JOURNAL OF ALGORITHMS 1, 259–295 (1980)

Fast Solution of Toeplitz Systems of Equations and Computation of Padé Approximants

RICHARD P. BRENT,* FRED G. GUSTAVSON,† AND DAVID Y. Y. YUN†

**Department of Computer Science, Australian National University, Box 4, Canberra, ACT 2600, Australia; and †Mathematical Sciences Department, IBM T. J. Watson Research Center, P.O. Box 218, Yorktown Heights, New York 10598*

Received December 22, 1979; revised April 20, 1980

We present two new algorithms, ADT and MDT, for solving order-n Toeplitz systems of linear equations $Tz = b$ in time $O(n \log^2 n)$ and space $O(n)$. The fastest algorithms previously known, such as Trench's algorithm, require time $\Omega(n^2)$ and require that all principal submatrices of T be nonsingular. Our algorithm ADT requires only that T be nonsingular. Both our algorithms for Toeplitz systems are derived from algorithms for computing entries in the Padé table for a given power series. We prove that entries in the Padé table can be computed by the Extended Euclidean Algorithm. We describe an algorithm EMGCD (Extended Middle Greatest Common Divisor) which is faster than the algorithm HGCD of Aho, Hopcroft and Ullman, although both require time $O(n \log^2 n)$, and we generalize EMGCD to produce PRSDC (Polynomial Remainder Sequence Divide and Conquer) which produces any iterate in the PRS, not just the middle term, in time $O(n \log^2 n)$. Applying PRSDC to the polynomials $U_0(x) = x^{2n+1}$ and $U_1(x) = a_0 + a_1 x + \cdots + a_{2n} x^{2n}$ gives algorithm AD (Anti-Diagonal), which computes any (m,p) entry along the antidiagonal $m + p = 2n$ of the Padé table for U_1 in time $O(n \log^2 n)$. Our other algorithm, MD (Main-Diagonal), computes any diagonal entry (n,n) in the Padé table for a normal power series, also in time $O(n \log^2 n)$. MD is related to Schönhage's fast continued fraction algorithm. A Toeplitz matrix T is naturally associated with U_1, and the (n,n) Padé approximation to U_1 gives the first column of T^{-1}. We show how a formula due to Trench can be used to compute the solution z of $Tz = b$ in time $O(n \log n)$ from the first row and column of T^{-1}. Thus, the Padé table algorithms AD and MD give $O(n \log^2 n)$ Toeplitz algorithms ADT and MDT. Trench's formula breaks down in certain degenerate cases, but in such cases a companion formula, the discrete analog of the Christoffel–Darboux formula, is valid and may be used to compute z in time $O(n \log^2 n)$ via the fast computation (by algorithm AD) of at most four Padé approximants. We also apply our results to obtain new complexity bounds for the solution of banded Toeplitz systems and for BCH decoding via Berlekamp's algorithm.

1. INTRODUCTION

We present two new algorithms for solving $(n + 1) \times (n + 1)$ Toeplitz systems of linear equations $Tz = b$. Trench's algorithm [31] and other similar algorithms [2, 3, 7, 21, 27, 28, 33] all require time $\Omega(n^2)$, but our

0196-6774/80/030259-37$02.00/0

algorithms take only time $O(n \log^2 n)$ on a random-access machine [1]. One algorithm, ADT, is based on the fast Extended Euclidean Algorithm (EEA), and requires only that T be nonsingular. The other algorithm, MDT, was derived from fast continued fraction algorithms (Schönhage [29]) and does not use the Euclidean algorithm. Like Trench's algorithm, it requires that certain minors of T be nonzero. We note here that a Hankel matrix is just a Toeplitz matrix with its columns reversed. Thus our Toeplitz algorithms apply to the solution of Hankel systems with merely a change of notation.

Solving systems of Toeplitz equations is closely related to finding a type of rational function approximation to power series, known as Padé approximants. For a power series $A(x) = a_0 + a_1 x + a_2 x^2 + \ldots$, the (m,p) Padé approximant to A is the rational function $R_{mp}(x) = U(x)/V(x)$, where U and V are polynomials with $\deg(U) \leq m$ and $\deg(V) \leq p$. The set of Padé approximants to $A(x)$ collected in a two-dimensional array indexed by (m,p) is called the Padé table [13]. The entries of the Padé table can be computed by the Extended Euclidean Algorithm. McEliece and Shearer [23] made this discovery by generalizing the work of Sugiyama *et al.* [30], who showed that Euclid's algorithm could be used to solve the key equation of Goppa codes. Our result was discovered prior to publication of [23]. Warner [32] has shown that all entries of the Hermite Rational Interpolation Table can be computed by Kronecker's Division Algorithm [19]. The Padé table is a special case of the Rational Interpolation Table, and Warner's result applied to the Padé table is essentially the same as McEliece and Shearer's. In fact, we have discovered that Kronecker's algorithm is essentially the Extended Euclidean Algorithm applied to computing elements of the Rational Interpolation Table. It seems that the original suggestion of Padé table entries being computable by Euclid's algorithm is due to Kronecker [15]. However, he applied the technique to the more general problem of rational Hermite interpolation and the division algorithm was named after him.

Our computational results are stronger than those mentioned above. We have improved and extended the HGCD algorithm of [1] and the fast GCD algorithm of Moenck [25] in two significant ways. First, we have developed an improved HGCD algorithm called EMGCD (for Extended Middle GCD). The cost of EMGCD is less than the cost of HGCD; however, both algorithms have $O(n \log^2 n)$ asymptotic cost. The second improvement comes from generalizing EMGCD. We have produced an algorithm PRSDC (Polynomial Remainder Sequence Divide and Conquer) which computes any iterate in the PRS sequence and *not* just the middle term [15]. The cost of algorithm PRSDC is also $O(n \log^2 n)$. The algorithm EEGCD of Yun [34] for the computation of the GCD of two polynomials along with the comultiplier polynomials is a special case of PRSDC.

Algorithm PRSDC has many useful applications. One example is the computation of the greatest common divisor of two polynomials P and Q. Our main application results in a fast computational algorithm for entries of the Rational Interpolation Table and, in particular, the Padé Table. We state such a result for the Padé table of the power series $A(x)$. Let $m + p = 2n$ and set $U_0 = x^{2n+1}$ and $U_1 = a_0 + a_1x + \cdots + a_{2n}x^{2n}$. By applying algorithm PRSDC to U_0 and U_1 we can compute any Padé table entry (m,p) along the antidiagonal $m + p = 2n$ in time $O(n \log^2 n)$. This special case of algorithm PRSDC will be referred to as algorithm AD (Anti-Diagonal).

Our other algorithm, called MD (Main-Diagonal), computes any main-diagonal entry (n, n) in the Padé table for a normal power series, also in time $O(n \log^2 n)$. MD was derived from a fast continued fraction algorithm similar to that of Schönhage [29], but we give a more direct derivation below. Theorem 4, on which algorithm MD is based, may be regarded as a generalization of some of the well-known identities of Frobenius [9].

We also present new complexity results for banded Toeplitz systems. Let T_{bc} be an $(n + 1) \times (n + 1)$ banded Toeplitz matrix whose semibandwidths are b and c, and let $w = b + c$. By applying algorithm PRSDC we can solve $T_{bc}z = h$ in time $O(n \log n) + O(w \log^2 w)$, which is an improvement over the $O(n \log n) + O(w^2)$ result due to Jain [16]. Morf and Kailath [17] give an $O(n \log w) + O(w^2)$ algorithm. Dickinson [6] showed an $O(nw) + O(w^2)$ algorithm. If our full Toeplitz algorithm is used to modify the Toeplitz inversion step of the Morf and Kailath algorithm, we easily obtain an $O(n \log w) + O(w \log^2 w)$ improvement. However, both algorithms require nonsingularity of two auxiliary matrices, while their algorithm imposes additional normality assumptions. With polynomial multiplication and division algorithms achieving $O(n \log w)$ (for degree n by degree w operations), we can modify our PRSDC and ADT algorithm to solve Toeplitz systems in $O(n \log w) + O(w \log^2 w)$ without any restrictions. We defer the details and proof of this algorithm to a forthcoming more specialized paper by Gustavson and Yun.

The rest of this paper is divided into eight sections as follows:

2. Padé Approximants

3. The Extended Euclidean Algorithm

4. The Antidiagonal Computation via the Extended Euclidean Algorithm

5. The Main Diagonal Algorithm

6. Fast Solution of Toeplitz Equations

7. Special Cases of Toeplitz Equations

8. Applications to Shift Register Synthesis and BCH Decoding

9. Summary and Conclusions

The section on Padé approximants presents two basic theorems which describe the Padé table associated with a power series $A(x)$. Theorem 1 is due to Frobenius, and we give a new proof which shows that the solution of a Toeplitz system already appears in this context. Theorem 2 is due to Gragg [13]; it describes the full structure of the Padé table. However, the reader who is mainly interested in the Toeplitz problem may omit this section.

This section on the Extended Euclidean Algorithm covers its algebraic and computational properties. We also present our new algorithm EMGCD and include a short description of EMGCD's enhancements relative to those of HGCD. Many useful quantities for the Toeplitz and Padé problems are, in fact, computed as by-products of this process. The major objective of this section is to establish the fast (i.e., $O(n \log^2 n)$) computation of these quantities. We also briefly describe the algorithm PRSDC and sketch a proof of its computational cost, $O(n \log^2 n)$.

Section 4 shows how the Euclidean algorithm can be used to compute entries of the Padé table. We prove two fundamental lemmas which describe exactly how the iterates of the Extended Euclidean Algorithm compute Padé approximants. These two lemmas imply our Theorem 3, which states that all entries along an antidiagonal of the Padé table are computed uniquely by the Extended Euclidean Algorithm. The computation necessary for the Toeplitz problem, namely, the (n, n) Padé approximant, can be carried out quickly and directly with the EMGCD algorithm stepping through the antidiagonal, hence the name algorithm AD. Thus, the reader merely interested in the Toeplitz computation need only understand the introductory paragraph of this section and the statement of Theorem 3.

In Section 5 we state and prove Theorem 4, which is the basis for the main-diagonal algorithm MD. Algorithm MD is, then, extended to a more general recursive algorithm MD2. We describe algorithms MD and MD2 and sketch a proof of their correctness. Unless the reader is interested in the details or the underlying techniques of algorithm AD or algorithm MD, it is only important for him to be convinced of the fact that the (n, n) Padé approximant can, somehow, be computed rapidly, as its computation is a key step of our Toeplitz algorithms.

The main section, Section 6, deals with Fast Toeplitz Computation and it is divided into five subsections:

(a) Application of EEA to the Solution of Toeplitz Systems of Equation

(b) Solving $Tz = b$, Given x and y, in the Normal Case

(c) The Exceptional Case, $x_0 = 0$

(d) Solving $Tz = b$, Given x and y, without Restrictions

(e) Examples of the Computational Process

In (a) we show that the Toeplitz computation is embedded in the extended Euclidean computation when the latter is applied to the computation of the (n, n) Padé approximant of the polynomial $P(x) = a_0 + a_1 x + \cdots + a_{2n} x^{2n}$. The Toeplitz matrix associated with this polynomial is

$$T = \begin{bmatrix} a_n & & a_0 \\ & \ddots & \\ a_{2n} & & a_n \end{bmatrix}$$

Conversely, for the problem of Toeplitz equations, the given matrix T can clearly be represented by the polynomial $P(x)$. We show that the (n, n) Padé approximant contains the solution to the system $Tx = e_0$, i.e., the first column of T^{-1}. Similarly, the (n, n) Padé approximant associated with T^t (the transpose of T) gives the first row of T^{-1}. We also prove Theorem 5, which states that $\det(T) \neq 0$ if and only if the numerator polynomial of the (n, n) Padé approximant has full degree n. This theorem is important in that it readily determines the solvability of $Tz = b$ as our algorithm computes the (n, n) Padé approximant of P, i.e., U/V. Without damaging the understanding of our solution process, proofs of Lemma 3 and Theorem 5 can be skipped with the knowledge that $\det(T) = 0$ if and only if $\deg(U) < n$. Section (b) describes a new $O(n \log n)$ method for solving $Tz = b$ if one knows x and y, the first column and row of T^{-1}. This is done by making use of a formula originally due to Trench [31]. However, his computations used this formula in a $\Omega(n^2)$ manner. In this regard, it seems that our $O(n \log n)$ method can be used to improve computation with the Trench formula, hence improving Trench's algorithm for solving Toeplitz systems. A similar $O(n \log n)$ method was employed by Chin and Steiglitz [5] for computations with Szegö polynomials. We present Gohberg and Semencul's [12] formulation of Trench's formula, since we found theirs to be most suitable and directly relevant to our algebraic point of view. In (c) we discuss the degenerate case $x_0 = T_{00}^{-1} = 0$, where the Trench formula breaks down. However, a companion formulation exists which we show is always valid when $x_0 = 0$ and $\det(T) \neq 0$. This formula, again attributed to Gohberg and Semencul, is the discrete analog of the Christoffel–Darboux formula [18]. The companion formula also allows one to compute z in time $O(n \log n)$. We end (b) with the $O(n \log^2 n)$ algorithm AD that computes an appropriate x and y to be used with one of these two formulas whenever $\det(T) \neq 0$. A summary of (a), (b), and (c) is given by Theorem 8, which states that the Extended Euclidean Algorithm is always a basis for solving a Toeplitz system $Tz = b$. It is only essential to

understand this theorem and our Trench-type formulations to continue with our Toeplitz equation solver. In (d) we show that the Trench formula and the discrete analog of the Christoffel–Darboux formula can be viewed as one formula. Then we show that the combined formula can be interpreted as polynomial multiplication. Finally we present the $O(n \log n)$ algorithm SOLVE which computes z in four polynomial multiplications of size n. Polynomial multiplication, like n-point FFT and convolution, can clearly be done in $O(n \log n)$. This subsection ends with our algorithms ADT and MDT, each consisting of two steps—(1) call AD or MD, respectively, for finding x and y; and (2) call SOLVE to find the solution z. It is obvious that these are $O(n \log^2 n)$ algorithms. These algorithms for solving Toeplitz equations are demonstrated by three examples in (e). The first example shows the computational steps of algorithm ADT. The second example illustrates how ADT handles the exceptional case $x_0 = 0$. Algorithm MDT is partially carried out in the last example, by illustrating algorithm MD.

Section 7, which deals with special cases for Toeplitz equations, is divided into four subsections:

(a) Iterative Refinement for Toeplitz Systems

(b) Banded Toeplitz Systems

(c) Complexity of the Banded Toeplitz Computation

(d) Theorems for the Banded Case

In subsection (a) we consider iterative refinement for Toeplitz systems. By exploiting the special nature of Toeplitz systems, we indicate that one can actually obtain quadratic convergence of the iterative refinement process. This is an additional efficiency not attainable with the iterative refinement processes applied to general matrices, which usually converge only linearly. We note that the famous Hilbert matrix [8, pp. 80–86] of order $n H_{ij} = 1/(i + j - 1)$, where $1 \leq i,j \leq n$, is known to be extremely ill-conditioned: Hilbert matrices are also Hankel matrices. Thus one might expect numerical difficulties with Toeplitz systems and wish to use iterative refinement. Subsections (b), (c), and (d) detail the banded Toeplitz problem. All results of the full matrix case carry over directly to the banded case. However, our algebraic approach allows us generality by taking full advantage of such sparsities. We demonstrate this power by considering several special-purpose approaches and show that they can be covered and improved by our method.

A short section on the Berlekamp algorithm [4] is included. We define the key equation of BCH decoding and show how the Extended Euclidean Algorithm can be used to solve it. We note that the recent books by McEliece [24] and MacWilliams and Sloane [22] describe the solution to

the key equation in terms of Euclid's algorithm. The key equation that the Berlekamp algorithm solves is a Toeplitz system; the complexity of Berlekamp's algorithm is $O(n^2)$ [14]. We show that the solution to the key equation can be found by applying our algorithm AD or algorithm MD, and hence the complexity of solving the key equation is reduced to $O(n \log^2 n)$.

We conclude the paper with a short summary, and mention some problems for future research.

2. Padé Approximants

Let $A(x) = a_0 + a_1 x + a_2 x^2 + \ldots$ be a power series with $a_0 \neq 0$. A rational function of the form $U(x)/V(x)$ is an (m, n) Padé approximant to $A(x)$ if

$$\deg(U) \leq m, \tag{1}$$

$$\deg(V) \leq n, \tag{2}$$

$$A(x)V(x) - U(x) = O(x^{m+n+1}). \tag{3}$$

This definition is due to Frobenius [9]; see also Padé [26] and Gragg [13]. Equation (3) can be written as a linear system of $m + n + 1$ equations in $m + n + 2$ unknowns (v_i, $i = 0, \ldots, n$, corresponding to $V(x) = \sum_{i=0}^{n} v_i x^i$ and u_i, $i = 0, \ldots, m$, corresponding to $U(x) = \sum_{i=0}^{m} u_i x^i$)

$$\begin{bmatrix} a_0 & & & \\ a_1 & a_0 & & \\ & & & \\ a_m & & a_{m-n} & \\ & \ddots & & \\ & & \ddots & \\ a_{m+n} & & a_m & \end{bmatrix} \begin{bmatrix} v_0 \\ \cdot \\ \cdot \\ v_n \end{bmatrix} = \begin{bmatrix} u_0 \\ \cdot \\ \cdot \\ u_m \\ 0 \\ \cdot \\ 0 \end{bmatrix} \tag{4}$$

where $a_v \equiv 0$ if $v < 0$. System (4) can be rearranged to give the block

triangular system

$$\begin{bmatrix} a_m & a_{m+n-2} & a_{m+n-1} & 0 & & 0 \\ & \ddots & & & & \\ & & & & & \\ a_{m-n+1} & a_{m-1} & a_m & 0 & & 0 \\ a_{m-n} & a_{m-2} & a_{m-1} & -1 & & \\ & \ddots & a_0 & & \ddots & \\ & & 0 & a_0 & & \\ 0 & & & 0 & & -1 \end{bmatrix} \begin{bmatrix} v_n \\ \vdots \\ v_1 \\ u_m \\ \vdots \\ \vdots \\ \vdots \\ u_0 \end{bmatrix} = - \begin{bmatrix} a_{m+n} \\ \vdots \\ a_{m+1} \\ a_m \\ \vdots \\ \vdots \\ \vdots \\ a_0 \end{bmatrix} v_0 \tag{5}$$

The complexity of (5) is associated with solving the first n equations which are an $n \times n$ Toeplitz system. We shall denote this matrix by A_{mn}, formally $A_{mn} = (a_{m+i-j})_{i,j=1}^{n}$, $n \geq 1$. The matrix A_{mn} is of fundamental importance in the study of Padé approximants. Using (5) we can give a simple proof of a theorem due to Frobenius [9].

THEOREM 1. *There always exist solutions to* (1), (2), *and* (3), *or equivalently, to the system of equations* (5). *Furthermore, the ratio* U/V *is unique.*

Proof. If $\det(A_{mn}) \neq 0$ then with $v_0 = 1$ we may solve (5) to find a unique solution. Suppose $\det(A_{mn}) = 0$. By setting $v_0 = 0$ if necessary we see that (5) always has solutions. Any solution of (5) with $V(x) = O(x^\lambda)$ has $U(x) = O(x^\lambda)$ because of the triangular nature of equations governing U; thus we can assume that common factors are removed from U and V, and that $V(0) = 1$. With this assumption U and V are unique (see Gragg [13, Theorem 1] for details). □

Another theorem, due to Padé [26] and extended by Gragg [13], is:

THEOREM 2. *Let* P/Q *be a Padé solution of* (1), (2), *and* (3), $\deg(P) = m$ *and* $\deg(Q) = n$, *and let the power series* $A(x)Q(x) - P(x)$ *begin exactly with the power* $x^{m+n+l+1}$. *Then the following statements are true.*

(a) $l \geq 0$.

(b) *The* (μ, ν) *Padé approximant equals* P/Q *if and only if*

$$m \leq \mu \leq m + l \quad \text{and} \quad n \leq \nu \leq n + l. \tag{6}$$

For (μ, ν) *satisfying* (6):

(c) U/V *is a* (μ, ν) *Padé approximant for* $A(x)$ *if and only if*

$$U(x) = x^{\lambda\mu\nu} D(x) P(x), \qquad V(x) = x^{\lambda\mu\nu} D(x) Q(x)$$

with $\lambda_{\mu\upsilon} = max\{0, (\mu - m) + (\upsilon - n) - l\}$ *and* $D \neq 0$ *of degree at most*

$$\kappa_{\mu\upsilon} = \begin{cases} \lfloor l/2 \rfloor - \max\{|\mu - m - \lfloor l/2 \rfloor|, |\upsilon - n - \lfloor l/2 \rfloor|\} & \text{if } l < \infty \\ \min\{\mu - m, \nu - n\} & \text{if } l = +\infty. \end{cases}$$

(d) $\upsilon - \operatorname{rank}(A_{\mu\upsilon}) = \kappa_{\mu\upsilon}$.

(e) $\det(A_{\mu n}) \neq 0$, $m \leq \mu \leq m + l$,
$\det(A_{m\upsilon}) \neq 0$, $n \leq \upsilon \leq n + l$,
$\det(A_{\mu\upsilon}) = 0$, $m < \mu \leq m + l$ *and* $n < \upsilon \leq n + l$.

A proof of this theorem can be found in [13]. In fact, we strongly urge interested readers to refer to this excellent survey paper for a standard in both the structure and the nomenclature of Padé tables. This paper also contains an example (pp. 11–15), which clearly demonstrates the features of Padé tables to the extent of establishing a geometric conception. Our approaches and algorithms, AD and MD specifically, indeed refer to diagonals of Padé tables and rely on these concepts.

3. The Extended Euclidean Algorithm

In this section, we present the algebraic and computational properties of the well-known Extended Euclidean Algorithm. The version of the algorithm, called EMGCD (acronym for Extended Middle GCD), is a key routine that besides computing GCDs of polynomials yields important quantities necessary for computing inverses. By characterizing the Padé and Toeplitz problems as inverses of GCD computations we reduce their complexities to the GCD problem, namely, $O(n \log^2 n)$. Thus we digress to present EMGCD as a unifying algebraic concept and approach for inverse-type problems, numerical or otherwise.

Given M and P, let

$$\begin{aligned} U_0 &= M, & U_1 &= P, \\ V_0 &= 0, & V_1 &= 1, \\ W_0 &= 1, & W_1 &= 0, \end{aligned} \tag{7}$$

and define

$$\begin{aligned} U_{i+1} &= U_{i-1} - Q_i U_i, \\ V_{i+1} &= V_{i-1} - Q_i V_i, \\ W_{i+1} &= W_{i-1} - Q_i W_i, \end{aligned} \tag{8}$$

where Q_i results when the division algorithm is applied to U_{i-1} and U_i. The

division algorithm, given A and B, finds Q and R such that $A = BQ + R$ with $\deg(R) < \deg(B)$. The following relations hold for all $i \geq 0$:

$$W_i U_0 + V_i U_1 = U_i, \tag{9}$$

$$\mathrm{GCD}(V_i, W_i) = 1. \tag{10}$$

The series in (8) terminates at $i = k$ when $U_{k+1} \equiv 0$; i.e., when U_k divides U_{k-1}. U_k equals $\mathrm{GCD}(M, P)$ and W_k and V_k are unique polynomials, called "co-multipliers," satisfying $W_k M + V_k P = \mathrm{GCD}(M, P)$, with $\deg(W_k) < \deg(P)$ and $\deg(V_k) < \deg(M)$.

It turns out that the quotient polynomial in (8) can be found by using *only* the higher-order terms of the dividend and divisor polynomials U_{i-1} and U_i. This observation and the divide-and-conquer procedure lead to a fast algorithm for performing the extended Euclidean computation, which we call EMGCD.

$(\overline{U}, \overline{W}, \overline{V}) \leftarrow \mathrm{EMGCD}(U_0, U_1)$

1. **if** $\deg(U_1) < 1/2 \deg(U_0)$ **then** $\left\{(\overline{U}, \overline{W}, \overline{V}) \leftarrow \begin{pmatrix} U_0 & 1 & 0 \\ U_1 & 0 & 1 \end{pmatrix}\right\}$

   else **begin**

2. let $U_0 = B_0 x^m + C_0$,
   where $m = \lceil n/2 \rceil$, $\deg(C_0) < m$,
   and $n = \deg(U_0)$
3. let $U_1 = B_1 x^m + C_1$, where $\deg(C_1) < m$
4. $(\overline{U}_1, \overline{W}_1, \overline{V}_1) \leftarrow \mathrm{EMGCD}(B_0, B_1)$
5. $\begin{pmatrix} D \\ E \end{pmatrix} \leftarrow \overline{U}_1 x^m + (\overline{W}_1, \overline{V}_1)\begin{pmatrix} C_0 \\ C_1 \end{pmatrix}$
6. **if** $\deg(E) < \frac{n}{2}$ **then** $\left\{(\overline{U}, \overline{W}, \overline{V}) \leftarrow \left(\begin{pmatrix} D \\ E \end{pmatrix}, \overline{W}_1, \overline{V}_1\right)\right\}$

   else **begin**

7. $Q \leftarrow$ quotient of D by E
8. $F \leftarrow D - QE$
9. $k \leftarrow 2m - \deg(E)$
10. let $E = G_0 x^k + H_0$, where $\deg(H_0) < k$
11. let $F = G_1 x^k + H_1$, where $\deg(H_1) < k$
12. $(\overline{U}_2, \overline{W}_2, \overline{V}_2) \leftarrow \mathrm{EMGCD}(G_0, G_0)$

13. $$(\bar{U}, \bar{W}, \bar{V}) \leftarrow \left(\bar{U}_2 x^k + (\bar{W}_2, \bar{V}_2)\begin{pmatrix} H_0 \\ H_1 \end{pmatrix}, (\bar{W}_2, \bar{V}_2)\begin{pmatrix} 0 & 1 \\ 1 & -Q \end{pmatrix}(\bar{W}_1, \bar{V}_1)\right)$$

end

end

Our algorithm is an extension and improvement of the HGCD algorithm described in [1]. The HGCD algorithm computes the matrix

$$R = \begin{pmatrix} W_j & V_j \\ W_{j+1} & V_{j+1} \end{pmatrix}$$

consisting of successive terms of the comultiplier polynomials. The index j equals $l(n/2)$, a function we shall define shortly. It follows from Eq. (9) that

$$\begin{pmatrix} U_j \\ U_{j+1} \end{pmatrix} = R\begin{pmatrix} U_0 \\ U_1 \end{pmatrix}.$$

Algorithm EMGCD computes the matrix of polynomials

$$\begin{pmatrix} U_j & W_j & V_j \\ U_{j+1} & W_{j+1} & V_{j+1} \end{pmatrix}.$$

Our extension of the HGCD algorithm is in the computation of U_j, U_{j+1}. Our improvement to the HGCD algorithm results from the fact that EMGCD's computation cost is less than HGCD's computation cost despite the fact that U_j and U_{j+1} are also computed. However, both algorithms run in time $O(n \log^2 n)$. Algorithm HGCD and other related algorithms in [1] are incorrect in that they sometimes do not return the proper two iterates in the PRS (Polynomial Remainder Sequence). This is especially true in the nonnormal case which occurs when some of the quotient polynomials have degree greater than one.

Moenck [25] gave a fast GCD algorithm predating [1] and was able to avoid defining the l function. However, his implicit concept of l is akin to ours and, like ours, differs from [1]. We define l for any remainder sequence as follows: for any nonnegative number $r \leq \deg(U_0)$, $l(r)$ is the unique integer such that $\deg(U_{l(r)}) \geq r$ and $\deg(U_{l(r)+1}) < r$.[1] The proof technique of EMGCD is similar to that of HGCD; specifically, it is given by modifications of Lemmas 8.6 and 8.7 and Theorem 8.17 of [1]. Our modifications to their statements and our proofs for EMGCD will be given in a forthcoming paper by Gustavson and Yun.

[1]Our definition differs from that used in [1] with $\geq r$ and $< r$ replacing $> r$ and $\leq r$.

The complexity of EMGCD is $O(n \log^2 n)$, and the proof of this result is similar to proofs given in [1, 25]. The space $S(n)$ needed by EMGCD satisfies the recurrence relation $S(n) \leq S(\lfloor n/2 \rfloor) + O(n)$, so $S(n) = O(n)$, i.e., EMGCD only requires linear space.

The EMGCD algorithm can be modified to return

$$M_j = \begin{pmatrix} U_j & W_j & V_j \\ U_{j+1} & W_{j+1} & V_{j+1} \end{pmatrix}$$

with $\deg(U_j) \geq r$ and $\deg(U_{j+1}) < r$, where $n/2 \leq r \leq n = \deg(U_0)$. This modification does not increase the time required by algorithm EMGCD because the work to be done decreases as r increases from $n/2$ to n. We have produced a fast routine, called PRSDC1 (Polynomial Remainder Sequence Divide and Conquer 1), which returns the matrix M_j, where $j = l(r)$. Now, if $0 \leq r < n/2$ we can apply PRSDC1 several times with $r = n/2$ until the input polynomials U_i, U_{i+1} satisfy the conditions for PRSDC1. At most $\lceil \log n \rceil$ calls to PRSDC1 will be needed and since each successive call has its problem size halved, the total complexity of all these computations is bounded by $O(n \log^2 n)$. We have also produced algorithm PRSDC, which returns M_j for $j = l(r)$ and $0 \leq r \leq n$ by executing PRSDC1 several times if necessary. We note that PRSDC1 reduces to EMGCD when $r = n/2$. Using PRSDC we can compute any term U_i, V_i, W_i in the extended Euclidean computation with cost bounded by $O(n \log^2 n)$ given U_0 and U_1.

4. The Antidiagonal Computation via the Extended Euclidean Algorithm

Many algorithms, including MD, that compute entries of the Padé table start in the upper left corner of the table and proceed to compute along the main diagonal; i.e., start at the northwest corner and proceed in the southeast direction. In contrast, our algorithm AD starts at the element $(m + n, 0)$ along the west border and proceeds in a northeast direction along the *anti*diagonal defined by the straight line $x + y = m + n$. This unnatural starting place and direction is probably the reason why the connection between the Euclidean computation and Padé approximation has gone virtually unnoticed until recently. Lemmas 1 and 2 and Theorem 3 below will show the intimate connection between Padé approximations along antidiagonal directions and extended Euclidean computations. It turns out that our algorithms ADT and MDT for solving Toeplitz equations proceed by computing the (n, n) Padé approximant.

Let $M = x^{m+n+1}$ and $P = \sum_{v=0}^{m+n} a_v x^v$, which is the truncation of a power series A (i.e., $P \equiv A \bmod M$). Now apply the extended Euclidean computation to M and P.

LEMMA 1. *Each step of the extended Euclidean computation gives rise to a unique entry (in lowest terms) of the Padé table.*

Proof. Let $\mu = \deg(U_i) = m + n + 1 - \Sigma_{\sigma=1}^{i} \deg(Q_\sigma)$ and $v = \deg(V_i) = \Sigma_{\sigma=1}^{i-1} \deg(Q_\sigma)$ so that $\mu + v = m + n + 1 - \deg(Q_i)$. It follows from Eq. (9) that $PV_i - U_i = O(x^{\mu+v+l+1})$, $l \geq 0$. Thus $AV_i - U_i = O(x^{\mu+v+l+1})$ since $A \equiv P \bmod M$ and so U_i/V_i is a (μ, v) Padé approximant. Let (μ, v) be identified with the (m, n) of Eq. (5) and apply Theorem 1 to this pair (μ, v). The unique solution guaranteed in Theorem 1 is obtained from U_i and V_i by removing their GCD. □

LEMMA 2. *The term U_i/V_i of the extended Euclidean computation gives rise to $\deg(Q_i)$ equal entries of the Padé table along the $(m + n)$th antidiagonal.*

Proof. The proof of Lemma 2 reduces to showing how the term U_i/V_i fits the general form of a Padé entry as described in Theorem 2. First we show that $\mathrm{GCD}(U_i, V_i) = x^\lambda$. Suppose that $\mathrm{GCD}(U_i, V_i) = G(x)$. Then $G(x)$ divides $W_i M$. Also, $G(x)$ is relatively prime to W_i, for $\mathrm{GCD}(V_i, W_i) = 1$. So $G(x)$ divides M or $G(x) = x^\lambda$ for some $\lambda \geq 0$. Now we have two cases to consider. Suppose $\lambda > 0$. Then $PV_i - U_i = W_i M$ and $W_i(0) \neq 0$. Hence $AV_i - U_i$ is a power series whose first nonzero term starts with x^{n+m+1}. Now the unique reduced form of U_i/V_i is $U_i' = x^{-\lambda}U_i/(x^{-\lambda}V_i)(0)$, $V_i' = x^{-\lambda}V_i/(x^{-\lambda}V_i)(0)$, where $\deg(U_i) = \mu$ and $\deg(V_i) = v$. Since $\deg(U_i') = \mu - \lambda$, $\deg(V_i') = v - \lambda$, $\mu + v = m + n + 1 - \deg(Q_i)$, and the l as defined in Theorem 2, is equal to $\deg(Q_i) - 1 + \lambda$. Hence a Padé block of size $l + 1$ by $l + 1$ exists with upper left corner at $(\mu - \lambda, v - \lambda)$ (see Fig. A). Furthermore, the $(\mu + v + \deg(Q_i) - 1)$th antidiagonal intersects this block at $\deg(Q_i)$ lattice points.

Now consider the case $\lambda = 0$. The entry U_i/V_i is in lowest terms and the l of Theorem 2 is greater than or equal to $\deg(Q_i) - 1$, as we know from (9) that $PV_i - U_i = O(x^{m+n+1})$ and $W_i(0)$ might equal 0. It follows that $PV_i - U_i$ has an $l \geq \deg(Q_i) - 1$. Thus the (μ, v)th Padé approximant resides in the upper left-hand corner of a block of size $l + 1$ by $l + 1$ and the $(m + n)$th antidiagonal intersects this block at $\deg(Q_i)$ entries (see Fig. B, where α equals $\deg(Q_i)$). By Theorem 2 all entries in any Padé block are equal (to the upper left-hand corner entry). Hence in both cases U_i/V_i gives rise to $\deg(Q_i)$ equal entries of a Padé block. □

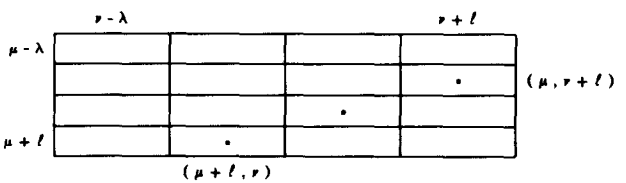

FIGURE A

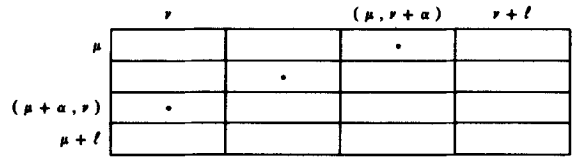

FIGURE B

Now $\Sigma_{i=1}^{k}\deg(Q_i) + m + n + 1$ so the terms U_i/V_i, $i = 1, \ldots, k$, from the extended Euclidean computation produce all entries of the Padé Table along the $(m + n)$th antidiagonal. This remark together with Lemmas 1 and 2 establishes the following:

THEOREM 3. *All entries along the $(m + n)$th antidiagonal of the Padé table for $A(x)$ are computed uniquely by the Extended Euclidean Algorithm. By using algorithm PRSDC we can compute any entry in time $O((n + m)\log^2(m + n))$.*

We suggest that the reader work out Gragg's example [13] $A(x) = 1 + x + x^2 + 2x^3 + 3x^4 + \ldots$ in the Euclidean context, by computing all the Padé approximants along the sixth antidiagonal; i.e., (6, 0), (5, 1), ..., (1, 5), (0, 6) Padé table entries. This is done by applying the Extended Euclidean Algorithm to $U_0 = x^7$ and $U_1 = 1 + x + x^2 + 2x^3 + 3x^4 + 4x^5 + 5x^6$. Note that this computation is partly carried out in Section 6d. However, by doing the entire computation, the reader will see all the structure and generality of this section.

5. THE MAIN DIAGONAL ALGORITHM

It is convenient to define some notation. The (m, n) Padé approximant to a power series A is denoted by $\mathcal{P}_{m,n}(A)$. (By convention, $\mathcal{P}_{-1,0}(0) = 0$ and $\mathcal{P}_{0,-1}(\infty) = \infty$, where the polynomial ∞ is the inverse of the zero polynomial.) $\operatorname{ord}(A)$ is the index of the first nonzero coefficient of A, i.e., $\operatorname{ord}(A) = \min\{i : a_i \neq 0\}$. (By convention $\operatorname{ord}(0) = \infty$.) If A is a polynomial (or a power series with only a finite number of nonzero terms), $\deg(A) = \max\{i : a_i \neq 0\}$ denotes the degree of A. (By convention $\deg(0) = -\infty$.)

A power series A is *normal* if all its Padé approximants are distinct, i.e., $\mathcal{P}_{i,j}(A) = \mathcal{P}_{m,n}(A)$ if and only if $i = m$ and $j = n$. Several necessary and sufficient conditions for normality are given in Gragg [13]. (Some follow easily from Theorem 2.)

For computational purposes we are interested in finite segments of power series or Padé tables. If $A(x) = a_0 + a_1x + \ldots$, then $A(x) \bmod x^n$

denotes the polynomial $a_0 + a_1x + \cdots + a_{n-1}x^{n-1}$. We say that a power series or polynomial A is (M, N)-normal if its Padé approximants $\mathcal{P}_{i,j}(A)$ are distinct for $0 \le i \le M$ and $0 \le j \le N$. (This differs from Gragg's definition.)

We now present an algorithm MD which computes entries along the main diagonal of the Padé table for a normal power series A. The algorithm is based on the following theorem.

THEOREM 4. *Let $C(x)$ and $D(x)$ be power series with* $\operatorname{ord}(C) = \operatorname{ord}(D) = 0$. *Let k, m, n, s be integers, $m \ge 0$, $n > 0$, $k \ge s$, $s = 0$ or 1. Let U_i, V_i $(i = 1, \ldots, 4)$ be polynomials such that* $\operatorname{ord}(U_i) = \operatorname{ord}(V_i) = 0$ $(i = 1, 2, 3)$,

$$
\begin{aligned}
U_1/V_1 &= \mathcal{P}_{m,n-1}(C/D),\\
U_2/V_2 &= \mathcal{P}_{m,n}(C/D),\\
E &= (U_1D - V_1C)/x^{m+n},\\
F &= (U_2D - V_2C)/x^{m+n+1},\\
U_3/V_3 &= \mathcal{P}_{k,k-s}(E/F),\\
U_4 &= U_2U_3 - xU_1V_3,\\
V_4 &= V_2U_3 - xV_1V_3.
\end{aligned}
$$

Finally, if C/D is $(k + m + 1 - s, \max(k, 1) + n)$-normal, then $U_4/V_4 = \mathcal{P}_{k+m+1-s,k+n}(C/D)$ and $\operatorname{ord}(U_4) = \operatorname{ord}(V_4) = 0$.

Proof. Since $U_1/V_1 = \mathcal{P}_{m,n-1}(C/D)$ and $U_2/V_2 = \mathcal{P}_{m,n}(C/D)$, E and F are power series in x. By the normality condition, $\operatorname{ord}(E) = \operatorname{ord}(F) = 0$. We also have

$$
\begin{aligned}
U_4D - V_4C &= (U_2D - V_2C)U_3 - x(U_1D - V_1C)V_3\\
&= x^{m+n+1}(FU_3 - EV_3) = O(x^{2k+m+n+2-s}),
\end{aligned}
$$

but $\deg(U_4) \le \max(k + m, k + m + 1 - s) = k + m + 1 - s$ and $\deg(V_4) \le \max(k + n, k + n - s) = k + n$. Finally, U_4 and V_4 are not both zero, for $U_4V_1 - V_4U_1 = (U_2V_1 - U_1V_2)U_3 \ne 0$ by the normality condition. Thus the result follows. □

Algorithms MD and MD2

The "main diagonal" algorithm MD returns the (n, n) Padé approximant to an (n, n)-normal power series (or polynomial of degree $2n$) A, when given the parameters n and A. It simply invokes another recursive algorithm MD2 and returns the desired outputs:

$$(\bar{U}, \bar{V}) \leftarrow \mathrm{MD}(n, A) : (U, V, \bar{U}, \bar{V}) \leftarrow \mathrm{MD2}(n, 0, A, 1).$$

The recursive algorithm MD2(N, S, C, D) returns $(U_4, V_4, \overline{U}_4, \overline{V}_4)$ such that $U_4/V_4 = \mathcal{P}_{N+S-1,N}(C/D)$ and $\overline{U}_4/\overline{V}_4 = \mathcal{P}_{N+S,N}(C/D)$, where $N \geq 0, S = 0$ or 1, C and D are power series with $\operatorname{ord}(C) = \operatorname{ord}(D) = 0$, and C/D is $(N + S, N)$-normal:

$(U_4, V_4, \overline{U}_4, \overline{V}_4) \leftarrow \text{MD2}(N, S, C, D)$

1. **if** $N + S \leq 0$ **then** $(U_4, V_4, \overline{U}_4, \overline{V}_4) \leftarrow (0, 1, c_0, d_0)$ **else**
2. $\{k \leftarrow \lfloor N/2 \rfloor, n \leftarrow N - k, m \leftarrow n + S - 1$
3. $(V_1, U_1, V_2, U_2) \leftarrow$
   $\text{MD2}(m, 1 - S, D \bmod x^{N+S+1}, C \bmod x^{N+S+1})$
4. $(E, F) \leftarrow ((U_1 D - V_1 C)/x^{m+n} \bmod x^{N+1},$
   $(U_2 D - V_2 C)/x^{m+n+1} \bmod x^{N+1})$
5. $(V_3, U_3, \overline{V}_3, \overline{U}_3) \leftarrow \text{MD2}(k, 0, F, E)$
6. $(U_4, V_4, \overline{U}_4, \overline{V}_4) \leftarrow \{(U_2 U_3 - x U_1 V_3) \bmod x^{N+S},$
   $(V_2 U_3 - x V_1 V_3) \bmod x^{N+1},$
   $(U_2 \overline{U}_3 - x U_1 \overline{V}_3) \bmod x^{N+S+1},$
   $(V_2 \overline{U}_3 - x V_1 \overline{V}_3) \bmod x^{N+1})\}$

Correctness of Algorithm MD2

The proof of correctness of algorithm MD2 is by induction on $I = 2N + S$. Clearly the algorithm is correct if $I = 0$, for then $N = S = 0$, $0 = \mathcal{P}_{-1,0}(C/D)$, and $c_0/d_0 = \mathcal{P}_{0,0}(C/D)$. Hence, suppose that $I > 0$ and that the algorithm is correct if $2N + S < I$. The inductive hypothesis is applicable to the recursive calls of MD2, so

$$U_1/V_1 = \mathcal{P}_{m,n-1}(C/D),$$
$$U_2/V_2 = \mathcal{P}_{m,n}(C/D),$$
$$U_3/V_3 = \mathcal{P}_{k,k-1}(E/F),$$
$$\overline{U}_3/\overline{V}_3 = \mathcal{P}_{k,k}(E/F),$$

where k, m, and n are as at step 2 of the algorithm. Thus, by Theorem 4 with $s = 1$, $U_4/V_4 = \mathcal{P}_{k+m,k+n}(C/D) = \mathcal{P}_{N+S-1,N}(C/D)$. Also, by Theorem 4 with $s = 0$, $\overline{U}_4/\overline{V}_4 = \mathcal{P}_{k+m+1,k+n}(C/D) = \mathcal{P}_{N+S,N}(C/D)$. Thus, the result follows by induction on I.

Complexity of Algorithms MD and MD2

Let $T(N)$ be the time required by algorithm MD2$(N, \ldots)$. From the recursive definition of MD2 it is clear that $T(N) \leq T(\lceil N/2 \rceil)$

$+ T(\lfloor N/2 \rfloor) + O(N \log N)$, where the term $O(N \log N)$ accounts for the computation in steps 4 and 6 (assuming the FFT is used for polynomial multiplication). Thus $T(N) = O(N \log^2 N)$. Clearly the time required by algorithm MD(n, A) is $T(n) = O(n \log^2 n)$, the same as that required by the antidiagonal algorithm.

The space $S(N)$ required by MD2$(N, \ldots)$ satisfies the recurrence relation $S(N) \leq S(\lceil N/2 \rceil) + O(N)$, so $S(N) = O(N)$. Thus, MD(n, A) requires space $O(n)$, which is the same as that required by the antidiagonal algorithm.

Algorithms MD2 and MD were originally derived by a recursive computation of the continued fraction for a normal power series, using a method similar to that of Schönhage [29]. Theorem 4 followed from the relationship between truncated continued fractions and Padé approximants. We prefer to present a direct proof of Theorem 4, avoiding the use of continued fractions, because the direct proof is simpler and generalizes more easily to power series over a noncommutative field. This generalization has applications to the solution of block Toeplitz systems.

The normality assumption of Theorem 4 may be weakened, but to remove it entirely requires substantial complications in the algorithm and proof. Since we describe the antidiagonal algorithm in full generality (without any normality assumption), we prefer to restrict our description of the main-diagonal algorithm to the (n, n)-normal case.

6. Fast Solution of Toeplitz Equations

(a) *Application of EEA to the Solution of Toeplitz Systems of Equations*

Recall the extended GCD computation defined earlier; i.e., $U_0 = x^{2n+1}$ and $U_1 = a_0 + a_1 x + \cdots + a_{2n} x^{2n}$. Let (u, v, w) denote the polynomials (U_j, V_j, W_j), where $j = l(n+1) + 1$. Thus $\deg(U_{j-1}) \geq n + 1$ and $\deg(U_j) < n + 1$ and (u, v, w) is a solution to the system

$$\begin{bmatrix} a_0 & & & \\ \cdot & \cdot & & \\ \cdot & & \cdot & \\ a_n & & & a_0 \\ \cdot & \cdot & & \cdot \\ \cdot & & \cdot & \cdot \\ a_{2n} & & & a_n \\ & \cdot & & \cdot \\ & & \cdot & \cdot \\ & & & a_{2n} \end{bmatrix} \begin{bmatrix} v_0 \\ \cdot \\ \cdot \\ v_n \end{bmatrix} = \begin{bmatrix} u_0 \\ \cdot \\ \cdot \\ u_n \\ 0 \\ \cdot \\ 0 \\ -w_0 \\ \cdot \\ -w_{n-1} \end{bmatrix} \tag{11}$$

Equation (11) is a statement of Eq. (9) in its component form. The middle $n + 1$ equations in (11) are

$$\begin{bmatrix} a_n & & a_0 \\ & \ddots & \\ a_{2n} & & a_n \end{bmatrix} \begin{bmatrix} v_0 \\ \vdots \\ v_n \end{bmatrix} = \begin{bmatrix} u_n \\ 0 \\ \vdots \\ 0 \end{bmatrix}, \tag{12}$$

which corresponds precisely to the Toeplitz matrix, T, of interest. In fact, it is already clear that the vector, v, corresponding to the polynomial V_j, solves the Toeplitz system of equations $Tv = u_n e_0$.

The polynomials U_j and V_j also compute the (n, n) Padé approximant, as the first $2n + 1$ equations in (11) are the same equations as (4) with $m = n$. The following lemma and theorem state when the matrix T is nonsingular. We first give some nomenclature to aid the understanding of their proofs. Recall that a Padé table for a power series is an infinite two-dimensional array of rational functions, which we called R_{mn}. A Padé block is a square subarray of the Padé table consisting of $(l + 1)^2$ entries of equivalent rational functions (see Fig. A and B of Section 4). The perimeter of a Padé block has four borders, which we label north, east, south, and west. Also recall doubly indexed matrices of the form $A_{mn} = (a_{m+i-j})_{i,j=1}^{n}$, $n \geq 1$, corresponding to any infinite power series. These matrices, having fundamental importance to the study of the Padé tables, are clearly square Toeplitz matrices.

LEMMA 3. $\det T \neq 0$ *if and only if the* (n, n) *Padé approximant lies either on the north or east border of its Padé block.*

Proof. First, note that $T = A_{n,n+1}$. Let (i,j) be any entry of a Padé block. Theorem 2, part (e), says that $\det(A_{ij}) \neq 0$ if and only if (i,j) belongs to either the north or west border of its Padé block. If the (n,n) element of the Padé table does not lie on either the north or east border then the $(n, n + 1)$ element will also lie in the Padé block but *not* on the north or west border; hence $\det(T) = 0$. If the (n, n) element resides on the east border, then the $(n, n + 1)$ element resides on the west border of an adjacent block; hence $\det(T) \neq 0$. If the (n, n) element resides only on the north border, then the $(n, n + 1)$ element also resides on the north border; hence $\det T \neq 0$. □

We now use Lemma 3 to prove the following Theorem 5.

THEOREM 5. $\det(T) = 0$ *if and only if* $u_n = \mathrm{lc}(U_j) = 0$.

Proof. By Lemma 3 it suffices to show that $u_n = 0$ if and only if the (n, n) element of the Padé table lies either on the north or east border of its Padé block. The EMGCD algorithm computes $(U_{j-1}, W_{j-1}, V_{j-1})$ and

(U_j, W_j, V_j), where $j = l(n+1)+1$. Hence $\deg(U_{j-1} \geq n+1$ and $\deg(U_j) < n+1$. It follows from Lemma 2 that (U_j, V_j) computes $\deg(Q_j)$ equal entries of a Padé block with $n - \deg(Q_j) < \deg(U_j) \leq n$. We use the same notation and distinguish the same two cases as those in Lemma 2. Thus $\mu + v = 2n + 1 - \deg(Q_j)$, $\mu = \deg(U_j)$, and $v = \deg(V_j)$. Suppose $\mu = n - i$, $0 \leq i < \deg(Q_j)$. Then $v = n + i - \deg(Q_j) + 1$, and $u_n = 0$ if and only if $i = 0$.

Case 1. $\lambda = 0$. We know $l \geq \deg(Q_j) - 1$. Also, (μ, v) defines the northwest corner of the $l+1$ by $l+1$ Padé block (see Fig. B of Section 4). Hence $(n,n) = (\mu + i, v - i + \deg(Q_j) - 1)$ and the (n,n) element lies on the north border if and only if $i \neq 0$.

Case 2. $\lambda > 0$. We know $l = \deg(Q_j) - 1 + \lambda$ and that (μ', v') define the northwest element of the $l+1$ by $l+1$ Padé block (see Fig. A of Section 4). It follows that $(n,n) = (\mu' + \lambda + i, v' + l - i)$, and the (n,n) element lies on the east border if and only if $i \neq 0$. □

Theorem 5 states a necessary and sufficient condition on the singularity of T. Thus, by using algorithm EMGCD, we can, in $O(n \log^2 n)$ operations, either find the first column of T^{-1} or state that T is singular.

(b) *Solving $Tz = b$, Given x and y, in the Normal Case*

We now describe a new $O(n \log n)$ method for solving $Tz = b$ if one knows x and y, the first column and row of T^{-1}. This is done by making use of a formula originally due to Trench [31]. However, both Trench's algorithm and Zohar's analysis [35] used the formula in a $\Omega(n^2)$ manner. We present Gohberg and Semencul's [10, 12] formulation of Trench's formula (also noted by Kailath *et al.* [18]), since we found theirs to be most suitable and directly relevant to our algebraic point of view.

THEOREM 6 (Gohberg and Semencul). *If the Toeplitz matrix T is such that each of the systems of equations $Tx = e_0$, $Ty^r = e_n$ is solvable where $y^r = (y_n, \ldots, y_0)$, and the condition $x_0 = y_0 \neq 0$ is fulfilled, then the matrix T is invertible, and its inverse S is formed according to the formula*

$$S = \frac{1}{x_0}\left\{\begin{bmatrix} x_0 & 0 & \cdot & 0 \\ x_1 & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & 0 \\ x_n & \cdot & x_1 & x_0 \end{bmatrix}\begin{bmatrix} y_0 & y_1 & \cdot & y_n \\ 0 & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & y_1 \\ 0 & \cdot & 0 & y_0 \end{bmatrix}\right.$$

$$\left. - \begin{bmatrix} 0 & \cdot & \cdot & 0 \\ y_n & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot \\ y_1 & \cdot & y_n & 0 \end{bmatrix}\begin{bmatrix} 0 & x_n & \cdot & x_1 \\ \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & x_n \\ 0 & \cdot & \cdot & 0 \end{bmatrix}\right\}. \qquad (13)$$

The importance of Theorem 6 is that the inverse S is expressed as a linear combination of products of Toeplitz matrices. Hence to solve $Tz = b$ we can form $z = Sb$ by formula (13) and do each of four matrix vector multiplications by the FFT. This follows since the matrices in Eq. (13) are all Toeplitz. the cost of computing Sb using the FFT by Eq. (13) given x and y is $O(n \log n)$.

Now if we consider the transpose of T or $U_1 = a_0 x^{2n} + \cdots + a_{2n}$ we can find the first row of T^{-1}, also in $O(n \log^2 n)$. Hence, by applying algorithm EMGCD twice and by using Eq. (13), we have a means of solving $Tz = b$ in $O(n \log^2 n)$. We observe that (13) is not valid when $x_0 \neq 0$. However, Trench's algorithm requires a normality condition that all principal minors of T be nonzero. This condition implies $x_0 \neq 0$, since x_0 is the ratio of two principal minors. Our next objective will be to show that we can remove such a normality condition for our Toeplitz algorithm ADT.

(c) *The Exceptional Case,* $x_0 = 0$

When $x_0 = T_{00}^{-1} = 0$ the convolutional formula (13) no longer applies. The simple example

$$T = \begin{bmatrix} 0 & 0 & a \\ c & 0 & 0 \\ d & c & 0 \end{bmatrix},$$

where

$$T^{-1} = \begin{bmatrix} 0 & c^{-1} & 0 \\ 0 & -dc^{-2} & c^{-1} \\ a^{-1} & 0 & 0 \end{bmatrix},$$

shows that T^{-1} cannot be expressed in terms of its first row and column. Hence to solve $Tz = b$ when $x_0 = 0$ we must proceed differently. Now Gohberg and Semencul state another theorem, which gives a convolutional -type formula for the inverse of an $n \times n$ Toeplitz matrix in terms of the first row and column of the inverse of a bordered $(n + 1) \times (n + 1)$ Toeplitz matrix. Kailath *et al.* [18] show formula (14) to be the discrete analog of the Christoffel–Darboux formula.

THEOREM 7 (Gohberg and Semencul). *If solutions* $\tilde{x} = (\tilde{x}_0, \ldots, \tilde{x}_{n+1})$ *and* $\tilde{y}^T = (\tilde{y}_{n+1}, \ldots, \tilde{y}_0)$ *of the system* $\tilde{T}\tilde{x} = e_0$ *and* $\tilde{T}\tilde{y}^T = e_{n+1}$ *exist, where*

$$\tilde{T} = \begin{bmatrix} a_n & & & a_0 & a_{-1} \\ & \ddots & & & \\ & & \ddots & & \\ a_{2n} & & & a_n & \\ a_{2n+1} & & & & a_n \end{bmatrix},$$

and the condition $\tilde{x}_0 = \tilde{y}_0 \neq 0$ *is fulfilled, then the matrix* T *is invertible, and its inverse is*

$$S = \frac{1}{\tilde{x}_0}\left\{\begin{bmatrix} \tilde{x}_0 & 0 & \cdot & 0 \\ \tilde{x}_1 & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & 0 \\ \tilde{x}_n & \cdot & \tilde{x}_1 & \tilde{x}_0 \end{bmatrix}\begin{bmatrix} \tilde{y}_0 & \tilde{y}_1 & \cdot & \tilde{y}_n \\ 0 & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \tilde{y}_1 \\ 0 & \cdot & 0 & \tilde{y}_0 \end{bmatrix}\right.$$

$$\left. - \begin{bmatrix} \tilde{y}_{n+1} & 0 & \cdot & 0 \\ \tilde{y}_n & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & 0 \\ \tilde{y}_1 & \cdot & \tilde{y}_n & \tilde{y}_{n+1} \end{bmatrix}\begin{bmatrix} \tilde{x}_{n+1} & \tilde{x}_n & \cdot & \tilde{x}_1 \\ 0 & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \tilde{x}_n \\ 0 & \cdot & 0 & \tilde{x}_{n+1} \end{bmatrix}\right\}. \tag{14}$$

We shall use the matrix $\tilde{T}_\beta$ (i.e., set $a_{2n+1} = \beta$) as a representative of matrix $\tilde{T}$. Since $u_n \neq 0$ we know that $\tilde{x}_0 = \tilde{T}_{00}^{-1} = \det(T)/\det(\tilde{T}_\beta)$ is nonzero. We now show that we always can choose β so that $\det(\tilde{T}_\beta) \neq 0$ and so Theorem 7 above applies.

Let

$$\tilde{T}_\beta = \begin{pmatrix} T & c \\ r^t & a_n \end{pmatrix} = \begin{pmatrix} T & 0 \\ r^t & f \end{pmatrix}\begin{pmatrix} I & s \\ 0 & 1 \end{pmatrix},$$

where $r^t = (\beta, a_{2n}, \ldots, a_{n+1})$, $c^t = (0, a_0, \ldots, a_{n-1})$, $f(\beta) = a_n - r^t s$, and $s = T^{-1}c$. This block factorization is permissible since $\det(T) \neq 0$ and it follows that $\det(\tilde{T}_\beta) = f \det(T)$. Let $r^t = r_1^t + r_2^t$ where $r_1^t = \beta e_0^t$ and $r_2^t = (0, a_{2n}, \ldots, a_{n+1})$. Substitute r and s into the expression for f and get $f(\beta) = a_n - r_2^t T^{-1}c - \beta(e_0^t T^{-1}c)$. We now prove

LEMMA 4. *If* $y_0 = x_0$ *and* $\det(T) \neq 0$ *then* $e_0^t T^{-1}c \neq 0$.

Proof. $y^t = e_0^t T^{-1}$ is the first row of T^{-1}. The matrix T^{-1} is persymmetric; hence $Ty^r = e_n$. Now suppose $e_0^t T^{-1}c = 0$. We shall show $\det(T) = 0$. The relations $Ty = e_n$, $y_0 = 0$, and $e_0^t T^{-1}c = 0$ can be combined to give the equations

$$\begin{bmatrix} a_{n-1} & \cdot & \cdot & a_0 \\ a_n & \cdot & \cdot & a_1 \\ \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot \\ a_{2n-1} & \cdot & \cdot & a_n \end{bmatrix}\begin{bmatrix} y_n \\ \cdot \\ \cdot \\ y_1 \end{bmatrix} = 0$$

and $y \neq 0$. This equation says that a nonzero combination of the last n columns of T is equal to the zero vector; and hence $\det(T) = 0$. □

Lemma 4 establishes that $f(\beta)$ is indeed a linear function in β and hence vanishes for exactly one value β_0. This fact allows us to establish an $O(n$

$\log^2 n$) algorithm to solve $Tz = b$ when $x_0 = T_{00}^{-1} = 0$, as follows. Suppose $x_0 = 0$. Then with $U_0 = x^{2n+3}$, $U_1 = \beta x^{2n+2} + \cdots + a_{-1}$, $\beta = 1$, $a_{-1} = 1$, use algorithm EMGCD to find U_{j-1} and U_j so that $\deg(U_{j-1}) \geq n + 2$, $\deg(U_j) < n + 2$. Suppose $\deg(U_j) = n + 1$. We know, by Theorem 5, that $\det(\tilde{T}_\beta) \neq 0$, and we can apply Theorem 7. If $\deg(U_j) < n + 1$ then, by Lemma 4, $\beta = \beta_0 = 1$ is the *only* value for which $\det(\tilde{T}_\beta) = 0$. Hence, change the value of β in U_1 to equal -1 and again use algorithm EMGCD to compute new U_{j-1} and U_j so that $\deg(U_{j-1}) \geq n + 2$, $\deg(U_j) < n + 2$. Now, by Lemma 4, $\deg(U_j) = n + 1$; i.e., we must have $\det(\tilde{T}_\beta) \neq 0$. Therefore we can apply Theorem 7. We can now state using Theorem 6 and formula (13) or Theorem 7 and formula (14) the following

THEOREM 8. *Let the upper right and lower left elements of a Toeplitz matrix T be nonzero. Then the extended Euclidean algorithm can be used as the basis for solving the associated Toeplitz system $Tz = b$.*

We summarize the last three sections with the $O(n \log^2 n)$ algorithm AD to compute vectors x and y with $x_0 = y_0 \neq 0$. It follows that either Theorem 6 or Theorem 7 can be applied, and so $Tz = b$ can be obtained from x and y by (13) or (14) in time $O(n \log n)$. Note that when Theorem 6 holds we append 0 to x and y (see the next subsection for an explanation). Algorithm AD has input parameters n and $a = (a_0, a_1, \ldots, a_{2n})$, which fully describes the Toeplitz matrix T. One output parameter is the Boolean variable s and $s = 1$ if and only if $\det(T) \neq 0$. The other output parameters are the vectors $x = (x_0, \ldots, x_n, x_{n+1})^t$ and $y = (y_0, \ldots, y_n, y_{n+1})^t$ and these have meaning only if $s = 1$.

AD$(n, a : x, y, s)$

1. $U_0 \leftarrow t^{2n+1}$, $U_1 \leftarrow a_0 + a_1 t + \cdots + a_{2n} t^{2n}$, $s \leftarrow 1$
2. $\begin{pmatrix} U_j & W_j & V_j \\ U_{j+1} & W_{j+1} & V_{j+1} \end{pmatrix} \leftarrow$ EMGCD(U_0, U_1)
3. **If** $(\deg(U_{j+1}) < n)$ **then** $\{s \leftarrow 0$ **comment** "T is singular"$\}$
4. **else** $\{$**If** $V_{j+1}(0) \neq 0$ **then**
5. $\{x \leftarrow (\text{lc}(U_{j+1})^{-1} V_{j+1}, 0)$
6. $U_0 \leftarrow t^{2n+1}$, $U_1 \leftarrow a_{2n} + a_{2n-1} t + \cdots + a_0 t^{2n}$,
7. $\begin{pmatrix} U_j & W_j & V_j \\ U_{j+1} & W_{j+1} & V_{j+1} \end{pmatrix} \leftarrow$ EMGCD(U_0, U_1)
8. $y \leftarrow (\text{lc}(U_{j+1})^{-1} V_{j+1}, 0)\}$
9. **else** $\{U_0 \leftarrow t^{2n+3}$,
$U_1 \leftarrow 1 + a_0 t + \cdots + a_{2n} t^{2n+1} + t^{2n+2}$

10. $\begin{pmatrix} U_j & W_j & V_j \\ U_{j+1} & W_{j+1} & V_{j+1} \end{pmatrix} \leftarrow \text{EMGCD}(U_0, U_1)$

11. **If** $(\deg(U_{j+1}) = n + 1)$ **then**

12. $\{x \leftarrow \text{lc}(U_{j+1})^{-1} V_{j+1}$

13. $U_0 \leftarrow t^{2n+3}$,
$U_1 \leftarrow 1 + a_{2n}t + \cdots + a_0 t^{2n+1} + t^{2n+2}$

14. $\begin{pmatrix} U_j & W_j & V_j \\ U_{j+1} & W_{j+1} & V_{j+1} \end{pmatrix} \leftarrow \text{EMGCD}(U_0, U_1)$

15. $y \leftarrow \text{lc}(U_{j+1})^{-1} V_{j+1}\}$

16. **else** $\{U_0 \leftarrow t^{2n+3}$,
$U_1 \leftarrow 1 + a_0 t + \cdots + a_{2n} t^{2n+1} - t^{2n+2}$

17. $\begin{pmatrix} U_j & W_j & V_j \\ U_{j+1} & W_{j+1} & V_{j+1} \end{pmatrix} \leftarrow \text{EMGCD}(U_0, U_1)$

18. $x \leftarrow \text{lc}(U_{j+1})^{-1} V_{j+1}$

19. $U_0 \leftarrow t^{2n+3}$,
$U_1 \leftarrow -1 + a_{2n}t + \cdots + a_0 t^{2n+1} + t^{2n+2}$

20. $\begin{pmatrix} U_j & W_j & V_j \\ U_{j+1} & W_{j+1} & V_{j+1} \end{pmatrix} \leftarrow \text{EMGCD}(U_0, U_1)$

21. $y \leftarrow \text{lc}(U_{j+1})^{-1} V_{j+1}\}\}\}$

Statements 4 to 8 compute x and y in the normal case. Statements 9 through 21 handle the exceptional case. The cases $\beta_0 = 1$ and -1 are covered by statements 9 through 15 and statements 16 through 21, respectively.

(d) *Solving $Tz = b$, Given x and y, without Restrictions*

We develop in this section a polynomial $Z(t) = \sum_{i=0}^{n} z_i t^i$ whose coefficients $z = (z_0, \ldots, z_n)$, representing the vector z, are the solution to $Tz = b$. We show that the cost of finding $Z(t)$ can be accomplished by an algorithm, which we call SOLVE, in four multiplications and one addition of polynomials of degree n. The inputs to algorithm SOLVE are the coefficients of the polynomials $X(t) = \sum_{i=0}^{n+1} x_i t^i$, $Y(t) = \sum_{i=0}^{n+1} y_i t^i$, and $B(t) = \sum_{i=0}^{n} b_i t^i$. The coefficients of $Y(t)$ and $X(t)$ represent the first row and column of T^{-1} or $\tilde{T}^{-1}$ and the coefficients of $B(t)$ represent the right-hand-side vector b.

Theorems 6 and 8 gave two formulas, (13) and (14), that expressed $S = T^{-1}$ in terms of the first row and column of the matrices T^{-1} and

$\tilde{T}^{-1}$, respectively. Suppose we set $\tilde{x}_{n+1} = \tilde{y}_{n+1} = 0$ in (14); then formula (14) is the same as formula (13). This means that formula (14) is a general expression for S if we agree to extend x and y in (13) by appending $x_{n+1} = y_{n+1} = 0$ to them.

The use of formula (14) as T^{-1} to compute the solution vector $z = T^{-1}b$ involves four multiplications of Toeplitz matrices and vectors. We emphasize the importance of the matrices being Toeplitz for the following reasons. First, these multiplications can be, and often are, viewed as vector convolutions. Additionally, we can consider both types of operations as polynomial multiplications and, thereby, present the solution process from a consistent algebraic point of view. The close relationship of vector convolution and polynomial multiplication is probably most evident. However, it is the Toeplitz structure of these matrices that allows the problem to be viewed in several ways and to be susceptible to solution methods from all of these disciplines.

Now we show that formula (14) can be viewed as polynomial multiplication and addition. We need to introduce the concept of the reverse polynomial. Let $P(t) = \sum_{i=0}^{n} p_i t^i$, whose formal degree is n. The reverse polynomial, $P^r(t)$, associated with $P(t)$, is defined to be the polynomial $\sum_{i=0}^{n} p_{n-i} t^i$. Consider the computation

$$\begin{bmatrix} c_0 \\ \cdot \\ \cdot \\ c_n \end{bmatrix} = \begin{bmatrix} y_0 & y_1 & \cdot & y_n \\ 0 & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & y_1 \\ 0 & \cdot & 0 & y_0 \end{bmatrix} \begin{bmatrix} b_0 \\ \cdot \\ \cdot \\ b_n \end{bmatrix}.$$

Let $Q(t) = (Y(t)B^r(t)) \bmod t^{n+1}$ where $Q(t) = \sum_{i=0}^{n} q_i t^i$. An elementary computation shows that the coefficients of $Q^r(t)$ correspond precisely to the desired c_i's. Similar computations can be done for the other three matrix–vector multiplications. We now present algorithm SOLVE, which will establish our claim that $Z(t)$ can be obtained in four polynomial multiplications and one addition.

SOLVE$(x, y, b : z)$

1. $X(t) \leftarrow x_0 + x_1 t + \cdots + x_{n+1} t^{n+1}$
   $Y(t) \leftarrow y_0 + y_1 t + \cdots + y_{n+1} t^{n+1}$
   $B(t) \leftarrow b_0 + b_1 t + \cdots + b_n t^n$
2. $P(t) \leftarrow (X^r(t)B^r(t)) \bmod t^{n+1}$
   $Q(t) \leftarrow (Y(t)B^r(t)) \bmod t^{n+1}$
3. $Z(t) \leftarrow (1/x_0)(X(t)Q^r(t) - Y^r(t)P^r(t)) \bmod t^{n+1}$
   $z \leftarrow (z_0, z_1, \ldots, z_n)$ of $Z(t)$

We complete this section with some remarks on the complexity of algorithm SOLVE. The five reverse operations and one polynomial addi-

tion in steps 2 and 3 have a complexity $O(n)$. The four polynomial multiplications in steps 2 and 3 can be done in $O(n \log n)$ if the FFT or convolution method is used, or in $O(n^2)$ if the ordinary inner product method is used.

We can now state our algorithms, ADT and MDT, for solving Toeplitz systems of equations. Basically, these are two-step algorithms: 1. Find the first column, x, and row, y, of T^{-1} by calling algorithms AD and MD, respectively. 2. Apply our SOLVE algorithm for finding z in terms of x, y, and b.

Algorithm ADT$(n, a, b : s, z)$	Algorithm MDT$(n, a, b : z)$
1. **call** AD$(n, a : x, y, s)$	1.a $(U, V) \leftarrow$ MD(n, a); $x \leftarrow (V/\text{lc}(U), 0)$
	1.b $(U, V) \leftarrow$ MD(n, a^r); $y \leftarrow (V/\text{lc}(U), 0)$
2. **if** s **then call** SOLVE$(x, y, b : z)$	2. **call** SOLVE$(x, y, b : z)$

The computational cost for algorithm ADT or MDT is clearly dominated by that of step 1, i.e., $O(n \log^2 n)$. The second step costs only $O(n \log n)$, which gives the necessary advantage for the enhancement by iterative refinement (described in Section 7a) without increasing the complexity.

(e) *Examples of the Computational Process*

The examples of this section were computed using floating-point arithmetic. However, for clarity we have converted the floating-point numbers to their exact rational values. Also, t will be used as the indeterminate of all polynomials because x will denote the first column of T^{-1}.

Let

$$T = \begin{bmatrix} 2 & 1 & 1 & 1 \\ 3 & 2 & 1 & 1 \\ 4 & 3 & 2 & 1 \\ 5 & 4 & 3 & 2 \end{bmatrix} \quad \text{and} \quad b = \begin{bmatrix} 5 \\ 8 \\ 10 \\ 14 \end{bmatrix}.$$

T is an $(n + 1) \times (n + 1)$ Toeplitz matrix with $n = 3$. Thus the input polynomials to EMGCD are $U_0 = t^7$ and $U_1 = 5t^6 + 4t^5 + 3t^4 + 2t^3 + t^2 + t + 1$. The output of EMGCD is the matrix M_j, where $j = l(4) = 2$. The elements of M_j are the polynomials

$$U_2 = (4 - t - t^2 + 3t^3 + 2t^4 + t^5)/25,$$
$$U_3 = 25(1 - t - t^3),$$
$$V_2 = (4 - 5t)/25,$$
$$V_3 = 25(1 - 2t + t^2),$$
$$W_2 = 1,$$
$$W_3 = 25(6 - 5t).$$

Now $u_n = \mathrm{lc}(U_{j+1}) = 25$, so, by Theorem 5, $\det(T) \neq 0$. Also, $x = u_n^{-1}v = (1, -2, 1, 0)$. To find y, we form the transpose polynomial $U_1^T = t^6 + t^5 + t^4 + 2t^3 + 3t^2 + 4t + 5$ and apply EMGCD to U_0 and $U_1 = U_1^T$. For this U_0 and U_1 we shall trace the execution flow of EMGCD.

Since $\deg(U_1) = 6 > 7/2 = n/2$ we execute the first **else** clause. At lines 2 and 3 we split U_0 and U_1 as follows:

$$m = 4,$$
$$B_0 = t^3,$$
$$B_1 = t^2 + t + 1,$$
$$C_0 = 0,$$
$$C_1 = 2t^3 + 3t^2 + 4t + 5.$$

We now suspend execution of EMGCD to recursively compute $(\overline{U}_1, \overline{W}_1, \overline{V}_1) = \mathrm{EMGCD}(U_0, U_1)$ where $U_0 = B_0$ and $U_1 = B_1$.

Since $\deg(U_1) = 2 > 3/2 = n/2$ we execute the first **else** clause and split the new U_0 and U_1:

$$m = 2,$$
$$B_0 = t,$$
$$B_1 = 1,$$
$$C_0 = 0,$$
$$C_1 = t + 1.$$

Again we suspend execution of EMGCD to compute $(\overline{U}_1, \overline{W}_1, \overline{V}_1) = \mathrm{EMGCD}(U_0, U_1)$, where $U_0 = B_0$ and $U_1 = B_1$.

Now $\deg(U_1) = 0 \leq 1/2 = n/2$ so EMGCD returns at line 1

$$(\overline{U}_1, \overline{W}_1, \overline{V}_1) = \begin{pmatrix} t & 1 & 0 \\ 1 & 0 & 1 \end{pmatrix},$$

which is the result of line 4 of the last suspended execution of EMGCD. At line 5 we compute

$$\begin{pmatrix} D \\ E \end{pmatrix} = \begin{pmatrix} t \\ 1 \end{pmatrix} t^m + \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \begin{pmatrix} C_0 \\ C_1 \end{pmatrix} = \begin{pmatrix} t^3 \\ t^2 + t + 1 \end{pmatrix}.$$

Since, at line 6, $\deg(E) = 2 \geq 3/2$, the **else** clause is executed:

$$Q = t - 1,$$
$$F = 1,$$
$$k = 2,$$
$$G_0 = 1,$$
$$G_1 = 0,$$
$$H_0 = t + 1,$$
$$H_1 = 1.$$

We suspend execution at line 12 to compute $(\bar{U}_2, \bar{W}_2, \bar{V}_2) = \text{EMGCD}(G_0, G_1)$. The result

$$\begin{pmatrix} G_0 & 1 & 0 \\ G_1 & 0 & 1 \end{pmatrix}$$

is returned as the condition at line 1 is satisfied. The two computations at line 13 yield

$$\begin{pmatrix} t^2 + t + 1 & 0 & 1 \\ 1 & 1 & 1 - t \end{pmatrix},$$

which is the result of line 4 of the first call to EMGCD. At line 5 we compute

$$\begin{aligned} \begin{pmatrix} D \\ E \end{pmatrix} &= \begin{pmatrix} t^2 + t + 1 \\ 1 \end{pmatrix} t^4 + \begin{pmatrix} 0 & 1 \\ 1 & 1 - t \end{pmatrix} \begin{pmatrix} C_0 \\ C_1 \end{pmatrix} \\ &= \begin{pmatrix} t^6 + t^5 + t^4 + 2t^3 + 3t^2 + 4t + 5 \\ -t^4 - t^3 - t^2 - t + 5 \end{pmatrix} \end{aligned}$$

and at line 6 the condition is false. Thus we compute for lines 7 to 11 the results

$$\begin{aligned} Q &= -t^2, \\ F &= t^3 + 8t^2 + 4t + 5, \\ k &= 4, \\ G_0 &= -1, \\ G_1 &= 0, \\ H_0 &= -t^3 - t^2 - t + 5, \\ H_1 &= F, \end{aligned}$$

and at line 12 we suspend execution to compute EMGCD (G_0, G_1). The result

$$\begin{pmatrix} G_0 & 1 & 0 \\ G_1 & 0 & 1 \end{pmatrix}$$

is returned as the condition of line 1 is true. On return at line 13 we compute

$$\bar{U} = \begin{pmatrix} -1 \\ 0 \end{pmatrix} t^4 + \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \begin{pmatrix} H_0 \\ H_1 \end{pmatrix} = \begin{pmatrix} E \\ F \end{pmatrix}$$

and

$$(\bar{W}, \bar{V}) = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}\begin{pmatrix} 0 & 1 \\ 1 & t^2 \end{pmatrix}\begin{pmatrix} 0 & 1 \\ 1 & 1-t \end{pmatrix} = \begin{pmatrix} 1 & 1-t \\ t^2 & 1+t^2-t^3 \end{pmatrix},$$

which is the final result. The value $j = l(4) = 2$. We can compute j by inserting the statement $j \leftarrow j + 1$ after line 8. The variable j is a global variable and should be initialized to zero before calling EMGCD. Thus EMGCD computes

$$\begin{aligned} U_2 &= 5 - t - t^2 - t^3 - t^4, \\ U_3 &= 5 + 4t + 8t^2 + t^3, \\ V_2 &= 1 - t, \\ V_3 &= 1 + t^2 - t^3, \\ W_2 &= 1, \\ W_3 &= t^2. \end{aligned}$$

Now $y = u_n^{-1}v = (1, 0, 1, -1)$ and we are ready to call SOLVE. First we must append 0 to x and y as they are to be considered polynomials of formal degree $n + 1$. At Step 1 of SOLVE we compute

$$\begin{aligned} X(t) &= 1 - 2t + t^2 + 0t^3 + 0t^4, \\ Y(t) &= 1 + 0t + t^2 - t^3 + 0t, \\ B(t) &= 5 + 7t + 10t^2 + 14t^3. \end{aligned}$$

At Step 2 we form the polynomials

$$\begin{aligned} B^r(t) &= 14 + 10t^2 + 7t + 5t^3, \\ X^r(t) \bmod t^4 &= t^2 - 2t^3 \end{aligned}$$

and compute

$$\begin{aligned} P(t) &= 14t^2 - 18t^3, \\ Q(t) &= 14 + 10t + 21t^2 + t^3. \end{aligned}$$

At Step 3 we form the polynomials

$$\begin{aligned} Y^r(t) \bmod t^4 &= -t + t^2, \\ P^r(t) &= -18 + 14t, \\ Q^r(t) &= 1 + 21t + 10t^2 + 14t^3, \end{aligned}$$

and compute the products

$$X(t)Q^r(t) \bmod t^4 = 1 + 19t - 31t^2 + 15t^3,$$
$$Y^r(t)P^r(t) \bmod t^4 = 18t - 32t^2 + 14t^3.$$

Now $Z(t) = 1 + t + t^2 + t^3$ and $z = (1, 1, 1, 1)$ is the solution to $Tz = b$.

The second example illustrates the exceptional case. We will apply the algorithm following Theorem 8.

Let

$$T = \begin{bmatrix} 0 & 0 & 2 \\ 1 & 0 & 0 \\ 4 & 1 & 0 \end{bmatrix} \quad \text{and} \quad b = \begin{bmatrix} 2 \\ 1 \\ 5 \end{bmatrix}.$$

Then at line 1, $n = 2$ and $U_0 = t^5$, $U_1 = 2 + t^3 + 4t^4$. At line 2, apply EMGCD to U_0 and U_1 and it returns $j = l(3) = 2$ and the matrix M_j, whose elements are

$$U_2 = 1/16(2 - 8t + t^3),$$
$$U_3 = 32t^2,$$
$$V_2 = 1/16(1 - 4t),$$
$$V_3 = 16t^2,$$
$$W_2 = 1,$$
$$W_3 = -16(1 + 4t).$$

Since $\mathrm{lc}(U_2) = 1/16 \neq 0$ we know by Theorem 5 that $\det(T) \neq 0$. Hence the condition at line 3 is false and we execute the **else** clause at line 4. However, $V_3(0) = 0$; hence $x_0 = 0$. The condition at line 4 is false and we execute the **else** clause at line 9. Thus we consider

$$\tilde{T} = \begin{bmatrix} 0 & 0 & 2 & 1 \\ 1 & 0 & 0 & 2 \\ 4 & 1 & 0 & 0 \\ 1 & 4 & 1 & 0 \end{bmatrix}$$

by setting $U_0 = t^7$ and $U_1 = 1 + 2t + t^4 + 4t^5 + t^6$. We again apply EMGCD at line 10. It returns $j = l(4) = 3$ and the matrix M_j:

$$U_3 = (1 - 2t + 7t^2 + 30t^3 + t^4)/225,$$
$$U_4 = 225(2 - 4t + 14t^2 + 59t^3),$$
$$V_3 = (1 - 4t + 15t^2)/225,$$
$$V_4 = 225(2 - 8t + 30t^2 - t^3),$$
$$W_3 = -(56 + 15t)/225,$$
$$W_4 = 225(-111 - 26t + t^2).$$

Since $\mathrm{lc}(U_4) = 13275 \neq 0$ we know $\det(\tilde{T}) \neq 0$. This means the condition at line 11 is true and we execute the **then** clause. Also, as must be the case, $V_4(0) = 450 \neq 0$ so $x_0 \neq 0$. Now, at line 12, $x = u_n^{-1}v = 1/59(2, -8, 30, -1)$ is the first column of $\tilde{T}^{-1}$. To find the first row of $\tilde{T}^{-1}$ we set, at line 13, $U_0 = t^7$ and $U_1 = 1 + 4t + t^2 + 2t^5 + t^6$. At line 14 we again apply EMGCD. It returns $j = l(4) = 3$ and M_j:

$$U_3 = (t^2 + 4t^3 + t^4)/4,$$
$$U_4 = 2 + 7t + 14t^2 + 59t^3,$$
$$V_3 = t^2/4,$$
$$V_4 = 2 - t + 16t^2 - 4t^3,$$
$$W_3 = -(2 + t)/4,$$
$$W_4 = -31 - 8t + 4t^2.$$

Now at line 15 we compute $y = u_n^{-1}v = 1/59(2, -1, 16, -4)$ and we exit from the algorithm. We now apply algorithm SOLVE. At Step 1 we form the polynomials

$$X(t) = (2 - 8t + 30t^2 - t^3)/59,$$
$$Y(t) = (2 - t + 16t^2 - 4t^3)/59,$$
$$B(t) = 2 + t + 5t^2.$$

At Step 2 we form the polynomials

$$B^{\mathrm{r}}(t) = 5 + t + 2t^2,$$
$$X^{\mathrm{r}}(t) \bmod t^3 = (-1 + 30t - 8t^2)/59$$

and compute the polynomials

$$P(t) = (-5 + 149t - 12t^2)/59,$$
$$Q(t) = (10 - 3t + 83t^2)/59.$$

At Step 3 we form the polynomials

$$Y^{\mathrm{r}}(t) \bmod t^3 = (-4 + 16t - t^2)/59,$$
$$P^{\mathrm{r}}(t) = (-12 + 149t - 5t^2)/59,$$
$$Q^{\mathrm{r}}(t) = (83 - 3t + 10t^2)/59$$

and compute the products

$$X(t)\,Q^{\mathrm{r}}(t) \bmod t^3 = (166 - 670t + 2534t^2)/3481,$$
$$Y^{\mathrm{r}}(t)P^{\mathrm{r}}(t) \bmod t^3 = (48 - 788t + 2416t^2)/3481.$$

Now $1/x_0$ times the difference of these two polynomials gives $Z(t) = 1 + t + t^2$ and $z = (1, 1, 1)$ is the solution to $Tz = b$.

The final example illustrates Algorithm MD. Let $n = 2$, $A(t) = 60 + 30t + 20t^2 + 15t^3 + 12t^4$. At line 1 of MD we suspend execution to compute MD2(2, 0, A, 1). Thus, when execution of MD2 commences we have $N = 2$, $S = 0$, $C = 60 + 30t + 20t^2 + 15t^3 + 12t^4$, $D = 1$. The condition $N + S \leq 0$ at line 1 of MD2 is not satisfied, so we execute line 2: $k \leftarrow 1$; $n \leftarrow 1$; $m \leftarrow 0$. At line 3 we set $(V_1, U_1, V_2, U_2) \leftarrow$ MD2(0, 1, 1, $60 + 30t + 20t^2$). This sets $V_1 \leftarrow 1$; $U_1 \leftarrow 60$; $V_2 \leftarrow 30(2 - t)$; $U_2 \leftarrow 3600$ (we omit details of the recursion). At line 4 we set $E \leftarrow -5(6 + 4t + 3t^2)$ and $F \leftarrow -30(10 + 10t + 9t^2)$. Then, at line 5, we set $(V_3, U_3, \bar{V}_3, \bar{U}_3) \leftarrow$ MD2(1, 0, F, E). This sets $V_3 \leftarrow 90000$; $U_3 \leftarrow 3000(3 - t)$; $\bar{V}_3 \leftarrow 270000000(-1 + 2t)$; $\bar{U}_3 \leftarrow 9000000(-3 + 16t)$ (we again omit details of the recursion: it is easily verified that $V_3/U_3 = \mathcal{P}_{0,1}(F/E)$ and $\bar{V}_3/\bar{U}_3 = \mathcal{P}_{1,1}(F/E)$). Finally we set $U_4 \leftarrow 16200000(2 - t)$; $V_4 \leftarrow 90000(6 - 6t - t^2)$; $\bar{U}_4 \leftarrow -3240000000(30 - 21t + t^2)$; $\bar{V}_4 \leftarrow -162000000(10 - 12t + 3t^2)$. Thus, MD returns the (2, 2) Padé approximant $U/V = \bar{U}_4/\bar{V}_4 = 20(30 - 21t + t^2)/(10 - 12t + 3t^2)$.

The reader who works through these or similar examples will quickly appreciate two points:

(a) It is best not to apply EMGCD and MD2 recursively when N is small, but to use a more straightforward algorithm for small N. The optimal value of N above which recursion should be used depends on the implementation of the algorithm.

(b) If rational arithmetic is used, the coefficients can grow rather rapidly, and our assumption that each arithmetic operation takes unit time is dubious. If floating-point arithmetic is used, scaling is desirable to avoid underflow and overflow.

7. Special Cases of Toeplitz Equations

(a) *Iterative Refinement for Toeplitz Systems*

If our algorithms are implemented using floating-point arithmetic, rounding errors will usually cause the computed solution $\hat{z}$ of $Tz = b$ to be inaccurate, and it may be worthwhile to use the technique of iterative refinement [8] to improve the accuracy of the solution.

To perform one step of iterative refinement we first compute the residual $r = T\hat{z} - b$ accurately. This may be done in time $O(n \log n)$ using the FFT, since T is Toeplitz. We then solve $Tc = r$ in time $O(n \log n)$ using algorithm SOLVE, and set $z \leftarrow \hat{z} - c$. The process may be repeated if necessary until the desired accuracy is obtained for z. For details of stopping criteria and conditions under which the process converges, see [8].

Since each iteration takes time $O(n \log n)$ we can afford to perform $O(\log n)$ iterations without increasing the overall complexity $O(n \log^2 n)$ of our Toeplitz Algorithms ADT or MDT by more than a constant factor.

We now show that the iterative refinement process can be speeded up from linear to quadratic convergence when the underlying matrix T is Toeplitz. The key lies in using the Trench–Christoffel–Darboux formula as a representation of the inverse S of T. The idea is to improve the systems $Tx = e_0$ and $Ty^r = e_n$ by computing x and y more accurately, by refinement. This simultaneously improves the accuracy of S as S is a function of x and y. Suppose x and y are initially correct to d digits. Then one step of iterative refinement computes x and y to 2d digits. Now use x and y in the formulas for S, (13) and (14), and improve again. This gives new x and y correct to 4d digits $\cdots$; hence we get quadratic convergence.

To make these ideas more precise we state algorithm Toeplitz REFINE. Toeplitz REFINE takes vectors $\hat{x}$ and $\hat{y}$ as inputs and gives improved vectors x and y as outputs.

Algorithm Toeplitz REFINE($\hat{x},\hat{y} : x,y$)

1. *Compute*	$r_1 = e_0 - T\hat{x}$	
	$r_2 = e_n - T\hat{y}^r$	
2. *Solve*	$T\Delta x = r_1$	via algorithm SOLVE
	$T\Delta y^r = r^2$	
3. *Compute*	$x = \hat{x} + \Delta x$	
	$y = \hat{y} + \Delta y$	

Finally we mention another approach: Use the formula for S in an implicit equation and then apply Newton's methods as a means of correcting x and y. Because of quadratic convergence we can accelerate the refinement process. This approach will be applicable to our Toeplitz solution process if each iteration step in Newton's method is bounded by $O(n \log n)$.

(b) *Banded Toeplitz Systems*

Let T_{bc} be an $(n+1) \times (n+1)$ banded Toeplitz matrix with row and column bandwidths equal to b and c; i.e.,

$$T_{bc} = \begin{bmatrix} a_n & a_{n-1} & \cdot & \cdot & a_{n-b} & 0 & \cdot & \cdot & 0 \\ a_{n+1} & a_n & & & & & & & \cdot \\ \cdot & & & & & & & & \cdot \\ \cdot & & & & & & & & 0 \\ a_{n+c} & & & & & & & & a_{n-b} \\ 0 & \cdot & & & & & & & \cdot \\ \cdot & & & & & & & & \cdot \\ \cdot & & & & & & & a_n & a_{n-1} \\ 0 & \cdot & \cdot & 0 & a_{n+c} & \cdot & \cdot & a_{n+1} & a_n \end{bmatrix}$$

Define $w = b + c$, $U_0 = M = x^{n+b+1}$ and $U_1 = P = \sum_{j=0}^{w} \alpha_j x^j$ where $\alpha_j = a_{n-b+j}$, $j = 0, \ldots, w$. Apply the Extended Euclidean Algorithm to U_0 and U_1 to compute U_i, V_i, W_i, $i = 1, \ldots, k$, and recall that this computation produces all Padé approximants along the $(b + n)$th antidiagonal for the power series $A(x) = P(x) + M(x)B$, where $B(x)$ is arbitrary. In particular, note that the (b, n) Padé approximant U/V satisfies $U = VP + WM$ where $U = \sum_{j=0}^{b} u_j x^j$ and $V = \sum_{j=0}^{n} v_j x^j$. As in Theorem 5 we have that $\det(T_{bc}) = 0$ if and only if $\deg(U) = b$ and so $x = u_b^{-1} v = T_{bc}^{-1} e_0$.

(c) *Complexity of the Banded Toeplitz Computation*

We shall demonstrate that the (b, n) Padé approximant to P can be computed in time $O(n \log n) + O(w \log^2 w)$. We break up the computation into three parts. First we compute

$$\begin{pmatrix} U_1 \\ U_2 \end{pmatrix} = \begin{pmatrix} 0 & 1 \\ 1 & -Q_1 \end{pmatrix} \begin{pmatrix} U_0 \\ U_1 \end{pmatrix}.$$

Second, we apply the Extended Euclidean Algorithm to $\tilde{U}_0 = U_1$ and $\tilde{U}_1 = U_2$, with degrees of $\tilde{U}_0$ and $\tilde{U}_1 \leq w$, and we compute $\tilde{U}_j (= U_{j+1})$, $\tilde{V}_j$, and $\tilde{W}_j$, where $\deg(U_j) > b$ and $\deg(U_{j+1}) \leq b$. Third, we compute $V_{j+1} = \tilde{W}_j - Q_1 \tilde{V}_j$. The first computation computes Q_1 and U_2 via the division algorithm and if fast Fourier methods are used this computation can be done in time $O(n \log n)$. (See [20].) Note that $\deg(Q_1) = n - c + 1$ and $\deg(U_2) \leq w - 1$. The third computation consists mainly of multiplying Q_1 by $\tilde{V}_j$, where $\deg(\tilde{V}_j) \leq c - 1$. This computation is bounded by $O(n \log n)$ if we see FFT to do the polynomial multiplication. Hence U_{j+1}/V_{j+1}, the (b, n) Padé approximant to P, can be computed in $O(n \log n)$ plus the cost of the second computation. It is evident that the second problem is equivalent to computing $\tilde{U}_i, \tilde{V}_i, \tilde{W}_i$, an arbitrary term in the extended Euclidean computation. We now indicate how the second problem can be solved in $O(w \log^2 w)$. Using PRSDC described in Section 4, we can compute any term $\tilde{U}_i, \tilde{V}_i, \tilde{W}_i$ in the extended Euclidean computation in work bounded by $O(w \log^2 w)$ given $\tilde{U}_0$ and $\tilde{U}_1$. In particular, we can compute $\tilde{U}_j, \tilde{V}_j, \tilde{W}_j$, the solution to the second problem.

The main diagonal algorithm MD can be modified in a similar manner to solve banded problems in time $O(n \log n) + O(w \log^2 w)$, provided certain minors of T are nonzero. We omit the details.

Our bound $O(n \log n) + O(w \log^2 w)$ improves Jain's bound $O(n \log n) + O(w^2)$ given in [16], and is better than the bound $O(nw^2)$ obtained by Gaussian elimination unless $w^2 = O(\log n)$.

(d) *Theorems for the Banded Case*

It turns out that Theorems 5, 6, and 7 and Lemmas 3 and 4 are true in the banded case. We state the results but omit proofs.

LEMMA 3B. $\det(T_{bc}) \neq 0$ *if and only if the* (b, n) *Padé approximant lies on the north or east border of its Padé block.*

THEOREM 5B. $\det(T_{bc}) = 0$ *if and only if* $u_b = \mathrm{lc}(U_j) = 0$.

Theorems 6, 7 and Lemma 4 make no assumptions on the elements of T; hence they hold for banded matrices.

THEOREM 8B. *The Extended Euclidean Algorithm can be used as the basis for solving any Toeplitz system* $Tz = b$.

Theorem 8B is a more general result than Theorem 8. Essentially, it says we can remove the restrictions $a_0 \neq 0$ and $a_{2n} \neq 0$, which means the polynomial U_1 can be arbitrary.

8. Applications to Shift Register Synthesis and BCH Decoding

Let $S(x) = s_1 x + \cdots + s_{2n} x^{2n}$ be a given syndrome polynomial. The key equation to finding the error location polynomial of BCH decoding is

$$(1 + S(x))\sigma(x) \equiv \omega(x) \bmod x^{2n+1},$$

where $\sigma(x) = 1 + \sum_{i=1}^{e} \sigma_i x^i$ and $\omega(x) = 1 + \sum_{i=1}^{e} \omega_i x^i$ and $e = \deg(\sigma) = \deg(\omega)$ is small. Berlekamp's algorithm [4] is an $O(n^2)$ method [14] for computing $\sigma(x)$ and $\omega(x)$. Another application of algorithm PRSDC solves this problem. Let $U_0(x) = x^{2n+1}$ and $U_1(x) = 1 + S(x)$. Then the iterate (U_j, V_j, W_j) of the Extended Euclidean Algorithm which computes the (n, n) Padé approximant to U_1 is the solution to the key equation. The main-diagonal algorithm MD could also be used if a normality condition is satisfied. By either method the complexity of the problem is reduced to $O(n \log^2 n)$.

9. Summary and Conclusion

We have presented two new fast techniques (algorithms AD and MD) for computation of Padé approximants and the solution of Toeplitz sys-

tems of equations in time $O(n \log^2 n)$ and space $O(n)$. Algorithm MD is a realization of fast computation based on the continued fraction approach.

The other direction of this paper is that the Extended Euclidean Algorithm can be used to compute Padé approximants and solve Toeplitz systems of equations. The advantage of this approach over most previous ones for solving Toeplitz systems is that their degeneracies are automatically handled. We presented a new fast algorithm called EMGCD to compute the middle iterate of the Extended Euclidean Algorithm. Algorithm PRSDC, a generalization to EMGCD, computes any iterate of the Extended Euclidean Algorithm. Both of these algorithms require time $O(n \log^2 n)$ and space $O(n)$. Finally, combining these results, we presented algorithm AD for computing Padé approximants and for solving Toeplitz systems of equations. A major conclusion to be drawn from this work, is that the Extended Euclidean Algorithm, continued fractions, the Padé tables, and Toeplitz systems are intimately related. Furthermore, the Extended Euclidean Algorithm appears to be the natural computational tool for computing Padé table entries and solving Toeplitz systems of equations.

In this paper we have not considered the question of numerical stability of the various algorithms presented. We are currently investigating this question and shall report on it later. We merely mention here that various forms of "pivoting" are possible and desirable.

The algorithms for solution of Toeplitz systems of equations can be generalized to deal with block Toeplitz systems. To do this, we need to consider power series and polynomials with matrix rather than scalar coefficients, hence in a non-commutative coefficient domain. This will be described in a separate paper.

ACKNOWLEDGMENTS

David Yun would like to acknowledge a fruitful discussion with Barry Trager during his summer visit at IBM Research in 1977 when the antidiagonal approach to Padé approximation was germinating. Fred Gustavson would like to acknowledge a stimulating discussion with Dr. Gerald Goertzel about implementing the antidiagonal algorithms in his structured APL language. It gave him the ability to quickly implement complicated structured recursive algorithms and its use was a key factor in the implementation of the antidiagonal algorithms in this paper. Gustavson would like to acknowledge a discussion with Dr. Victor Pan in which the idea to use Newton's method in iterative refinement arose. Richard Brent acknowledges the kind hospitality of the EECS Department, University of California, Berkeley, and the Department of Computer Science, Carnegie–Mellon University, during the development of the main-diagonal algorithm, and subsequent discussions with Professor B. Anderson, Professor E. Hannan, Dr. N. Strand, and Professor J. Traub. Both Gustavson and Yun would like to thank Drs. R. Brayton and J. Cooley for carefully reading parts of the manuscript and making valuable comments. The referee and Professor H. Wilf made a constructive suggestion on reorganizing the material so that the results on Toeplitz equations can be read in a self-contained manner. The referee also pointed out the result by Dickinson.

REFERENCES

1. A. V. AHO, J. E. HOPCROFT, AND J. D. ULLMAN, "The Design and Analysis of Computer Algorithms," Addison–Wesley, Reading, Mass., 1974.
2. H. AKAIKE, Block Toeplitz matrix inversion, *SIAM J. Appl. Math.* **24** (1973), 234–241.
3. E. H. BAREISS, Numerical solution of linear equations with Toeplitz and vector Toeplitz matrices, *Numer. Math.* **13** (1969), 404–424.
4. E. R. BERLEKAMP, "Algebraic Coding Theory," Chap. 7, McGraw–Hill, New York, 1968.
5. F. CHIN AND K. STEIGLITZ, Discrete Szegö polynomials and an $O(N \log N)$ algorithm for error evaluation, *in* "Proceedings, 12th Annual Allerton Conf. on Circ. and Syst. Theory, October 1975," pp. 120–129.
6. B. W. DICKINSON, Efficient solution of linear equations with banded Toeplitz matrices, *IEEE Trans. Acoust., Speech, Sig. Proc.* **ASSP-27**, No. 4 (1979), 421–423.
7. J. DURBIN, The fitting of time-series models, *Rev. Inst. Internat. Statist.* **28** (1959), 229–249.
8. G. E. FORSYTHE AND C. B. MOLER, "Computer Solution of Linear Algebraic Systems," Prentice–Hall, Englewood Cliffs, N.J., 1967.
9. G. FROBENIUS, Über Relationem zwischen den Naherungsbruchen von Potenzreihen, *J. fur Math.* **90** (1881), 1–17.
10. I. C. GOHBERG AND I. A. FELDMAN, "Convolution Equations and Projection Methods for Their Solutions," Translations of Math. Monographs, Vol. 41, Amer. Math. Soc., Providence, R.I., 1974.
11. I. C. GOHBERG AND G. HAJNIK, Inversion of finite Toeplitz with entries being elements from a noncommutative algebra, *Rev. Roumaine Math. Pures Appl.* **19** (1974), 623–663. [In Russian]
12. I. C. GOHBERG AND A. A. SEMENCUL, On the inversion of finite Toeplitz matrices and their continuous analogs, *Mat. Issled.* **2** (1972), 201–233. [In Russian]
13. W. B. GRAGG, The Padé table and its relation to certain algorithms of numerical analysis, *SIAM Rev.* **14**, No. 1 (1972), 1–62.
14. F. G. GUSTAVSON, Analysis of the Berlekamp–Massey linear feedback shift-register synthesis algorithm, *IBM J. Res. Develop.* **20**, No. 3 (1976), 204–212.
15. F. G. GUSTAVSON AND D. Y. Y. YUN, Fast algorithms for rational Hermite approximation and solution of Toeplitz systems, *IEEE Trans. Circuits and Systems* **CAS-26**, No. 9 (1979), 750–755.
16. A. K. JAIN, Fast inversion of banded Toeplitz matrices by circular decompositions, *IEEE Trans. Acoust., Speech, Sig. Proc.* **ASSP-26**, No. 2 (1978), 121–126; also appeared as Tech. Report AJ-76-001, Dept. of EE, SUNY, Buffalo, N.Y. 14260, Jan. 1976.
17. M. MORF AND T. KAILATH, Recent results in least-square estimation theory, *Ann. Econ. Social Meas.* **6**, No. 3 (1977), 261–274.
18. T. KAILATH, A. VIERA, AND M. MORF, Inverses of Toeplitz operators, innovations, and orthogonal polynomials, *SIAM Rev.* **20**, No. 1 (1978), 106–119.
19. L. KRONECKER, Zur Theorie der Elimination einer Variabelen aus zwei algebraischen Gleichungen, *Monatsb. Konigl. Preuss. Akad. Wiss. Berlin* (1881), 735–600.
20. H. T. KUNG, On computing reciprocals of power series, *Numer. Math.* **22** (1974), 341–348.
21. N. LEVINSON, The Wiener RMS (root-mean-square) error criterion in filter design and prediction, *J. Math. Phys.* **25** (1947), 261–278.
22. F. J. MACWILLIAMS AND N. J. SLOANE, "The Theory of Error Correcting Codes," Parts 1, 2, North-Holland, New York, 1977.
23. R. J. MCELIECE AND J. B. SHEARER, A property of Euclid's algorithm and an application to Padé approximation, *SIAM J. Appl. Math.* **34**, No. 4 (1978), 611–617.
24. R. J. MCELIECE, "The Theory of Information and Coding," Encyclopedia of Mathematics and Its Applications, Vol. 2, Addison–Wesley, Reading, Mass, 1977.

25. R. MOENCK, Fast computation of GCD's, *in* "Proceedings, Fifth Annual Symposium on theory of Computing, 1973," pp. 142–171.
26. H. PADÉ, (1892) "Sur la representation approchée d'une fonction par des fractions rationelles," Thesis, Ann. Ecole Nor. (3), 9, pp. 1–93, suppl.
27. J. RISSANEN, Algorithms for triangular decomposition of block Hankel and Toeplitz matrices with application to factoring positive matrix polynomials, *Math. Comp.* **27** (1973), 147–154.
28. J. RISSANEN, Solution of linear equations with Hankel and Toeplitz matrices, *Numer. Math.* **22** (1974), 361–366.
29. A. SCHÖNHAGE, Schnelle Berechnung von Kettenbruchentwicklungen, *Acta Informatica* **1** (1971), 139–144.
30. Y. SUGIYAMA, M. KASAHARA, S. HIRASAWA, AND T. NAMEKAWA, A method for solving key equations for decoding Goppa codes, *Inform. Contr.* **27** (1977), 87–99.
31. W. F. TRENCH, (Sept. 1964) An algorithm for the inversion of finite Toeplitz matrices, *J. SIAM* **12**, No. 3 (1964), 715–522.
32. D. D. WARNER, Kronecker's algorithm for Hermite interpolation with an application to sphere drag in a fluid-filled tube, *in* "Proceedings, Workshop on Padé Approximation, CNRS Marseilles, 1976" (D. Bessis, J. Gliewicz, and P. Mery, Eds.), pp. 48–71.
33. G. A. WATSON, An algorithm for the inversion of block matrices of Toeplitz form, *J. Assoc. Comput. Mach.* **20** (1973), 409–415.
34. D. Y. Y. YUN, Uniform bounds for a class of algebraic mappings, *SIAM J. Comp.* **8**, No. 3 (1979), 348–356.
35. S. ZOHAR, Toeplitz matrix inversion: The algorithm of W. F. Trench, *J. Assoc. Comput. Mach.* **16**, No. 4 (1969), 792–601.