New Deterministic Parallel Algorithms for the Characteristic Polynomial of a Matrix over Abstract Fields *

Victor Y. Pan
Department of Mathematics & Computer Science,
Lehman College, CUNY
Bronx, NY 10468
Internet address: vpan@lcvax.lehman.cuny.edu

February 19, 1999

Abstract

We study deterministic parallel computation of the characteristic polynomial of an $n \times n$ matrix A over an arbitrary field F of a positive characteristic $c \leq n$. Our new NC algorithms run in polylogarithmic time and simultaneously enable us to decrease the known upper bounds on processor complexity by order of magnitude except for matrices A forming an algebraic variety of codimension $d \geq c - 1$ in the space of $n \times n$ matrices. Our techniques of the algorithm design may be of independent interest.

Key words: Characteristic polynomial, deterministic algorithms, parallel complexity of computations in finite fields.

1991 Math. Subject Classification: 12Y05, 68Q40, 68Q25

1 Introduction: the problem, model of computation, background and our results

Our subject is the parallel computation (over an arbitrary field $\mathbf{F}$) of the characteristic polynomial $C_A(x) = \det(xI - A)$ of an $n \times n$ matrix A. We

*Supported by NSF Grants CCR 9625344 and CCR 9732206 and by PSC CUNY Awards 668365 and 669363

are interested in estimating the arithmetic complexity of this computation provided that A is either an arbitrary $n \times n$ matrix or a *generic* matrix, that is, a matrix from an algebraically open domain $\mathbf{D}$ in $\mathbf{F}^{n\times n}$, $\mathbf{D} = \mathbf{F}^{n\times n} - \mathbf{V}$, $\mathbf{V}$ being an algebraic variety of codimension $d \geq 1$ in $\mathbf{F}^{n\times n}$. (In this case we will call A a *d–generic matrix.*)

The sequential arithmetic complexity of this problem over any field $\mathbf{F}$ (to be denoted by $CP(n)$) is closely related to the complexity, $M(n)$, of $n \times n$ matrix multiplication, $M(n) = \Omega(n^2)$, $M(n) = O(n^\omega)$, $\omega < 2.376$. Namely, we have quite tight bounds, $CP(n) = O(M(n) \log n)$ [K-G85], $CP(n) = \Omega(M(n))$ ([BP94], exercise 2.6, p. 213). Furthermore, for 1-generic matrices A we have $CP(n) = O(M(n))$ [K-G85].

To state the parallel complexity estimates, we will assume the customary arithmetic EREW PRAM model of parallel computing [KR90], under which every processor is capable of performing in unit time a single operation in $\mathbf{F}$.

We will write $O(t, p)$ to denote the simultaneous bounds $O(t)$ on time and $O(p)$ on the number of the PRAM processors. By a variation of Brent's principle [KR90], [BP94], $O(t, pq)$ implies $O(qt, p)$ for any $q \geq 1$. In particular $O(t, p)$ implies $O(pt, 1)$, that is, the sequential time bound $O(pt)$.

Over any field $\mathbf{F}$, we have the following bounds on the parallel complexity of matrix multiplication (cf. [P87]):

$$(t_M, p_M) = O(\log n, P(n)) = O(\log n, n^{w^*}), w^* < 2.376, \tag{1.1}$$

$w^* < w + \epsilon$ for any fixed positive ϵ , and we have the bounds

$$(t_{CP}, p_{CP}) = O(\log^2 n, nP(n)) = O(\log^2, n^{3.376}) \tag{1.2}$$

on the parallel complexity of the computation of $C_A(x)$ [Be84], [Ch85].

A well known open problem is the design of an algorithm with polylogarithmic parallel time and with the product of t_{CP} and p_{CP} of order less than the above bound $nP(n)$. (Note that $(t_{CP})(p_{CP}) = \Omega(M(n))$.) Some partial progress in this direction was obtained in [PS78], [GP89], and [HP98] (based on refining the Leverrier-Csanky algorithm of [Cs76]) under the assumption that the field $\mathbf{F}$ has characteristic $c = 0$ or $c > n$. In particular, we have the bounds

$$(t_{CP}, p_{CP}) = O(\log^2 n, P(n)n^{0.5-\delta}) \tag{1.3}$$

for some $\delta > 0$ [GP89] and

$$(t_{CP}, p_{CP}) = O(\log^2 n, n^{2.837}) \tag{1.4}$$

[HP98].

By using randomization and assuming that A is 1-generic, one may obtain nearly optimal bounds over any field $\mathbf{F}$:

$$(t_{CP}, p_{CP}) = O((\gamma(n,c)\log n)^2, (P(n)\log\log(n/c))/\gamma(n,c)^2), \tag{1.5}$$

$$\gamma(n,c) = (\log n)/\log c \tag{1.6}$$

(see [KP91], [KP94], [P96] and our section 3).

In our present paper, we yield further progress by proving some *deterministic upper bounds* of this type for d-generic input matrices A where $d \geq c-1$. For the remaining matrices (to which our approaches do not apply), one may still apply the algorithms of [Be84] and [Ch85]. The next theorem summarizes our cost estimates in terms of c and n.

Theorem 1.1 *The characteristic polynomial of a (c–1)-generic $n \times n$ matrix over any field* $\mathbf{F}$ *can be computed at the deterministic cost bounded by*

$$(t_{CP}, p_{CP}) = O(\log^2 n + (\gamma(n,c)\log(n/c))^2, P^*(n)), \tag{1.7}$$

for $\gamma(n,c)$ of (1.6), for $P^*(n) = \min\{n^{2.837}, P(n)n^{0.5-\alpha}\}$ and for some positive α, and also by

$$(t_{CP}, p_{CP}) = O((\gamma(n,c)\log n)^2, P(n)(c^{1/2} + \log\log(n/c)/(\gamma(n,c))^2). \tag{1.8}$$

We will also point out some approaches to the expansion of the class of input matrices to which our more efficient parallel computation of $C_A(x)$ applies (cf. in particular, our bound (4.2), including polylog time and roughly n^3 processors and covering the class of $(\log n)^{O(1)}$ – generic input matrices A, and our algorithm 4.1, both in section 4).

Our algorithms supporting Theorem 1.1 and the latter heuristic approaches exploit the algorithm developed in [P96], [P97], [P99] that recovers a divisor of $C_A(x)$ from the power sums of its zeros. The power sums are computed based on the algorithms of [Cs76], [PS78], [GP89] and [HP98]. The proof of bound (1.7) is based on the combination of these two approaches and is given in the next section. Bound (1.8) is proved in section

3 based on the extension and randomization of the randomized algorithm of [Wi86], [KP91], [P96], which starts with computing the Krylov sequence $\{A^i\mathbf{v}, i = 1, 2, \ldots, 2n\}$ for a fixed or random vector $\mathbf{v}$ (we call this *Krylov-Wiedemann's approach*). Some heuristic algorithms based on the combination of the presented approaches are shown in section 4.

We hope that, besides our improvement of the parallel complexity estimates for a fundamental problem of matrix computations, our study of the power sum techniques, Krylov-Wiedemann's and other techniques as well as their combinations is of independent interest and provides some new insight into the problem of computing the characteristic polynomial of a matrix.

Achnowlegement. I thank the referee for helpful suggestions.

2 Trace of powers/power sum approach

Our study of the algorithms for the computation of $c_A(x) = \sum_{i=0}^{n} c_i x^i = \prod_{j=1}^{n}(x - x_j)$, the characteristic polynomial of an $n \times n$ matrix A, where $c_n = 1$, $x_1, \ldots, x_n$ are the eigenvalues of A, will start with the algorithms based on the computation of the traces of the powers of A,

$$s_l = \operatorname{trace}(A^k) = \sum_{j=1}^{n} x_j, \quad k = 1, 2, \ldots, \tag{2.1}$$

which are the power sums of the eigenvalues of A. In particular, we recall that the bounds (1.3), (1.4) were deduced by first

a) computing the power sums $s_1, \ldots, s_m$ for $m = O(n)$ and then

b) recovering $C_A(x)$ from $s_1, \ldots, s_m$.

Stage a) was reduced to multiplications of some rectangular matrices [Cs76], [GP89], which were performed within the cost bounds (1.3) and (1.4) in [GP89] and [HP98], respectively. Stage b) has a much less expensive solution over the fields $\mathbf{F}$ of characteristics $c = 0$ and $c > n$.

Over an arbitrary field $\mathbf{F}$, the characteristic polynomial $C_A(x)$ is not well defined by the sequence $s_1, s_2, \ldots, s_m$ for any m. Indeed, suppose that an eigenvalue x_j of A has algebraic multiplicity c, that is, is repeated exactly c times in the list of the eigenvalues. Then its replacement in this list by any other c-multiple value z_j would change $c_A(x)$ but not $s_1, s_2, \ldots$.

Let us next assume that we are given a natural $n > 1$, a field $\mathbf{F}$ of characteristic c, $2 \le c \le n$, and the power sums $s_1, s_2, \ldots$, and that we are looking for the greatest divisor $C_A^-(x) = \sum_{i=1}^{u} c_i^- x^i$, $c_u^- = 1, u \le n$, of the polynomial $C_A(x) = \prod_{j=1}^{n}(x - x_j)$ such that (2.1) holds and $C_A^-(x)$

has no divisors of the form $(x - x_j)^c$. (The latter restriction is due to the above observation that the factors $(x - x_j)^c$ contribute nothing to the power sums $s_1, s_2, \ldots$ over the field $\mathbf{F}$.) It is convenient for us to compute at first the polynomial $V_A(y) = \sum_{i=0}^{h} v_i y^i = y^u C_A^-(1/y)$, which is the reverse polynomial of $C_A^-(x)$, where $v_0 = 1, v_i = c_{u-i}^-$, $i = 0, 1, \ldots, h$, and $h = u - l$ for some nonnegative l. Clearly, $C_A^-(x)$ can be immediately recovered from $V_A(y)$.

Now, we recall the algorithm of [P97], [P99] (cf. also [P96]) that at cost bounded by

$$(t_{CP}^-, p_{CP}^-) = O((\gamma(n,c)\log(n/c))^2 + \log n, (n/\gamma(n,c))^2 \log\log n) \quad (2.2)$$

(for $\gamma(n,c)$ of (1.6)) computes over $\mathbf{F}$ a polynomial $P(y)$ of the minimal degree m, defined by the power sums s_k of its zeros and the normalization condition $P(0) = 1$. Let us prove

Fact 2.1 $P(y) = V_A(y)$, *provided that both polynomials* $P(y)$ *and* $V_A(y)$ *share all the power sums of their zeros.*

Proof. Represent $P(y)$ as $\prod_{j-1}^{m}(1 - y_j y)$ and deduce that $-\frac{P'(y)}{P(y)} = \sum_{j=1}^{m} \frac{y_i}{1-yy_j} = \sum_{j=1}^{m} y_i \sum_{k=0}^{\infty}(yy_i)^k = \sum_{k=0}^{\infty}(\sum_{j=1}^{m} y_j^{k+1})y^k = S(y)$, for $S(y) = \sum_{k=0}^{\infty} s_{k+1}y^k$. Similarly, $-\frac{V_A'(y)}{V_A(y)} = S(y)$. Therefore, $V_A'(y)P(y) = V_A(y)P'(y)$. It follows that every linear divisor $1 - yy_j$ of $V_A(y)$ divides $P(y)$ and vise versa. By dividing $P(y)$ and $V_A(y)$ by their greatest common divisor, $g(y)$, we arrive at the two polynomials, $\overline{P}(y)$ and $\overline{V}_A(y)$, still sharing all the power sums of their zeros. Then we obtain that $\overline{P}'(y)\overline{V}_A(y) = \overline{P}(y)\overline{V}'_A(y)$. Since $\overline{P}(y)$ and $\overline{V}_A(y)$ have only constant common divisors and since $\overline{P}(0) = \overline{V}_A(0) = 1$, it follows that $\overline{P}(y) = \overline{V}_A(y)$. Multiply both sides of this equation by $g(y)$ and arrive at Fact 2.1. $\square$

Fact 2.1 enables us to extend cost bound (2.2) to the computation of $V_A(x)$ and $C_A^-(x)$, provided that all the power sums s_k of (2.1) have been computed.

Comparing (2.2) with (1.7), we can see that the overall cost of computing $C_A^-(x)$ is within the bounds (1.7).

By the definition of $C_A^-(x)$, we immediately obtain that $C_A(x) = C_A^-(x)$ if and only if deg $C_A^-(x) > n - c$. (Otherwise $C_A(x)$ can be equal to any polynomial of the form $C_A^-(x)\prod_j(x - z_j)^c$ for any set $\{z_j \in \mathbf{F}\}$).

To prove bounds (1.7) of theorem 1.1, it remains to show that deg $C_A^-(x) \leq n - c$ (or, equivalently, that $C_A(x) = \det(xI - A)$ has a zero of multiplicity at least c) only for matrices A forming an algebraic variety of codimension $c-1$ in $\mathbf{F}^{n\times n}$. Recall that the zeros of $C_A(x)$ of multiplicity c are the common zeros of $C_A(x) = C_A^{(0)}(x)$ and its derivatives $C_A^{(h)}(x)$ of order h for $h = 1, \ldots, c-1$. The existence of such a zero implies that $R(C_A^{(h)}(x), C_A^{(h-1)}(x)) = 0$ for $h = 1, \ldots, c-1$ (that is, the $c-1$ resultants of the pairs of the polynomials $(C_A^{(h)}(x), C_A^{(h-1)}(x))$ vanish). This imposes $c-1$ algebraically undependent polynomial equations (of degrees at most n) on the coefficients of the above polynomials and, consequently, on the entries of A, which completes the proof of estimates (1.7) of Theorem 1.1. (In fact, all the resultants are polynomials in the entries of A of degrees at most n^2). $\square$

3 Krylov-Wiedemann's approach

In this section, we rely on an alternative approach to the computation of $C_A(x)$, where we start with a pair of fixed or random vectors $\mathbf{u}$ and $\mathbf{v}$ of dimension n and compute the vector

$$\mathbf{h} = (\mathbf{u}^T A^i \mathbf{v})_{i=0}^{2n}. \tag{3.1}$$

Remark 3.1 *The computation of the vector* $\mathbf{h}$ *for given* $\mathbf{u}$, $\mathbf{v}$ *and* A *starts with the computation of the Krylov classical sequence of vectors* $\{A^i\mathbf{v}, i = 0, 1, \ldots\}$. *Wiedemann in [Wi86] used the vector* $\mathbf{h}$ *of (3.1) for randomized matrix computations in finite fields. Lemma 2 of [KP91] shows that, under a random choice of the components of* $\mathbf{u}$ *and* $\mathbf{v}$ *independently of each other from a fixed finite set* S *in* $\mathbf{F}$ *(assuming the uniform probability distribution on* S*), the minimum polynomials,* $m_A(x)$ *of the matrix* A *and* $m_{A,\mathbf{u},\mathbf{v}}(x)$ *of the sequence, coincide with each other with a probability at least* $1 - 2$ *deg* $m_A(x)/$ *card* (S). *We will not use these results and will produce deterministic algorithm though for* $(c-1)$*-generic (rather than for arbitrary) input matrix.*

The cost of the computation of the vector $\mathbf{h}$ is $O(\log^2 n, P(n))$ (cf. e.g. [BP94], p. 97). The cost of the computation of $m_{A,\mathbf{u},\mathbf{v}}(x)$ for a given $\mathbf{h}$ is bounded by $O((\gamma(n,c)\log n)^2, (n/\log\gamma(n,c))^2 \log\log n)$ (see [P96]), and we arrive at the polynomial $m_{A,\mathbf{u},\mathbf{v}}(x)$ at the cost within bound (1.8). To complete the proof of Theorem 1.1, we need to show an efficient way to the recovery of the polynomial $C_A(x)$ from $m_{A,\mathbf{u},\mathbf{v}}(x)$ for $(c-1)$-generic matrix A. We recall that $m_{A,\mathbf{u},\mathbf{v}}(x)$ divides $C_A(x)$, write

$$q(x) = C_A(x)/m_{A,\mathbf{u},\mathbf{v}}(x), \tag{3.2}$$

$d = \deg\, m_{A,\mathbf{u},\mathbf{v}}(x)$, $f = \deg\, q(x) = n-d$, and assume that $c > f = n-d$.

Next, we recall an algorithm of [PS78], which computes (over any field) the traces $s_k = \text{trace}(A^k)$, $k = 1, \ldots,\ O(n)$, at the cost bounded by (1.3) for $\delta = 0$. We apply this algorithm to solve the simplified task of computing the first $f-1$ traces $s_1, \ldots, s_{f-1}$, in which case the cost decreases to $O(\log^2 n, c^{1/2}P(n))$. Now, we apply the algorithm of [P96], [P97], [P99], which uses the values $s_1, \ldots, s_{f-1}$ to recover the f leading coefficients, $c_0 = 1, c_1, \ldots, c_{f-1}$ of $C_A(x)$ at the cost $O(\log^2 f, f/\log f)$.

Now, we recall (3.2) and easily recover $q(x)$ from $m_{A,\mathbf{u},\mathbf{v}}(x)$ and from the f leading coefficients of $C_A(x)$. Here is a simple may for doing this. Fix any polynomial of degree n with the leading coefficients $1, c_1,\ \ldots, c_{f-1}$. Divide it by $m_A(x)$, ignore the remainder and output the quotient as $q(x)$. Finally, recover $C_A(x)$ from (3.2). This algorithm gives us $C_A(x)$ unless

$$d = \deg\, m_{A,\mathbf{u},\mathbf{v}}(x) \leq n-c. \tag{3.3}$$

To obtain bound (1.8) of Theorem 1.1, it remains to prove that relations (3.2) imply at least c algebraically independent polynomial equations on the entries of A. This follows from the well known fact [KP91], [BP94] that deg $m_{A,\mathbf{u},\mathbf{v}}(x) = \text{rank}\ H_{u,v}$ where $H_{u,v}$ is an $n \times n$ matrix, $H_{u,v} = (h_{i,j})_{i,j=0}^{n-1}$, and $h_{i,j} = \mathbf{u}^T A^{i+j}\mathbf{v},\ \ i,j = 0, \ldots n-1$. The entry $h_{i,j}$ of $H_{u,v}$ is a polynomial of degree $i+j$ in the entries of the matrix A, and the relation rank $H_{u,v} \leq n-c+1$ (required to have $d \leq n-c+1$) implies the vanishing of $c-1$ algebraically independent polynomials in the entries of A (that is, of some subdeterminants of $H_{u,v}$). This completes the proof of Theorem 1.1. □

4 A combination of the power sum approach and the Krylov/Wiedemann approach

Suppose that the algorithms of the two preceding sections computed two polynomials $C_A^-(x)$ and $m_{A,\mathbf{u},\mathbf{v}}(x)$, which both divide $C_A(x)$. If follows that the least common multiple (l.c.m.), $L(x)$, of these two polynomials also divides $c_A(x)$. If deg $L(x) = n$, then $C_A(x) = L(x)$, and we solve our task. Even if we just have deg $L(x) > n-c$, we may recover $C_A(x)$ from its divisor $L(x)$ by applying the techniques of the preceding section.

To extend this approach to the case where

$$\lambda = n - \deg\, L(x) \geq c, \tag{4.1}$$

it suffices to add another algorithm for the computation of the λ leading coefficients of $C_A(x)$. Here is the solution of the latter task by means of a simple extension of the Samuelson-Berkowitz algorithm [FF63], [Be84]. The bottleneck stage of this algorithm is the computation of the products $M_t^i \mathbf{s}_t$ where M_t are $(n-t) \times (n-t)$ submatrices of A, $\mathbf{s}_t$ are subvectors of some columns of A, $t = 1, \ldots, n-1$, and i ranges from 1 to λ if we are seeking the λ leading coefficients of $C_A(x)$. If c and λ are small, we may (successively) λ times multiply M_t by vectors, by applying the straightforward algorithm (concurrently for all t). We arrive at the overall cost bound $O(\lambda \log n, n^3/\log n)$ for all multiplications. Together with (2.2), the latter bound immediately leads us to the following estimate:

$$(t_{CP}, p_{CP}) = O((\lambda + (\gamma(n,c))^2 \log n) \log n, n^3/\log n), \tag{4.2}$$

which can be slightly refined by using Brent's principle. As long as λ is polylogarithmic in n, we still have polylogarithmic time bound, and simultaneously a processor bound improving the known one (of [Be84] and [Ch85]) by roughly factor $n^{0.376}$. The result holds for $(\lambda - 1)$-generic input matrix A. If λ of (4.1) is large, we may try to obtain a higher degree divisor of $C_A(x)$ by applying the following algorithm, which uses factorization of a polynomial over $\mathbf{F}$.

Algorithm 4.1

Input: $C_A^-(x)$, $L(x)$ $\mathbf{F}, n, c$.

Output: a polynomial $L^+(x)$ that both divides $C_A(x)$ and is a multiple of $L(x)$.

Computations:

1. Compute the polynomial $m(x) = L(x)/C_A^-(x)$. (Note that $m(x)$ divides $m_A(x)$, has only constant common divisors with $C_A^-(x)$, and has the maximal degree under these two conditions.)

2. Compute the complete factorization of $m(x)$ over $\mathbf{F}$, $m(x) = \prod_{j=1}^s (m_j(x))^{d_j}$.

3. Compute the polynomial $\Delta(x) = \prod_{j-1}^s (m_j(x)^{d_j^+ - d_j}, d_j^+ = [d_j/c], j = 1, \ldots, s$.

4. Compute and output $L^+(x) = L(x)\Delta(x)$.

Correctness of the algorithm follows because the multiplicity of each zero of the polynomial $C_A(x)/C_A^-(x)$ is divisible by c (see section 2). Since

$L^+(x)$ divides $C_A(x)$, we have $L^+(x) = C_A(x)$ if and only if deg $L^+(x) = n$. (The latter equation must hold if deg $L(x) > n - c$, but for a large class of matrices A it holds even where deg $L(x) \leq n - c$.) Stages 1, 3 and 4 of the algorithm are reduced to polynomial division and multiplications, whose overall cost is clearly dominated by (1.7) and (1.8) (cf. [BP94]), and so is also the cost of the computation of the l.c.m. $L(x)$. At stage 2, we may apply parallel versions of the algorithms of [GS92], [KS95/98] (cf. also [HP98]). If the cardinality q of $\mathbf{F}$ is bounded, say, by $n^{O(1)}$, the expected parallel cost of performing these algorithms is also dominated by (1.7) and (1.8).

References

[Be84] S. Berkowitz, On Computing the Determinant in Small Parallel Time Using a Small Number of Processors, *Information Processing Letters*, 18, 3, 147–150, 1984.

[BP94] D. Bini, V.Y. Pan, *Polynomial and Matrix Computations, Volume 1: Fundamental Algorithms*, Birkhäuser, Boston, 1994.

[Ch85] A.L. Chistov, Fast Parallel Calculation of the Rank of Matrices over a Field of Arbitrary Characteristics, *Proc. FCT 85, Lecture Notes in Computer Science*, 199, 63–69, Springer, Berlin, 1985.

[Cs76] L. Csanky, Fast Parallel Matrix Inversion Algorithm, *SIAM J. Comput.,* 5, 4, 618-623, 1976.

[FF63] D.K. Faddeev, V.N. Faddeeva, *Computational Methods of Linear Algebra*, W.H. Freeman, San Francisco, 1963.

[GP89] Z. Galil, V. Y. Pan, Parallel Evaluation of the Determinant and of the Inverse of a Matrix, *Information Processing Letters,* 30, 41–45, 1989.

[GS92] J. von zur Gathen, V. Shoup, Comuting Frobenius Maps and Factoring Polynomials, *Comput. Complexity,* 2, 187-224, 1992.

[HP98] X. Huang, V. Y. Pan, Fast Rectangular Matrix Multiplication and Applications, *J. of Complexity*, 14, 257-299, 1998.

[K-G85] W. Keller-Gehrig, Fast Algorithms for Characteristic Polynomial, *Theoretical Computer Science,* 36, 309-317, 1985.

[KP91] E. Kaltofen, V.Y. Pan, Processor Efficient Parallel Solution of Linear Systems over an Abstract Field, *Proc. 3rd Ann. ACM Symp. on Parallel Algorithms and Architectures*, 180-191, ACM Press, New York, 1991.

[KP94] E. Kaltofen, V.Y. Pan, Parallel Solution of Toeplitz and Toeplitz-like Linear Systems over Fields of Small Positive Characteristic, *Proc. First Intern. Symposium on Parallel Symbolic Computation* (PASCO '94), Linz (edited by Hoon Hong), Austria (Sept. 1994), *Lecture Notes in Computing,* 5, 225-233, World Scientific Publishing Company, Singapore, 1994.

[KR90] R. Karp, V. Ramachandran, A Survey of Parallel Algorithms for Shared Memory Machines, *Handbook for Theoretical Computer Science* (J. van Leeuwen Editor), 869-941, North Holland, Amsterdam, 1990.

[KS95/98] E. Kaltofen, V. Shoup, Subquadratic-Time Factoring of Polynomials over Finite fields, *Proc. 27th Annual Symposium on Theory of Comput.,* 398-406, ACM Press, New York, 1995; and *Math. Comput.,* 67, 223, 1179-1197, 1998.

[P87] V. Y. Pan, Complexity of Parallel Matrix Computations, *Theoretical Computer Science*, 54, 65–85, 1987.

[P96] V. Y. Pan, Parallel Computation of Polynomial GCD and Some Related Parallel Computations over Abstract Fields, *Theoretical Computer Science*, 162, 2, 173-223, 1996.

[P97] V. Y. Pan, Fast Solution of the Key Equation for Decoding the BCH Error-Correcting Codes, *Proc. 27th Ann. ACM Symp. on Theory of Computing*, 168-175, ACM Press, New York, 1997.

[P99] V. Y. Pan, New Techniques for Decoding BCH Error-Correcting Codes, preprint, 1999.

[PS78] F. P. Preparata, D. V. Sarwate, An Improved Parallel Processor Bound in Fast Matrix Inversion, *Information Processing Letters*, 7, 3, 148–149, 1978.

[Wi86] D.H. Wiedemann, Solving Sparse Linear Equations Over Finite Fields, *IEEE Trans. on Inf. Theory,* IT-32, 1, 54–62, 1986.