $d_{k+1} = d_k + \deg(f_{k+1})$.
7) Set $k = k + 1$ and go to step 2.

The only reason for introducing the y_k is that it avoids a separate computation of (2) at the end. If it is only desired to compute f, $b_{k+1} = f_{k+1}(A)b_k$ can be computed directly in step 6. To find f an initial estimate of one is set to the polynomial in step 1, and in step 6 the current value should be multiplied by f_{k+1}.

Note that the computation in step 4 can be done in $4n$ memory locations if the vectors A^ib_k are not kept as they are generated. Although this is recommended, it requires recomputation of A^ib_k in step 6. In Section VI it is proved that Algorithm 1 stops after three passes with probability at least 70 percent. Selecting u_k to be the kth unit vector, can convert the Algorithm 1 to a deterministic algorithm. The following algorithm always completes in $O(n(\omega + n \log n \log\log n))$ operations but requires $2n^2$ extra memory locations.

Algorithm 2

1) Compute A^ib for $i = 0, 1, \cdots, 2n - 1$, and save all results.
2) Set $k = 0$ and $g_0(z) = 1$.
3) Set u_{k+1} to be the $k + 1$st unit vector.
4) Extract from the result of step 1 the sequence $\{(u_{k+1}, A^ib)\}_{i=0}^{2n-1}$.
5) Apply $g_k(z)$ to this sequence.
6) Set f_{k+1} to be the minimum polynomial of the sequence produced in step 5.
7) Set $g_{k+1} = f_{k+1}g_k$.
8) Set $k = k + 1$; and then if $\deg(g_k) < n$ and $k < n$ go to step 3.
9) Using $f = g_k$ and the table computed in step 1, compute the solution x by (2).

Some comments are required to see that Algorithm 2 runs in the claimed time. In step 5 a polynomial is *applied* to a sequence. Given a polynomial $g(z)$ of degree d and a sequence $\{s_i\}_{i=0}^{l}$, let

$$s(z) = \sum_{i=0}^{l} s_i z^i, \tag{3}$$

and say that by definition the result of applying g to the sequence is

$$\{(g(z^{-1})s(z))[i]\}_{i=0}^{l-d}. \tag{4}$$

Clearly, applying a polynomial to a sequence can be done by multiplying polynomials. In step 5 fast polynomial multiplications taking $O(n \log n \log\log n)$ operations [9] are used. The sequence produced in step 5 is in fact $(u_{k+1}, A^ig_k(A)b)$, and it follows that the minimum polynomial extracted in step 6 is a factor of $f(z)/g_k(z)$.

To bound the work in step 6, the additional observation must be used that if the Berlekamp–Massey algorithm is applied to a sequence of length $2n$ to obtain a polynomial of degree d, the number of operations is $O(nd)$. The total work in step 6 is $O(n^2)$, because the sum of the degrees of the factors extracted in step 6 is $\deg f \leq n$.

III. Singular and Nonsquare Cases

If A is square and singular, b must lie in a proper subspace of E^n for a solution to (1) to exist. If b is selected at random, then with probability at least $1 - 1/q \geq 1/2$ there will be no solution. If (1) has no solution, then applying either of the algorithms will result in a proof that A is singular. In this case, one of the coordinate recurrences will have zero constant term, because this is the only way the algorithms can fail to produce a solution to (1). If $z|f(z)$ then $A(f^-(A))b = f(A)b = 0$ so the vector $f^-(A)b$ provides a linear combination of the columns resulting in zero. Moreover, $f^-(A)b \neq 0$, by minimality of f.

If a column that is dependent on the other columns is found, it can be eliminated from the system. Also, if a

dependent row is located, it can be eliminated along with the corresponding entry of b, after checking that the eliminated equation is consistent with the equations with which it is dependent. We have just seen how dependencies can be found in square matrices. By transposing if necessary and ignoring excess columns, finding dependencies in nonsquare matrices can always be reduced to the case of finding a column dependency in an $n \times (n+1)$ matrix, say M. Remove the final column from M, breaking it into an $n \times n$ matrix A and a column vector b. Then applying algorithms of the previous section produces either a solution to $Ax = b$ or a column dependency of A. In either case, a column dependency of M has been produced.

A method for solving singular or nonsquare systems is to continue eliminating dependent rows and columns until an inconsistency or a square nonsingular system is obtained. This can be done deterministically if Algorithm 2 is used, and we note that the elimination of rows or columns does not otherwise change the coefficient matrix, so this matrix need not be modified. A difficulty with this method is that the work becomes large if many rows and columns must be eliminated. Another possible difficulty is that it requires application of the transpose of the coefficient matrix to column vectors, a possibly inconvenient operation.

Let A be an $m \times n$ matrix, and let $n_0 = \min(m, n)$ and $n_1 = \max(m, n)$. A random method will now be described for finding a solution to $Ax = b$ if one exists. This method requires that rank $(A) = n_0$ and will require $O(n_1(\omega + n_1 \log n_1))$ field operations to find a solution with probability $1/2$. The idea is to extend A to an $n_1 \times n_1$ nonsingular matrix B by adjoining randomly selected rows or columns. If $m > n$, the extension will have extra variables but no extra equations. The solution to the extended system $By = b$ will set to zero the extra variables and will therefore restrict to a solution of the original equations. If $m < n$ the extra rows correspond to extra equations that may be taken to be homogeneous. The solution to the extended set must also be a solution to the original set. The following propositions are useful elementary inequalities for bounding the number of sparse vectors that are included in a subspace. The number of nonzero entries in a vector is called its Hamming weight.

Proposition 1: Let C be any linear vector space of dimension k in E^n. Let $a[j]$ denote the number of elements of C with Hamming weight j. Then for each i,

$$0 \le i \le n,$$

$$\sum_{j=0}^{i} a[j] \le \sum_{j=0}^{i} \binom{k}{j} (q-1)^j.$$

Proof: Consider a $k \times n$ matrix whose rows span C. Permute the columns, if necessary, so that the matrix can be written in the form (F, G), where F is a nonsingular $k \times k$ matrix. Column permutation does not alter the weight distribution of the rowspace. Each vector is now uniquely determined by its leading k coordinates, and any vector of weight at most i has weight at most i in its leading k coordinates. Exactly $\sum_{j=0}^{i} \binom{k}{j} (q-1)^j$ vectors of length k and weight at most i exist in their leading k coordinates, and the result follows.

Proposition 2: If C is a linear vector space of dimension k in E^n, and

$$a(r) = \sum_{j=0}^{n} a[j] r^j,$$

then for $0 \le r \le 1$, $a(r) \le (1 + (q-1)r)^k$.

Proof: Given the constraints on r, the numbers $r^i - r^{i+1}$, $i = 0, 1, \cdots, n-1$, are non-negative. Summing the inequalities of proposition 1 with these weights and the inequality at $i = n$ with weight r^n gives

$$\sum_{i=0}^{n-1} (r^i - r^{i+1}) \sum_{j=0}^{i} a[j] + r^n \sum_{j=0}^{n} a[j] \le \sum_{i=0}^{n-1} (r^i - r^{i+1}) \sum_{j=0}^{i} \binom{k}{j} (q-1)^j + r^n \sum_{j=0}^{n} \binom{k}{j} (q-1)^j.$$

Collecting coefficients of r^j on both sides gives

$$a(r) \le \sum_{j=0}^{n} r^j \binom{k}{j} (q-1)^j = (1 + (q-1)r)^k.$$

Assume that a matrix A is $m \times n$ with $m < n$ and rank $A = m$. The strategy for completing A to a square nonsingular matrix is to generate a row i for $i = m+1, m+2, \cdots, n$ as follows. For each i a parameter $w_i \le 1 - q^{-1}$ will be chosen in advance. Select each entry of row i to be 0 with probability $1 - w_i$; otherwise, set that entry to a uniform randomly selected nonzero element of E. All random selections are to be made independently.

Any vector of Hamming weight j has probability $(q-1)^{-j} w_i^j (1 - w_i)^{n-j}$ of being selected. Let a be the weight enumerator polynomial of the subspace spanned by the first $i-1$ rows. By proposition 2 the probability that row i lies in this subspace is

$$\begin{aligned}(1 - w_i)^n a(w_i/((1 - w_i)(q-1))) &\le (1 - w_i)^n (1 + w_i/(1 - w_i))^{i-1} \\ &= (1 - w_i)^{n-(i-1)}.\end{aligned}$$

If the first $i-1$ rows are linearly independent and row i happens not to be in the span of the previous $i-1$ rows, then the first i rows are linearly independent. Let $w_i = \min(1 - q^{-1}, 2(\log n)/(n+1-i))$. If $w_i = 2(\log n)/(n+1-i)$, then $(1 - w_i)^{n-(i-1)} \le (1 - 2(\log n)/(n+1-i))^{n+1-i} \le \exp(-2\log n) = n^{-2}$. On the other hand, if $w_i = 1 - q^{-1}$, $(1 - w_i)^{n-(i-1)} = q^{-(n+1-i)}$. Let $j = n+1-i$. The conditional probability that the first i rows are independent, given that the first $i-1$ rows are independent, is at least $1 - (n^{-2} + q^{-j})$. By multiplying all these together the probability that the whole $n \times n$ matrix is

nonsingular is at least

$$\prod_{j=1}^{n}\left(1-\left(n^{-2}+q^{-j}\right)\right)=\left(1-n^{-2}\right)^{n}\cdot\prod_{j=1}^{n}\left(1-\frac{n^2}{n^2-1}q^{-j}\right).$$

This bound does not depend on the value of m, and through some calculation it can be shown to be greater than $1/7$ for $n \geq 3$ and $q \geq 2$. Furthermore, the expected number of nonzero entries in the rows beyond the first m is

$$n\sum_{i=m+1}^{n} w_i \leq 2n\log n\sum_{i=m+1}^{n}\frac{1}{n+1-i}.$$

This does not exceed $2n(\log n)(1+\log(n-m)) \leq 2n(\log n)(1+\log n)$. Furthermore, the variance of the weight of any component is at most $1/4$, so the variance of the weight of the entire matrix is at most $n^2/4$. By Chebyshev's inequality the probability that the mean is exceeded by more than $8n$ is less than $1/256$.

Theorem 1: For any integers $n > m \geq 0$ a random procedure exists for generating $n-m$ row vectors of length n such that if A is an $m \times n$ matrix of rank m, then with probability at least $1/8$, both of the following statements are true. 1) The $n \times n$ matrix formed by A and theses rows is nonsingular. 2) The total Hamming weight of the additional rows is at most $2n(2+\log n)^2$.

Proof: For $n \geq 3$ the aforementioned procedure produces a nonsingular matrix with probability at least $1/7$. The procedure produces additional Hamming weight of more than $2n(2+\log n)^2 \geq 2n(\log n)(1+\log n)+8n$ with probability less than $1/256$. Therefore, the probability of satisfying statements 1 and 2 is at least $1/7-1/256 > 1/8$. To complete the proof, only the cases $n=1,2$ remain; but then statement 2 will always hold, and if the additional entries are selected uniformly at random, statement 1 is satisfied with probability at least $1/4$.

Theorem 1 is not asymptotically the best possible. The following improvement was made mostly through the efforts of one of the referees, and the author is grateful for being permitted to include it.

Theorem 1': Numbers $\epsilon > 0$ and c_1 exist, both independent of q, with the following property. For any integers $n > m \geq 0$ a random procedure exists for generating $n-m$ row vectors of length n such that if A is an $m \times n$ matrix of rank m, then with probability at least ϵ, the resulting $n \times n$ matrix is nonsingular and the total Hamming weight of the generated rows is at most $1+c_1 n\log n$.

Proof: Assume first that $q \leq n^2$, since larger values of q will be handled by a different method. We may suppose $n = m+k+c_2$, where $k \geq c_3 \log n$ and c_2, c_3 are constants to be selected later. If $n-m$ would be smaller than this, simply generate all $n-m$ rows uniformly at random, and the resulting matrix will be nonsingular with probability at least $1/4 < (1-q^{-1})(1-q^{-2})(1-q^{-3})\cdots$, and the weight bound is obviously satisfied. As it is, fill in the final c_2 rows uniformly at random. Let $z = 1-c_3(\log n)/k$. For each entry of the first k rows, the method sets that entry to zero with probability z and to a randomly selected element of GF(q) otherwise. The weight bound with an appropriate c_1 is satisfied with high probability. It must be shown that when A is adjoined to the first k rows the result has rank $m+k$ with probability bounded away from zero.

The probability that a particular sum of the first k generated rows with j nonzero coefficients yields a particular vector of weight i is

$$\left(z^j+\frac{1}{q}(1-z^j)\right)^{n-i}\left(\frac{1}{q}(1-z^j)\right)^{i}. \tag{5}$$

Let a be the weight enumerator polynomial of the row space of A. Let ρ be the probability that the $m+k$ rows are linearly dependent. By (5) and the union bound,

$$\rho \leq \sum_{i=0}^{n} a[i]\sum_{j=1}^{k}\binom{k}{j}(q-1)^j\cdot\left(z^j+\frac{1}{q}(1-z^j)\right)^{n-i}\left(\frac{1}{q}(1-z^j)\right)^{i}$$

$$\rho \leq \sum_{j=1}^{k}\binom{k}{j}(q-1)^j\left(z^j+\frac{1}{q}(1-z^j)\right)^{n}\cdot\sum_{i=0}^{n} a[i]\left(\frac{q^{-1}(1-z^j)}{\left(z^j+q^{-1}(1-z^j)\right)}\right)^{i}.$$

By proposition 2

$$\rho \leq \sum_{j=1}^{k}\binom{k}{j}(q-1)^j\left(z^j+\frac{1}{q}(1-z^j)\right)^{n-m}.$$

Now $z^j \leq \exp(-c_3 j(\log n)/k)$. Let $\beta = j/k$ and $c_3 = 3c_4$, and since $q \leq n^2$, $z^j \leq (nq)^{-c_4\beta}$. Then

$$\rho \leq \sum_{j=1}^{k}\binom{k}{j}(q-1)^j\left(\frac{1}{q}+\frac{q-1}{q}(nq)^{-c_4\beta}\right)^{n-m}. \tag{6}$$

Split the summation in (6) into two pieces, ρ_0 for $j < k\beta_0$ and ρ_1 for $j \geq k\beta_0$, where $\beta_0 > 0$ is a constant to be determined later. It will be shown that each piece can be made less than $1/3$. If $\beta \geq \beta_0$, then for c_4 sufficiently large,

$$\left(1+(q-1)(nq)^{-c_4\beta}\right)^{n-m} \leq \exp\left((q-1)(n-m)(nq)^{-c_4\beta}\right) \leq 2$$

and therefore

$$\rho_1 \leq 2q^{-(n-m)}\sum_{j>k\beta_0}\binom{k}{j}(q-1)^j \leq 2q^{-c_2},$$

and this is less than $1/3$ for $c_2 \geq 3$.

In order to bound ρ_0, invoke the inequality $\binom{k}{j} \leq (1/(\beta^\beta(1-\beta)^{1-\beta}))^k$ [10, lemma 10.7] and with (6) obtain

$$\rho_0 \leq \sum_{1 \leq j < k\beta_0} \left(\frac{(q-1)^\beta}{\beta^\beta(1-\beta)^{1-\beta}} \left(\frac{1}{q} + \frac{q-1}{q}(nq)^{-c_4\beta} \right) \right)^k. \tag{7}$$

Consider the function $f(q) = (q-1)^\beta(q^{-1} + (1 - q^{-1})(nq)^{-c_4\beta})$. It will be shown that if c_4 is sufficiently large, $0 < \beta < 1/4$, $n \geq 1$, and $q \geq 2$, then $f(q) \leq f(2)$. Differentiating with respect to q, we have that

$$\begin{aligned} q^2(q-1)^{-\beta} f'(q) &= \frac{\beta q}{q-1} + \beta(nq)^{-c_4\beta} q - 1 \\ &\quad - c_4\beta q(nq)^{-c_4\beta} + (c_4\beta + 1)(nq)^{-c_4\beta}. \end{aligned} \tag{8}$$

To show that (8) is negative, we first have

$$(\beta q - c_4\beta q + c_4\beta)(nq)^{-c_4\beta} < 0$$

because $c_4 > 2$ implies $c_4(q-1) > q$. When this is subtracted from (7), the remainder

$$\beta q(q-1)^{-1} - 1 + (nq)^{-c_4\beta} \tag{9}$$

is also negative. Clearly (9) is decreasing in n and q; so replace n by 1 and q by 2, giving $2\beta - 1 + 2^{-c_4\beta}$. Let $2^{-c_4} = e^{-4}$. A Taylor expansion and $0 < 4\beta < 1$ give

$$2\beta - 1 + e^{-4\beta} < -2\beta + 8\beta^2 < -2\beta + 2\beta = 0.$$

In fact, this means (9) and (8) are negative for all values of $c_4 \geq 4\log 2$, $0 < \beta < 1/4$, $n \geq 1$, and $q \geq 2$; therefore, the claim is proved.

In (7), $\beta \geq 1/k$, so $n^{-1} \leq \beta$. Hence

$$f(2) \leq \tfrac{1}{2} + \tfrac{1}{2}\beta^{c_4\beta}.$$

Using $\beta^\beta \approx 1 + \beta\log\beta$,

$$\beta^{-\beta}(1-\beta)^{-(1-\beta)} f(2) \leq 1 + \left(\tfrac{1}{2}c_4 - 1\right)\beta\log\beta + O(\beta)$$

as $\beta \to 0^+$. It follows that for some β_0, $0 < \beta_0 < 1$, for sufficiently large c_4, $n^{-1} \leq \beta < \beta_0$ implies $\beta^{-\beta}(1-\beta)^{-(1-\beta)} f(2) \leq \beta^\beta$. Then for sufficiently small β_0 and sufficiently large c_4,

$$\rho_0 \leq \sum_{1 \leq j < k\beta_0} \beta^{k\beta} \leq \sum_{j=1}^{\infty} \beta_0^j = \beta_0/(1-\beta_0) < 1/3$$

as required.

Having completed the case $q \leq n^2$, we use a different method for $q > n^2$. Begin by forming $n - m$ rows as the method would produce for $q = 2$, and call the resulting $0-1$ matrix M'. The matrix A must contain an $m \times m$ nonsingular submatrix. Such a submatrix must have at least one transversal with only nonzero entries. Let A' be a matrix over GF(2) with ones at positions corresponding to the transversal and zeros elsewhere. Since A' with M' adjoined has nonzero determinant with probability at least $\delta > 0$, A with M' adjoined possesses a transversal of nonzero entries with probability at least δ. Assume that A with M' adjoined has such a transversal.

Let M be a matrix formed by replacing each one-entry of M' with a different GF(q) indeterminate. Let B be the $n \times n$ matrix consisting of A with M adjoined. Under the assumption of the preceding paragraph, $\det(B)$ does not identically vanish. Because $\det(B)$ is a polynomial of degree at most n, a result of Schwartz [11] shows that if each variable entry of B is replaced by an independently selected random element of GF(q), $\det(B) \neq 0$ with probability at least $1 - n/q > 1 - n^{-1}$. The overall procedure of selecting M' and an instance of M produces a nonsingular matrix with probability at least $\delta(1 - n^{-1})$.

IV. Highly Singular Cases

If A is $m \times n$, $m < n$, and rank $A = m$, the following method is better than completing A to an $n \times n$ matrix when m is much smaller than $n/2$. Suppose a sparse $n \times m$ matrix Q can be found such that $\det(AQ) \neq 0$. Then the unique solution y to $AQy = b$ can be found, and $x = Qy$ will solve the equation $Ax = b$. A similar procedure will find a solution to an overdetermined system, say $A^Tx = b$, where A^T denotes the transpose of A. Now Q^T is a sparse $m \times n$ matrix such that $\det(Q^TA^T) \neq 0$. Applying Q^T to both sides of the equation gives $Q^TA^Tx = Q^Tb$, which can now be solved for x. The unique solution of this derived equation then must solve the original system if it is consistent.

Fortunately, Q can be generated by the method described in the previous section. In fact, let V be the subspace of E^n spanned by the rows of A. Let $V^\perp$ be the $(n-m)$-dimensional space of vectors having zero dot product with all members of V. If Q is an $n \times m$ matrix, then $\det(Q^TA^T) \neq 0$ if any only if the row space of Q^T together with $V^\perp$ generates all of E^n. This is because a nonzero vector v of length m such that $vQ^TA^T = 0$ exists if and only if there is a nonzero combination of the rows of Q^T that has zero dot product with each row of A and therefore lies in $V^\perp$. It follows that Q can be constructed exactly as if completing an $(n-m) \times n$ matrix to a nonsingular square matrix. In this case, the $(n-m) \times n$ matrix would represent a basis of $V^\perp$.

In solving the systems with coefficient matrix Q^TA^T or AQ, the two matrices never need be multiplied, because we may first apply one and then the other.

Now let A be the $m \times n$ coefficient matrix, with m and n unrestricted. Let $n_0 = \min(m, n)$ and $n_1 = \max(m, n)$, as in the previous section. We have not yet given a method that will find a solution for the case rank $A < n_0$. Suppose that the rank is known to be r. A method is to select $r \times m$ and $n \times r$ matrices P and Q, respectively, such that $\det(PAQ) \neq 0$. The selection of P and Q^T can be made according to Theorem 1′ and the probability that PAQ is nonsingular is bounded away from zero. Once the two matrices have selected, the solution to $PAQy = Pb$ can be found, and $x = Qy$ can be tested in (1). If PAQ is nonsingular, then a solution to (1) will be found, provided one exists.

If the rank r is unknown, then it can be arrived at using binary search. As a guess, s, $s \times m$, and $n \times s$ matrices P and Q are selected, and PAQ is tested probabilistically for singularity. If $s > r$ the result will always be singular. After being repeated several times the question, "Is $s > r$?" can be answered with probability of error at most $1/3$. This allows conducting a binary search with a constant level of uncertainty for each reply. The value of r can therefore be guessed with probability of error at most $1/3$ by conducting $O(\log n_0)$ tests of values s. Of course, more certainly can be obtained by a small increase in the number of tests. All this leads to an $O(n_0(\omega + n_1 \log n_1)\log n_0)$ expected time method of producing a solution to any linear system, providing a solution exists. We do not consider it obvious that binary search of a space of size n_0 can still be done in $O(\log n_0)$ queries even if each answer has a bounded probability of being incorrect. However, problems close to this are discussed by Rivest *et al.* [12], and the method of that paper extends to solve our problem.

In highly singular systems, it may be desired to produce more than one solution. In the remainder of this section methods will be given for extracting uniform randomly selected solutions to (1). One method is to find r and an $r \times m$ matrix P such that $\operatorname{rank}(PA) = r$. Then (1) is equivalent to solving $(PA)x = Pb$. Extend PA to an $n \times n$ matrix G by adjoining sparse rows as in Theorem 1′. If G can be shown singular, then another G must be constructed. Continue until a G that is thought to be nonsingular is found. Assuming G is nonsingular, solve $Gx = g$, where g is a column vector whose first r entries are Pb and whose remaining entries are filled in at random. The unique solution to $Gx = g$ must also solve $(PA)x = Pb$. Since a one-to-one correspondence exists between solutions to $(PA)x = Pb$ and the vectors g, a uniform random solution is selected by selecting g uniformly at random.

A second method may be better. Assume an $r \times m$ matrix P and an $n \times r$ matrix Q have been found such that PAQ is nonsingular. Extend Q to a square matrix M as shown:

$$M = \left[Q \middle| \frac{0}{I} \right]$$

where I denotes the identity matrix of size $n - r$. Suppose M is nonsingular. Random solutions to $PAMy = Pb$ can easily be found because the first r columns of PAM form PAQ, a nonsingular matrix. Thus the last $n - r$ coordinates of y can be set to any value, and the initial r coordinates can be subsequently filled in by solving an $r \times r$ system to yield a solution to $PAMy = Pb$. This again gives a method of uniform random selection of solutions $x = My$ to (1). There is a problem in using Theorem 1′ to select Q: both PAQ and the first $r \times r$ minor of Q must be nonsingular. The theorem bounds the probability of each event away from zero but not necessarily their intersection. However, the proof of Theorem 1′ can be modified to show that the same type of procedure will produce a set of row vectors that completes both of two matrices to nonsingular matrices with probability bounded away from zero for n sufficiently large. This method may be advantageous for $r \ll n$.

In this section some possibility of error was due to not knowning rank A and some was due to not being certain that a matrix is nonsingular. The next section gives a method of proving a sparse square matrix is nonsingular.

V. Finding the Determinant

Given a sparse $n \times n$ matrix A, a random algorthim has been given for establishing that the determinant is zero. A more difficult problem is to find the determinant of A when it is not zero. Let f^A be the minimum polynomial of A, and let f^b be the minimum polynomial of A_S where S is the space generated from the column vector b as in Section II. Each of the two algorithms in that section provides a method of computation of f^b for any given b. Since f^A is the lowest degree of polynomial such that $f^A(A)b = 0$ for all b, f^A is the least common multiple (lcm) of all the f^b. In fact, it will be shown in the next section that selecting $O(1)$ values of b uniformly at random and computing the lcm of the resulting f^b will nearly always give f^A. This method finds the minimum polynomial of A with arbitrarily high probability. Let c^A denote the characteristic polynomial of A. If $c^A = f^A$, A is said to be nonderogatory. This happens if and only if f^A has degree n. Whenever the lcm of the f^b has degree n, the minimum and characteristic polynomial have been determined with certainty.

If c^A has been found, then normalizing it to have leading coefficient one, and evaluating at zero gives $(-1)^n \det(A)$. Whenever A is nonderogatory, $\det(A)$ can be evaluated by finding f^A. This will take an expected number of field operations that is $O(n(\omega + n))$. If it is suspected that A is derogatory, one can randomly permute the rows of A and try to compute the determinant of the result. The two determinants are related by a known sign factor. However, some matrices cannot be permuted to form a nonderogatory matrix. An example of this is any sufficiently large identity matrix. To overcome this difficulty a slightly cumbersome method that may involve field extensions is now described.

If c^A is square-free, then $f^A = c^A$ because every irreducible factor of c^A must appear in f^A. Note that c^A is square-free if and only if its discriminant $\operatorname{disc}(c^A)$ is a nonzero element of E.

Lemma: If A is an $n \times n$ matrix over E and all leading principal minors of A, including A itself, are nonsingular, then the discriminant of the characteristic polynomial of $A \cdot \operatorname{diag}(y_1, \cdots, y_n)$ is not the zero polynomial in $y_1, \cdots, y_n$.

Proof (mathematical induction): The statement is obvious for $n = 1$ where the discriminant is identically one. Assume that it is true for $(n-1) \times (n-1)$ matrices, $n > 1$. Then substitute zero for y_n in $\det(zI - A \operatorname{diag}(y_1, \cdots, y_n))$ to obtain $z \det(zI' - A' \operatorname{diag}(y_1, \cdots, y_{n-1}))$, where I' is an $(n-1) \times (n-1)$ identity matrix and A' is a leading principal minor. The discriminant of

this is

$$\det^2(A' \operatorname{diag}(y_1, \cdots, y_{n-1})) \cdot \operatorname{disc}(\det(zI' - A' \operatorname{diag}(y_1, \cdots, y_{n-1}))).$$

The first factor equals $\det^2(A') y_1^2 \cdots y_{n-1}^2$ and is not identically zero by hypothesis. By the induction hypothesis the second factor is not identically zero.

The result of Schwartz [11] used in Section III states that for any polynomial of total degree d over a field of order l, if the variables are substituted with uniform random field elements, the result is zero with probability no more than d/l, provided the polynomial is not the zero polynomial. Applying this and the lemma to $y_1 y_2 \cdots y_n$ times the discriminant of $A \operatorname{diag}(y_1, \cdots, y_n)$ if A and all of its leading principal minors are nonsingular, then the y_i can be selected uniformly at random from any sufficiently large extension field of E, and with high probability $A \operatorname{diag}(y_1, \cdots, y_n)$ will be nonderogatory and nonsingular. Then $\det(A \operatorname{diag}(y_1, \cdots, y_n))$ can be evaluated, and by dividing this by $y_1 \cdots y_n$, $\det A$ is found.

If A is nonsingular there is some permutation matrix P such that AP has only nonsingular leading principal minors. This is easily proven by induction or by ordering the columns as they would be used in a Gaussian reduction of A. To incorporate the unknown permutation P into a probabilistic algorithm, we find a parametric family of nonsingular matrices that gives P for some setting of the parameters. Consider the matrix product

$$\begin{bmatrix} 1 & a \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 \\ b & 1 \end{bmatrix} \begin{bmatrix} 1 & c \\ 0 & 1 \end{bmatrix}. \tag{10}$$

This product is always nonsingular; in fact, its determinant is always one. For $(a, b, c) = (0, 0, 0)$ the result is the identity permutation matrix, and for $(a, b, c) = (1, -1, 1)$ the result is the transposition permutation matrix except that one entry is negated. Negation of some entries as they are permuted is unimportant because the result will be ultimately multiplied by an arbitrary diagonal matrix.

For $n > 2$ any, possibly incomplete, pairing on n entries of a vector can be taken and an independent member of (10) applied to each pair. For n a power of two, a highly structured sequence of pairings exists [13] such that each permutation on n elements is achieved by interchanging or leaving unchanged each of the pairs. There are such "permuting networks" for n not a power of two, but there is the option of extending the matrix A to have size a power of two by adding diagonal ones and zeros elsewhere. If A is extended in this way, the determinant will not change.

Let $n = 2^k$, $k > 1$, in the following.The permuting network will have $2k - 1$ levels or pairings. The result of all the associated applications of (10) is a matrix P in a large number of variables with each entry of total degree no more than $6k - 3$. Multiply this by $\operatorname{diag}(y_1, \cdots, y_n)$, where $y_1, \cdots, y_n$ are n new parameters, and call the resulting matrix Y. By the lemma, if A is nonsingular, $\operatorname{disc}_z(\det(zI - AY))$ is not identically zero, because for some substitution of the parameters of P, AP is in the form required by the lemma. Furthermore, the degree of the discriminant is at most $6kn(n-1)$. Now select all the parameters uniformly randomly from a field of order l containing E. Then $y_1 \cdots y_n \operatorname{disc}(\,)$ evaluates to a nonzero element with probability at least $1 - 6kn^2/l$. In such cases $\det(AY)$ can be found by finding its characteristic polynomial, and $\det(A) = y_1^{-1} \cdots y_n^{-1} \det(AY)$.

Theorem 2: Let A be an $n \times n$ matrix over a finite field. There is a random algorithm that with probability at least $1/2$ will evaluate $\det(A)$. The algorithm requires $O(n(\omega + n \log n))$ field operations in an extension field having at least $50n^2 \lg n$ elements, where it requires ω operations to apply A to a vector. The additional storage required is $O(n \log n)$ elements of the extension field.

Proof: In the foregoing discussion, which applied to a nonsingular matrix with size a power of two, applying Y required $O(n \log n)$ field operations, so Algorithm 1 may be applied to find the minimum polynomial in the required number of operations. If A is singular this can be established separately by the method in Section III, but it will also become known when a minimum polynomial of AY that has zero constant term is found. If n is not a power of two, A can be extended to be $\bar{n} \times \bar{n}$, where $\bar{n}$ is a power of two. Since $\bar{n}$ may be nearly $2n$, requiring that the extension field have size at least $l = 50n^2 \lg n$ forces $6\bar{n}^2 \lg \bar{n}/l < 2/3$ for any reasonably large value of n. Then the probability of success of the entire procedure is bounded away from zero. Repeating this a fixed number of times will provide $\det(A)$ with probability at least $1/2$.

VI. Probabilistic Analysis

In this section we prove statements previously made concerning the probability that the minimum polynomial is obtained. Let A be any $n \times n$ matrix, possibly singular. As in Section II let b be any column vector in E^n, let $S = \operatorname{span}\{b, Ab, A^2b, \cdots\}$ and let A_S be the operator A restricted to S. Define $f^b(z)$ to be the minimum polynomial of A_S. An important fact is that the elements of S can be identified with the elements of the ring $R = E[z]/(f^b(z))$. This is established by means of a mapping from $E[z]$ to S defined such that one is mapped to b and multiplication by z corresponds to application of A in S. The kernel of this mapping is the principal ideal $(f^b(z))$, and reduction by this kernel produces the bijection ψ: $R \to S$.

The set of linear functionals from R to E, R^*, may be identified with R by a bijection η: $R \to R^*$ defined as follows. Let $d = \deg f^b \geq 1$ and define $\eta(1)$ so that $\eta(1)(z^{d-1}) = 1$ and $\eta(1)(z^i) = 0$ for each value of i such that $0 \leq i < d - 1$. The definition of η can then be uniquely extended to all of R such that $\eta(zg)(h) = \eta(g)(zh)$ for all $g, h \in R$. It is easily established that η is injective and therefore a bijection, since R and R^* have the same dimension over E. Associated with ψ is the dual map ψ^*: $S^* \to R^*$ defined by the equation $\psi^*(l)(g) = (l)(\psi(g))$ for any $l \in S^*$ and $g \in R$. Note that ψ^* is also a bijection.

Now if u is any vector in E^n, dot product with u defines a linear functional $\zeta(u)$ on S. The mapping ζ: $E^n \to S^*$ is clearly surjective. It follows that if u is selected uniformly at random, then $\zeta(u)$ is uniform random in S^*. If g is the unique member of R such that $\psi^*(\zeta(u)) = \eta(g)$, the following sequences in E are equal:

$$\{(u, A^i b)\}_{i=0}^{\infty} = \{\eta(g)(z^i)\}_{i=0}^{\infty} = \{\eta(1)(z^i g)\}_{i=0}^{\infty}.$$

The maximum polynomial of any of these sequences is called f_u^b. By virtue of the last sequence, $f_u^b = f^b/\gcd(f^b, g)$. This is true because $h = f^b/\gcd(f^b, g)$ is the lowest degree polynomial such that in R, $hz^i g = 0$ for all nonnegative values of i. In fact, it is the lowest degree polynomial such that $hg = 0$. Therefore, the probability that $f_u^b = f^b$ is the probability that a randomly selected element of R is a unit. In analogy with the computation of the Euler phi-function of an integer, this fraction can be calculated by the following formula:

$$\Phi(f^b) = \prod_i (1 - q^{-d_i}) \tag{11}$$

where d_i is the degree of the ith irreducible factor of f^b, and in (11) each irreducible factor of f^b appears exactly once.

Proposition 3: If $\deg f \le q + \cdots + q^j$ for j a positive integer, then $\Phi(f) \ge 1/(6j)$.

Proof: Since the factorization of $z^{q^i} - z$ includes each irreducible polynomial of degree i, the number of irreducibles of degree i is at most q^i/i. Let $x_i(q^i/i)$ be the number of irreducibles of degree i that divide f. Then

$$0 \le x_i \le 1 \tag{*}$$

$$\sum_{i=1}^{\infty} x_i q^i \le \deg f \le q + \cdots + q^j. \tag{**}$$

Given the constraints in (*) and (**), the maximum possible value of $\sum_{i=1}^{\infty} x_i/i$ is $H_j = 1/1 + 1/2 + \cdots + 1/j$. This is because the constraint coefficients q^i are increasing in i and the objective functional coefficients $1/i$ are decreasing in i, so the maximum value is obtained by setting the first j variables x_i to one and all others to zero. Now since $q \ge 2$

$$\begin{aligned} \log \Phi(f) &= \sum_i \log(1 - q^{-i}) x_i q^i/i \\ &\ge -\sum (q^{-i} + q^{-2i}) x_i q^i/i \\ &\ge -\sum x_i/i - \log 2. \end{aligned}$$

Thus $\log \Phi(f) \ge -H_j - \log 2$. Exponentiating and using $H_j < 1 + \log j$ gives the result.

This result shows that, after a reasonable number of choices of u, one can be found such that $f_u^b = f^b$. However, Algorithm 1 will find f^b faster in general because, in effect, it computes the lcm of several of the f_u^b. Suppose f_u^b is computed for k randomly selected values of u. In analogy with (11) the probability that the lcm of the k polynomials is f^b is

$$\Phi_k(f) = \prod_i (1 - q^{-kd_i}). \tag{12}$$

Note that for $k > 1$

$$\begin{aligned} \Phi_k(f) &\ge 1 - \sum q^{-kd_i} \\ &> 1 - \left(\frac{q^1}{1} q^{-k} + \frac{q^2}{2} q^{-2k} + \cdots \right) \\ &= 1 - \log\left(\frac{q^{k-1}}{q^{k-1} - 1} \right). \end{aligned}$$

Even for $k = 2$ this is more than 0.3, so Algorithm 1 has at least a 30 percent chance of succeeding after the second pass.

Now we move on to a probability calculation for the minimum polynomial of A itself, f^A. What is the probability, for b selected at random, that $f^b = f^A$? A lower bound on this probability is obtained using the fact that for at least one vector v, $f^v = f^A$. This fact can be easily proved by consideration of the rational canonical form of A or from first principles.

Proposition 4: For $b_1, \cdots, b_k$ selected uniformly at random, the probability that $\operatorname{lcm}(f^{b_1}, \cdots, f^{b_k}) = f^A$ is at least $\Phi_k(f^A)$.

Proof: Let A be an arbitrary $n \times n$ matrix and v a length n vector such that $f^v = f^A$. For vectors $b_1, \cdots, b_k$, let $f_{b_j}^v$ be the minimum polynomial of $\{(b_j, A^i v)\}_{i=0}^{\infty}$. Let h be the lcm of the $f_{b_j}^v$. It is known that $h = f^v = f^A$ with probability exactly $\Phi_k(f^A)$.

Let B be the transpose of A. Since $\{(b_j, A^i v)\}_{i=0}^{\infty} = \{(v, B^i b_j)\}_{i=0}^{\infty}$, h is the minimum degree polynomial such that $(v, h(B)B^i b_j)$ is zero for all i and j. Therefore, if g is the miniumum degree polynomial such that $g(B)b_j = 0$ for all j, then $h|g$. However, g divides the minimum polynomial of B, $f^B = f^A$. Thus if $h = f^A$, $g = f^B$, so with probability at least $\Phi_k(f^B)$ the lowest degree polynomial g, such that $g(B)b_j = 0$ for $j = 1, 2, \cdots, k$, equals f^B. This proves the theorem for the transpose of an arbitrary square matrix A, and so the theorem holds for every square matrix.

VII. Concluding Remarks

Some applications require the solution of sparse linear equations over the integers modulo m. Using the Chinese remainder theorem, we can reduce this to the problem of finding a solution to a sparse system modulo p^k for each prime p dividing m. Suppose we wish to solve (1) modulo p^k where A is square and $\det(A) \not\equiv 0 \pmod p$. First find a solution y to (1) modulo p. Then if x is a solution to the original equation, $A(x - y) = b - Ay \pmod{p^k}$. Each coordinate of the right-hand side is divisible by p. Then dividing each side by p reduces the problem to solving $Az = b' \pmod{p^{k-1}}$. Once this is solved, $x = y + pz$ solves the original equation. Repetition of this reduction allows solving the original problem by solving k equations modulo p.

For the most part, the algorithms in this paper are probabilistic, and it would be of some interest to have deterministic versions and, more importantly, to remove in some algorithms the possibility of error. In addition, it

would be of interest to know if there is a fast method of computing the inverse of a sparse matrix. Another question to ask is whether a rapid method exists for finding the characteristic polynomial. This seems likely because the minimum polynomial can be computed, and the characteristic polynomial can be evaluated at any specific value.

With one exception the methods in this paper apply the coefficient matrix to column vectors and not to row vectors. For sparse matrices the distinction is probably unimportant, but situations may arise where application of a row vector may be difficult. At the beginning of Section III a technique of finding a dependent row in an $(n + 1) \times n$ matrix M was given. This involved transposing M and removing the final column, say b, to obtain a square matrix A. Attempting to solve $Ax = b$ produced a row dependency in M.

Briefly, we indicate how to salvage this technique when M can only be applied to column vectors. If A is the operator formed earlier, it is easy to apply A^T to any column vector. Then let us estimate the minimum polynomial of A^T to be, say, $g(z)$, with very small chance of error. The minimum polynomial of A is the same polynomial. If $g(0) \neq 0$, A can be assumed to be nonsingular and the final row of M is dependent on the others, because $Ax = b$ will have a solution. Suppose $g(0) = 0$. Then $A \cdot g^-(A) = 0$, but $g^-(A)$ is not the zero matrix. Since A is easily applied to row vectors, products $u^T g^-(A)$, where u^T is any row vector, can be computed in $O(n(\omega + n))$ operations. Beginning with a randomly selected u^T such that this product is nonzero, and using binary search, a unit vector e_i^T is found such that $e_i^T g^-(A) \neq 0$. This means $e_i^T g^-(A) e_j \neq 0$ for some unit vector e_j. Then $x = g^-(A) e_j$ is a nonzero solution to $Ax = 0$. Furthermore, if e_i has a one at position i, the ith component of x is nonzero, so the ith row of M is dependent on the others.

The method just described is not as nice as the method that permits application of row vectors to M. A much more ambitious task is to find a fast algorithm for solving $x^T A = b^T$ by application of A to column vectors only. We do not have an algorithm that does this.

Linear operators do exist that are easy to apply but whose matrices are not sparse. For example, x may be considered as a sequence of field elements, and convolution of this sequence with an arbitrary fixed sequence is such an operator. The author has not been able to find an example of a linear operator that is easy to apply but whose transpose is difficult to apply. If it can be shown that no such operator exists, then the restriction to application to column vectors is of little importance.

Acknowledgment

The author is grateful for the advice of M. Kaminski, R. C. Mullin, A. M. Odlyzko, and an anonymous referee in preparing this article.

References

[1] D. Coppersmith, "Fast evaluation of logarithms in fields of characteristic two," *IEEE Trans. Inform. Theory*, vol. IT-30, pp. 587–594, 1984.

[2] C. Pomerance, "Analysis and comparison of some integer factoring algorithms," in *Computational Number Theory*, part 1, H. W. Lenstra, Jr., and R. Tijdeman, Eds. Amsterdam, The Netherlands: Math. Centre, 1982, pp. 89–139.

[3] A. M. Odlyzko, "On the complexity of computing discrete logarithms and factoring integers," in *Fundamental Problems in Communication and Computation*, B. Gopinath and T. Loven, Eds. New York: Springer, to appear.

[4] E. R. Berlekamp, *Algebraic Coding Theory* New York: McGraw-Hill, 1968.

[5] J. L. Massey, "Shift-register synthesis and BCH decoding," *IEEE Trans. Inform. Theory*, vol. IT-15, pp. 122–127, 1969.

[6] W. H. Mills, "Continued fractions and linear recurrences," *Math. Comput.*, vol. 29, pp. 173–180, Jan. 1975.

[7] N. Zierler, "Linear recurring sequences and error-correcting codes," in *Error Correcting Codes*, H. B. Mann, Ed. New York: Wiley, 1968, pp. 47–59.

[8] R. P. Brent, F. G. Gustavson, and D. Y. Y. Yun, "Fast solution of Toeplitz systems of equations and computation of Pade approximants," *J. Algorithms*, vol. 1, pp. 259–295, 1980.

[9] A. Schönhage, "Schnelle Multiplikation von Polynomen über Körpen der Charakteristik 2," *Acta Inform.*, vol. 7, pp. 395–398, 1977.

[10] F. J. MacWilliams and N. J. A. Sloane, *The Theory of Error-Correcting Codes*. New York: North-Holland, 1977.

[11] J. T. Schwartz, "Fast probabilistic algorithms for verification of polynomial identities," *J. Ass. Comput. Mach.*, vol. 27, pp. 701–717, Oct. 1980.

[12] R. L. Rivest, A. R. Meyer, D. J. Kleitman, and K. Winklmann, "Coping with errors in binary search procedures," *J. Comput. Syst. Sci.*, vol. 20, p. 396–404, 1980.

[13] V. E. Benes, "Optimal rearrangeable multistage connecting networks," *Bell Syst. Tech. J.*, vol. 43, pp. 1641–1656, 1964.