1980.

[T] M. Tismenetsky, "Determinant of block-Toeplitz band matrices", *Lin. Alg. Appl.* **85** (1987) 165-184.

[Tr1] W. F. Trench, "Characteristic polynomials of symmetric rationally generated Toeplitz matrices", *Linear and multilinear Algebra* **21** (1987) 289-296.

[Tr2] W. F. Trench, "On the eigenvalue problem for Toeplitz band matrices", *Lin. Alg. Appl.* **64** (1985) 199-214

[B1] D. Bini, "Parallel solution of certain Toeplitz linear systems", *SIAM J. Comput.* **13** (1984) 268-276.

[B2] D. Bini, "Tensor and border rank of certain classes of matrices and the fast evaluation of determinant, inverse matrix and eigenvalues", *Calcolo* **22** (1985) 209-227.

[BP1] D. Bini, V. Pan, "Efficient algorithms for the evaluation of the eigenvalues of (block) banded Toeplitz matrices", *Math. Comp.***50** (1988) 431-448.

[BP2] D. Bini, V. Pan, "Polynomial division and its computational complexity", *J. of Complexity* **2** (1986) 179-203.

[BGH] A. Borodin, J. von zur Gathen, and J. Hopcroft, "The parallel matrix and GCD computations", *Information and Control* **52,3** (1982) 241-256.

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

[CKL] J. Canny, E. Kaltofen, Y. Lakshman, "Solving systems of nonlinear polynomial equations faster", Proc. ACM-SIGSAM ISSAC-89 (1989) 121-128.

[DG] P. Del Sarte, Y. Genin, "On the splitting of classical algorithms in linear prediction theory", *IEEE Trans. on Acoustics, Speech, and Signal Processing,* ASSP-35, **5** (1987) 645-653.

[EG] D. Eppstein, Z. Galil, "Parallel algorithmic techniques for combinatorial computation," *Ann. Review of Computer Science* **3** (1988) 233-283.

[GL] G. H. Golub, C.F. van Loan, *Matrix Computations.* Johns Hopkins Univ. Press, Baltimore, Maryland, 1989.

[PR] V. Pan, J. Reif, "Displacement structure and processor efficiency of fast parallel Toeplitz computations, Tech. Rep. TR 87-4, *Comp. Sci. Dept., SUNYA,* Albany 1987.

[P] V. Pan, "Sequential and parallel complexity of approximate evaluation of polynomial zeros", *Computer & Math. (with Applics.)* **14** (1987), 8, 591-622.

[RT] J. Reif and S. Tate, "Optimal size division circuits", *Proc. 21-st Ann. ACM Symp. Theory of Computing,* (1989) 264-270.

[SB] J. Stoer, R. Bulirsch, *Introduction to Numerical Analysis,* Springer, Berlin,

matrices. Due to their different nature, a more fair comparison of the complexities of such two algorithms could be made under the Boolean model of computation in terms of their bit-cost, where we should consider the number of Boolean operations involved as a function in n, k and the prescribed number of correct digits in the output. Such an analysis would be closely related to the study of the numerical stability of the algorithms.

3) Another interesting open problem is to give a nontrivial lower bound on the complexity of the computation of $\det A$ for a banded Toeplitz matrix A under the sequential arithmetic model of computation. Since $\det A$ is a multivariate polynomial of degree n in $k+1$ variables, a trivial lower bound on the complexity is $\log_2 n$ (compare [BM]). This lower bound holds under the sequential arithmetic model where only arithmetic operations are allowed as the unit cost operations. We observe that the arithmetic complexity of the solution can even be made constant if we also assign the unit cost to more complicated computations such as exponentiation of the variables to the n-th powers or computation of logarithmic and trigonometric functions of complex variable depending on n (see [Tr1]). This fact again suggests that the Boolean model should be more appropriate in order to analyze the complexity of the problem.

References.

[AG] G.S. Ammar, W.G. Gragg, "The generalized Schur Algorithm for the superfast solution of Toeplitz systems", *Lecture Notes in Mathematics* **1237**, pp.315-330, Springer, Berlin 1987.

[AHU] A.V. Aho, J.E. Hopcroft, and J.D. Ullman, *The Design and Analysis of Computer Algorithms,* Addison-Wesley, 1976.

[BC1] D. Bini, M. Capovani,"Fast parallel and sequential computations and spectral properties concerning band Toeplitz matrices", *Calcolo* **20** (1983) 177-189.

[BC2] D. Bini, M. Capovani, "Spectral and computational properties of band symmetric Toeplitz matrices", *Lin. Alg. Appl.* **52** (1983) 99-126.

[BC3] D. Bini, M. Capovani,"Tensor rank and border rank of band Toeplitz matrices, *SIAM J. Comput.* **16** (1987) 252-258.

by using matrix polynomials, that is, polynomials whose coefficients are matrices. As this is shown in [BP1], the first stage of algorithm 3 still applies to block matrices, provided that we only use the "right" quotients and the "right" remainders when we perform matrix-polynomial division, that is, in the matrix-polynomial division, the divisor is kept on the right of the quotient. The extensions of stages 2 and 3 is straightforward; it is also easy to verify the extension of the identity $F^i\mathbf{e}_k = \tilde{T}^{-1}\mathbf{e}_{k-i}$ to the block case.

By using the results of section 2 we deduce that, in the case of block matrices, the cost of algorithm 3 is given by $O(m^3 k \log n + m^2 k \log k \log n + m^3 k^2 \log^2 k)$ arithmetic operations, which can be reduced to $O(m^3 k \log n + m^2 k \log k \log n + m^3 k \log^2 k)$ if the block Toeplitz matrix V_s has a block LU -factorization.

6. Concluding remarks. 1) If k is small, then the asymptotically fast auxiliary algorithms that support the bounds of tables of section 2 become inferior to straightforward algorithms, which should be used in our computations in this case with the respective changes of the overall cost bounds, and similarly in the case where the computations are performed over the fields of constants that do not support fast Fourier transform (FFT), as well as where we apply the superfast algorithms of [CKL] (dealing with $k \times k$ transposed Vandermonde matrices) and of [AG] (computing the determinant of $k \times k$ Toeplitz matrices). The latter algorithms only involve $O(k \log^2 k)$ arithmetic operations but for smaller k they are inferior to the algorithm of [GL], p. 181, and [DG], respectively, which use roughly $2.5k^2$ and $2k^2$ arithmetic operations, respectively. (In fact the $O(k \log^2 k)$ algorithm of [CKL] is nonpractical due to the problems of numerical stability.)

2) Of our two sequential algorithms 2 and 3 for computing $p(\lambda) = \det(T - \lambda I)$, algorithm 2 computes an approximation to $p(\lambda)$ in $O(k \log n + k^2 \log k \log(bk))$ arithmetic operations (where b measures the absolute precision of the approximation to the zeros of an auxiliary k-th degree polynomial), and algorithm 3 exactly computes $p(\lambda)$ (in infinite precision arithmetic) in $O(k \log k \log n)$ arithmetic operations. Algorithm 3 can be extended to the computation of the ratio $p(\lambda)/p'(\lambda)$ and to the case of block

Therefore, the problem is reduced to the computation of the Toeplitz matrices V_s and V_s', or equivalently, of the vectors $\mathbf{u}$ and $\mathbf{u}'$ (note that all the derivatives are with respect to the variable λ and not z.)

Now, taking derivatives at stage 2 of algorithm 3, we obtain that

$$\tilde{T}\mathbf{u}' = a_s^{-1}\boldsymbol{\theta}' - \tilde{T}'\mathbf{u},$$

so that the evaluation of $\mathbf{u}'$, and therefore, of V_s', is reduced to:

a) computing the vector $\boldsymbol{\theta}'$;

b) computing the product $\tilde{T}'\mathbf{u}$, which amounts to the shift and scaling of the vector $\mathbf{u}$ (due to the definition of $\tilde{T}$, the matrix $\tilde{T}'$ only has nonzero entries on the single diagonal filled with a_s^{-1});

c) solving a lower triangular $k \times k$ Toeplitz system.

Adding few more operations at stage 1 of algorithm 3, we may compute the coefficients θ_i' of the derivative of the polynomial $\psi' = \sum_{i=0}^{k-1} \theta_i' z^i$. Indeed, taking derivatives at stage 1 of algorithm 3 (recall, from the end of section 4, that the derivative of φ with respect to λ is $-z^r$), we obtain that

$$\psi_{i+1}' = 2\psi_i\psi_i' - Q_i z^r \bmod \varphi,\ i = 0, \ldots, h-1,$$

where $\psi_i^2 = Q_i\varphi + \psi_{i+1}$. For a fixed i, assuming the coefficients of ψ_i, ψ_i', and Q_i precomputed, we may evaluate the coefficients of ψ_{i+1}' by using one polynomial multiplication and one polynomial division. When we arrive at the coefficients of $\psi' = \psi_h'$, the computational cost of stage 1 of algorithm 3 is roughly doubled. Thus, the evaluation of the ratio $p(\lambda)/p'(\lambda)$ can be carried out for the cost that roughly doubles the cost of the evaluation of $p(\lambda)$.

In the case where T is an $n \times n$ block matrix with $m \times m$ blocks $t_{j-i} \in \mathbf{C}^{m\times m}$, algorithm 3 can be extended by means of the technique used in [BP1]. The main difficulty of dealing with block entries is that the entries no more belong to a number field but to a noncommutative algebra. Thus, we have to take care of commutativity and of nonsingularity of the blocks. The transition from scalars to blocks leads to matrix-difference equations, which can be solved in terms of block Frobenius matrices,

Stage 3. *Compute* $(-1)^s a_s^{n+s} \det V_s$.

By using the results of section 2 we may easily check that the arithmetic cost of algorithm 3 is given by: $O(k \log k \log n)$ at stage 1, $O(k \log k)$ at stage 2, $O(\log n + s^2 \log^2 s)$ at stage 3 (reduced to $O(\log n + s \log^2 s)$ if the matrix V_s is strongly nonsingular), and $O(k \log k(\log n + k \log k))$ in the entire algorithm (reduced to $O(k \log k \log n)$ if the matrix V_s is strongly nonsingular).

Under the parallel arithmetic model of computation the algorithm 3 requires $O(\log k \log n)$ parallel steps and k^2 processors.

Since the matrix $A - \lambda I$ is still a $(k+1)$-diagonal Toeplitz matrix, algorithms 1, 2 and 3 can be applied to compute the value that the polynomial $p(\lambda) = \det(A - \lambda I)$ takes on at a point λ. Moreover, these algorithms can also be used to compute $\det(A - \lambda B)$ in the algorithms for the generalized eigenvalue problem, $A\mathbf{x} = \lambda B\mathbf{x}$, where A and B are band Toeplitz matrices (in this case, r and s are the greatest values of the corresponding parameters in the matrices A and B).

In the next section, we will modify algorithm 3 in order to compute the ratio $p(\lambda)/p'(\lambda)$, where $p(\lambda) = \det(A - \lambda I)$, with increasing the asymptotic cost only by a constant factor. In this case, we let λ vary (instead of setting $\lambda = 0$), and we replace a_0 by $a_0 - \lambda$ in the definition of $\varphi(z)$, so that $\varphi(z)$ turns into $\varphi(z, \lambda) = \sum_{i=0}^{k} z^i a_{i-r} - \lambda z^r$, and the derivative of φ with respect to λ is $-z^r$.

5. Extension to computing the ratio $p(\lambda)/p'(\lambda)$ and to the case of block matrices. Computing the ratio $p(\lambda)/p'(\lambda)$, we will follow the approach of [BP1]. We recall the following identity:

$$(\det X)' = \operatorname{trace}(\operatorname{Adj}(X)X'),$$

where $\operatorname{Adj}(X)$ is the adjugate (adjoint) matrix of a matrix X, and X' is the matrix whose (i,j)-entry is the first derivative of the (i,j)-entry of X. Thus, $\det X/(\det X)' = 1/\operatorname{trace}(X^{-1}X')$. From (3), we have

$$p(\lambda)/p'(\lambda) = \det V_s/(\det V_s)'$$

The computation of the vector $\mathbf{u}$ in (8) can be carried out by means of the technique of repeated squaring used in [BP1], since $\varphi(F) = 0$ for the polynomial $\varphi(z) = \sum_{i=0}^{k} z^i a_{i-r}$. Therefore, $F^{n+s-1} = \psi(F)$ if $\psi(z)$ is the polynomial of degree $k-1$ such that $\psi(z) = z^{n+s-1} \bmod \varphi(z)$. Thus, we may proceed in the following way: first compute the coefficients of the polynomial

$$\psi(z) = \sum_{i=0}^{k-1} \theta_i z^i = z^{n+s-1} \bmod \varphi(z)$$

by means of repeated squaring modulo $\varphi(z)$, then compute the vector $\mathbf{u} = a_s^{-1}\psi(F)\mathbf{e}_k$. Towards this purpose, observe that $F^i\mathbf{e}_k = \tilde{T}^{-1}\mathbf{e}_{k-i}$, $i = 0, \ldots, k-1$, where $\tilde{T}$ is the lower triangular $k \times k$ Toeplitz matrix whose first column is $(1, -g_{k-1}, \ldots, -g_1)^T$. Consequently,

$$\mathbf{u} = a_s^{-1} \sum_{i=0}^{k-1} \theta_i F^i \mathbf{e}_k = a_s^{-1} \sum_{i=0}^{k-1} \tilde{T}^{-1}\theta_i \mathbf{e}_{k-i} = a_s^{-1}\tilde{T}^{-1}\boldsymbol{\theta}, \quad \boldsymbol{\theta} = (\theta_{k-1}, \ldots, \theta_0)^T.$$

Now, we are ready to present our third algorithm, where, for the sake of simplicity, we assume that $n + s - 1 = 2^h$:

Algorithm 3 (exact solution algorithm)

Stage 1. *Compute the coefficients θ_j of the polynomial*

$$\psi(z) = \sum_{j=0}^{k-1} \theta_j z^j = z^{n+s-1} \bmod \varphi(z)$$

by performing to the following steps:

$$\psi_0(z) = z,$$

$$\psi_{i+1}(z) = \psi_i^2(z) \bmod \varphi(z), \ i = 0, 1, \ldots, h-1,$$

$$\psi(z) = \psi_h(z).$$

Stage 2. *Compute the vector $\mathbf{u}$ by solving the $k \times k$ triangular Toeplitz system*

$$\tilde{T}\mathbf{u} = a_s^{-1}\boldsymbol{\theta},$$

where $\tilde{T}$ is the lower triangular $k \times k$ Toeplitz matrix whose first column is $(1, -a_{s-1}/a_s, \ldots, -a_0/a_s)^T$ and where $\boldsymbol{\theta} = (\theta_{k-1}, \ldots, \theta_0)^T$.

Algorithm 2 can be performed in $O(k \log n + k^2 \log k \log(bk))$ arithmetic operations, which may be reduced to $O(k \log n + k^2 \log k \log b)$ if the matrix V_s is strongly nonsingular. Specifically, by using proposition 1 and the first table of section 2, we obtain that we need $O(k^2 \log k \log b)$ operations at stage 1; $O(k \log^2 k)$ operations to solve the transposed Vandermonde system at stage 2; $O(k \log n)$ operations in order to compute α_i^σ for all i (by means of repeated squaring) at stage 3; $O(k \log^2 k)$ in order to compute the product of the transposed Vandermonde matrix Ω_k by a vector, also at stage 3; $O(\log n)$ for computing a_s^{n+s} (by means of repeated squaring) at stage 4; $O(s^2 \log^2 s)$ for computing $\det V_s$, also at stage 4, which can be reduced to $O(s \log^2 s)$ if V_s is strongly nonsingular.

Analogously, it is possible to check that, under the parallel arithmetic model of computation, the algorithm has the following arithmetic parallel cost: $O(\log n + k \log k \log(bk))$ steps with k^2 processors.

Finally we will present our third algorithm, exploiting the expression of [BP1] for the solution of the linear difference equation in terms of the following $k \times k$ Frobenius matrix:

$$F = \begin{pmatrix} 0 & 1 & & & \\ & 0 & \ddots & & \\ & & \ddots & \ddots & \\ & & & 0 & 1 \\ g_0 & g_1 & \cdots & g_{k-2} & g_{k-1} \end{pmatrix}, \quad g_i = -a_{i-r}/a_s, \ i = 0, \ldots, k-1.$$

It is easy to check that the solution $\{x_i\}$ of (4) satisfying (5) can be rewritten in the following way:

$$\begin{pmatrix} x_{i-k+2} \\ \vdots \\ x_{i+1} \end{pmatrix} = F^i a_s^{-1} \mathbf{e}_k.$$

Here and hereafter, $\mathbf{e}_i$ denotes the i-th column of the $k \times k$ identity matrix. Therefore, the last $2s-1$ components of the vector $\mathbf{x}$ are equal to the last $2s-1$ components of the vector

$$\mathbf{u} = F^{n+s-1} a_s^{-1} \mathbf{e}_k. \tag{8}$$

so that $\Omega = \Omega_{k\times(n+s+k-1)}$ is the $k \times (n+s+k-1)$ transposed Vandermonde matrix (here we suppose for simplicity that $\alpha_i \neq \alpha_j$ if $i \neq j$; otherwise, see [SB], [Tr2]).

Imposing the initial conditions (5) implies that

$$\Omega_k \begin{pmatrix} \xi_1 \\ \xi_2 \\ \vdots \\ \xi_k \end{pmatrix} = \begin{pmatrix} x_{-k+2} \\ x_{-k+3} \\ \vdots \\ x_1 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ \vdots \\ a_s^{-1} \end{pmatrix} \tag{6}$$

where Ω_k is the $k \times k$ transposed Vandermonde matrix,
$\Omega_k = (\alpha_j^i)_{i=0,\ldots,k-1,\ j=1,\ldots,k}$. The last $k = s + r \geq 2s$ components of the vector $\mathbf{x}$ are given by

$$\begin{pmatrix} x_{n-r+1} \\ \vdots \\ x_{n+s} \end{pmatrix} = \Omega_k \ \mathrm{Diag}(\alpha_1^\sigma, \alpha_2^\sigma, \ldots, \alpha_k^\sigma) \begin{pmatrix} \xi_1 \\ \xi_2 \\ \vdots \\ \xi_k \end{pmatrix}, \ \sigma = n + s - 1. \tag{7}$$

Now, we are ready to present the following algorithm, which approximates $\det A$ with any precision:

Algorithm 2 (approximation algorithm)

Stage 1. *Approximate, up to within absolute error bound 2^{-b}, the zeros $\alpha_1, \alpha_2,$ $ldots, \alpha_k$ of the polynomial $\varphi(z)$.*

Stage 2. *Solve the system (6) (with the transposed Vandermonde coefficient matrix Ω_k).*

Stage 3. *Compute $x_{n-s+2}, \ldots, x_{n+s}$ in (7) obtaining V_s.*

Stage 4. *Compute* $\det A = (-1)^s a_s^{n+s} \det V_s$ *(V_s is a Toeplitz matrix).*

We observe that $\det A$, as a function of $\alpha_1, \alpha_2, \ldots, \alpha_k$, may be numerically ill-conditioned, that is, a small approximation error in the zeros
$\alpha_1, \alpha_2, \ldots, \alpha_k$, may imply a large error in the result. This means that in the complete complexity analysis of this algorithm we should estimate how small must be 2^{-b} (where b is a function $b = b(\varepsilon, n, k)$) in order to have absolute error at most ε in the result. Here, we simply present the arithmetic cost of the algorithm as a function in n, k and b.

Algorithm 1 (parallel implementation)

Stage 1. *Compute* T^{-1} *by solving the triangular Toeplitz system* $T\mathbf{x} = \mathbf{e}_1$, *where* $\mathbf{e}_1^T = (1, 0, \ldots, 0)$, *for the cost of* $O(\log(n+s))$ *steps using* $n(k+1)$ *processors.*

Stage 2. *Compute* $\det V_s$ *using* $O(\log^2 s)$ *steps and* s^2 *processors. (*V_s *is an* $s \times s$ *Toeplitz matrix with rational entries.)*

Stage 3. *Compute* $\det A = a_s^{n+s} d$ *by means of repeated squaring by using* $1 + \lceil \log_2(n+s) \rceil$ *steps on a single processor.*

The overall number of parallel steps required by the latter algorithm is quite small, that is, $O(\log n + \log^2 s)$, but the number of processors, and therefore, the total arithmetic work, is very large.

Trying to improve algorithm 1, we observe that we do not need to compute all the entries of T^{-1} in order to compute V_s, but it is sufficient to compute the last $2s-1$ components of the vector $\mathbf{x}$ such that

$$T\mathbf{x} = \mathbf{e}_1, \quad \mathbf{e}_1 = (1, 0, \ldots, 0)^T.$$

Due to the Toeplitz structure of T, this system can be viewed as the following linear difference equation:

$$\sum_{j=-r}^{s} a_j x_{j+i-s} = 0, \ i = 2, \ldots, n+s, \tag{4}$$

where the auxiliary variables satisfy the initial conditions

$$x_1 = a_s^{-1}, \ x_0 = x_{-1} = \ldots = x_{-k+2} = 0. \tag{5}$$

We will exploit this representation in the two algorithms that will follow.

Expressing the solution of the homogeneous difference equation in terms of the zeros $\alpha_1, \alpha_2, \ldots, \alpha_k$ of the associated polynomial $\varphi(z) = \sum_{i=0}^{k} z^i a_{i-r}$ and in terms of the new unknowns $\xi_1, \ldots, \xi_k$(see [SB]), we have

$$\mathbf{x} = \Omega \begin{pmatrix} \xi_1 \\ \xi_2 \\ \vdots \\ \xi_k \end{pmatrix}, \ \Omega = (\alpha_j^i), \ i = 0, 1, \ldots, n+s+k-2; \ j = 1, 2, \ldots, k,$$

reduction is by means of embedding the matrix A as a block P_n into an $(s+n)\times(s+n)$ triangular matrix T.

In the case where A is a banded Toeplitz matrix, such a reduction was effectively exploited in [B1], [BC3]. We will follow this line. In particular, we will embed a given $(r+s+1)$-banded Toeplitz matrix $A=(a_{j-i})$ of size $n\times n$, $a_i=0$ if $i>s$ or $i<-r$, $a_s a_{-r}\neq 0$, into a lower triangular Toeplitz matrix T of size $(n+s)\times(n+s)$ whose first column is the $(n+s)$-vector $(a_s,a_{s-1},\dots,a_{-r},0,\dots,0)^T$,

$$T=\begin{pmatrix} T_s & 0_{s\times(n-s)} & 0_s \\ & & 0_{(n-s)\times s} \\ & A & \\ & & T_s \end{pmatrix},$$

where T_s is a lower triangular Toeplitz matrix. Here and throughout the paper, we assume $s\le r$.

For instance, for $s=r=2$ and $n=4$ we have the following embedding:

$$\begin{pmatrix} a_2 & & & & & \\ a_1 & a_2 & & & & \\ a_0 & a_1 & a_2 & & & \\ a_{-1} & a_0 & a_1 & a_2 & & \\ a_{-2} & a_{-1} & a_0 & a_1 & a_2 & \\ 0 & a_{-2} & a_{-1} & a_0 & a_1 & a_2 \end{pmatrix}.$$

Fact 1 implies that

$$\det A=(-1)^{(n+s+1)s}\det T\det V_s=(-1)^{(n+s+1)s}a_s^{n+s}\det V_s. \tag{3}$$

Moreover, T^{-1} is a lower triangular Toeplitz matrix, since T is a nonsingular lower triangular Toeplitz matrix; in fact, the set of lower triangular Toeplitz matrices is a matrix algebra. Thus, the problem of computing $\det A$ is reduced to that of computing the determinant of the $s\times s$ Toeplitz matrix V_s. The same approach applies to computing $p(\lambda)=\det(A-\lambda I)$ at any point λ since $A-\lambda I$ remains a $(k+1)$-diagonal Toeplitz matrix.

4. The algorithms. Here is the first (and the most straightforward but not the most effective) implementation of this approach:

Evaluation of	Arithmetic operations
$q(x)$ and $r(x)$	$O(m^3n + m^2n\log n)$
$w(x) = u(x)v(x)$	$O(m^3n + m^2n\log n)$
T^{-1}	$O(m^3n + m^2n\log n)$
$\det A$	$O(m^3n^2\log^2 n)$ $(or O(m^3n\log n))$

In this table, u, v and w are matrix polynomials, q and r are the right quotient and the right remainder of the division of u and v, i.e., $u(x) = v(x)q(x) + r(x)$, degree$(r(x))$ <degree$(v(x))$, T is a block triangular block Toeplitz matrix, A is a block Toeplitz matrix, and the improved cost bound in the fourth row applies if the block Toeplitz matrix has a block LU-factorization.

3. The basic reduction of the problem. Hereafter, the subscripts of the block matrices will denote their size; for square blocks we will write s, n rather than $s \times s,\ n \times n$. We will rely on the following simple fact:

Fact 1. *Let a pair of $(n+s)\times(n+s)$ nonsingular matrices T and T^{-1} be represented as 2×2 block matrices as follows:*

$$T = \begin{pmatrix} M_{s\times n} & N_s \\ P_n & Q_{n\times s} \end{pmatrix},\ T^{-1} = \begin{pmatrix} U_{n\times s} & W_n \\ V_s & Z_{s\times n} \end{pmatrix}.$$

Then

$$\begin{pmatrix} 0_{s\times n} & I_s \\ P_n & Q_{n\times s} \end{pmatrix} = \begin{pmatrix} V_s & Z_{s\times n} \\ 0_{n\times s} & I_n \end{pmatrix} T,$$

whence

$$(-1)^{s(n+s+1)} \det P_n = \det V_s \det T,$$

$$P_n^{-1} = W_n - U_{n\times n} V_s^{-1} Z_{s\times n},$$

where 0 and I denote the null and the identity matrices, whose size is shown by the subscripts. ∎

Fact 1 can be used in order to reduce the evaluation of the determinant (and of the inverse) of an $n\times n$ banded matrix $A = (a_{ij})$ such that $a_{ij} = 0$ if $j - i \geq s$ to the evaluation of the the determinant (and of the inverse) of the $s\times s$ matrix V_s. The

size of the blocks or of the matrix coefficients of the polynomials, then we have the following sequential arithmetic cost bounds:

We also need the following auxiliary result:

Proposition 1. *Given an $n \times n$ Toeplitz matrix A, the value* $\det A$ *can be computed with $O(n \log^2 n)$ arithmetic operations if A is strongly nonsingular; otherwise, $O(n^2 \log^2 n)$ arithmetic operations are sufficient to compute* $\det A$. *Under the parallel model of computations, $O(\log^2 n)$ arithmetic steps with $O(n^2)$ processors are sufficient to compute* $\det A$, *if the entries of A are rational.*

Proof. If A is strongly nonsingular then there exists the LU-factorization of A. The diagonal entries of U can be computed by means of the generalized Schur algorithm having the arithmetic cost $O(n \log^2 n)$ [AG]. For a general Toeplitz matrix A it is sufficient to apply the following identity:

$$A_i = A + \alpha \omega^i I, \ i = 0, \ldots, h-1, \quad h \det A = \sum_{i=0}^{h-1} \det A_i$$

where $\log_2 h = \lceil \log_2(n+1) \rceil$, ω is a primitive h-th root of 1, and $\alpha > ||A||_\infty$. In this case the matrices A_i are strongly nonsingular. Concerning the parallel model of computation, the algorithm presented in [PR] computes $\det A$ in $O(\log^2 n)$ steps with $O(n^2)$ processors. ∎

Remark. We may relax the strong nonsingularity assumption if we only seek $|\det A|$. Indeed, we may compute the diagonal entries of the U factor in the LU-factorization of $A^T A$ (see [AG]) and then compute $\det(A^T A) = |\det A|^2$ in $O(n \log^2 n)$ arithmetic operations, for $A^T A$ is a strongly nonsingular Toeplitz-like matrix. Also, $|\det A|$ is given by the QR factorization of A, which can be computed in $O(n^2)$ arithmetic operations.

The algorithms for polynomial division and polynomial multiplication that give the cost bounds of the above table, as well as the algorithms for inverting a triangular Toeplitz matrix and for computing the determinant of a general Toeplitz matrix, can be easily extended to the case of matrix polynomials (i.e., polynomials having matrix coefficients) and to the case of block Toeplitz matrices (compare [BP1]). If m is the

remarks.

The authors thank the referees for helpful comments.

2. Preliminaries. The next table summarizes some upper estimates for the arithmetic complexity of certain computations with polynomials and structured matrices. In this table, $q(x)$ and $r(x)$ denote the quotient and the remainder of the division of two input polynomials, $u(x)$ and $v(x)$, of degrees at most n; A, T, T_k and Ω denote $n \times n$ matrices, where A is a Toeplitz matrix, T is a triangular Toeplitz matrix, T_k is a triangular Toeplitz matrix with the band-width at most $k+1$; the evaluation of all the zeros of $u(x)$ means their approximations within $\eta = 2^{-b}$; this specifically means performing, with the (lowest possible) precision of $O(n\eta)$ bits, the computation that output n numbers whose distances from the corresponding zeros are bounded by $\eta = 2^{-b}$ where b is a positive integer; Ω is the transpose of an $n \times n$ Vandermonde matrix, we need to assume that all the entries of Ω are rational in order to support the processor bound n^2 of the last line of the table ([PR]), otherwise we need $n^{3.5}$ processors.

Evaluation of	Arithm. operations	Parallel steps	Processors	Reference
$q(x)$ and $r(x)$	$O(n \log n)$	$O(\log n)$	n^2	[B1],[BP2]
		$O(\log n \log\log n)$	$n/\log\log n$	[RT]
$w(x) = u(x)v(x)$	$O(n \log n)$	$O(\log n)$	n	[BM]
All zeros of $u(x)$	$O(n^2 \log n \log b)$	$O(n \log n \log(bn))$	$n \log b/\log(bn)$	[P]
T^{-1}	$O(n \log n)$	$O(\log n)$	n^2	[B1],[BP2]
		$O(\log n \log\log n)$	$n/\log\log n$	[RT]
T_k^{-1}	$O(nk \log n)$	$O(\log n)$	nk	[B1]
$\Omega_n \mathbf{u}$, $\Omega_n^{-1}\mathbf{u}$	$O(n \log^2 n)$	$O(\log^2 n)$	n^2	[CKL]

We will call a matrix *strongly nonsingular* if its all leading principal submatrices are nonsingular; for a strongly nonsingular matrix there exists its unique LU-factorization where L is a unit lower triangular matrix. The set of the matrices that are not strongly nonsingular is a zero-measure subset and, moreover, an algebraic manifold of a lower dimension in the space of all the $n \times n$ matrices.

We implement our method in the form of three distinct approaches. The simplest approach gives algorithm 1 below, which we present mostly for demonstration and also because its parallel arithmetic time cost is very small under the PRAM model: $O(\log n + \log^2 k)$ steps for computing both $p(\lambda)$ and $p(\lambda)/p'(\lambda)$, although its sequential arithmetic cost is too high (beeing of an order higher than n).

In the second implementation, we adopt the model of approximate computation where the zeros of the k-th degree polynomial associated to the linear difference equation must be approximated within a fixed absolute error bound η. In this case, we obtain algorithm 2 which we implement by using $O(k \log n + k^2 + C(k, \eta))$ arithmetic operations. As in [Tr2], the algorithm cannot be extended to the case of block matrices and does not allow the direct computation of $p(\lambda)/p'(\lambda)$.

The third implementation, obtained by means of the technique of repeated squaring a Frobenius matrix, yields algorithm 3, which involves $O(k \log k \log n)$ arithmetic operations and can be implemented under the PRAM model for the cost of $O(\log k \log n)$ parallel steps with k^2 processors. This algorithm can be extended to the case of block matrices for the cost of $O(m^3 k(\log n + \log^2 k) + m^2 k \log k \log n)$ arithmetic operations; here m is the dimension of the blocks.

The above bounds on the computational complexity have been obtained relying on the known asymptotically fast auxiliary algorithms for the computations with polynomials and with general and special matrices over the field of complex numbers.

We will present our results in the following order: in section 2 we will recall the cost estimates of the auxiliary algorithms for the computations with polynomials and structured matrices. In section 3, we will present a matrix identity that allows us to reduce the computations for the banded Toeplitz matrix A to the computations for an $s \times s$ triangular Toeplitz matrix. In section 4, we will present the three algorithms (1,2 and 3) for computing the values of the characteristic polynomial $p(\lambda)$ of a banded Toeplitz matrix together with their cost estimates. This will also show the basic techniques involved. In section 5 we will indicate how to extend the algorithms to computing both $p(\lambda)$ and $p(\lambda)/p'(\lambda)$ in the case where A is a banded Toeplitz matrix or a banded Toeplitz block matrix. We leave section 6 for a discussion and concluding

[B2] (compare also [BC3]). The high sequential arithmetic cost of the algorithms of [BC1],[BC2] is substantially reduced in [Tr1] and [Tr2] for the price of changing the model of computation. Instead of computing the polynomial of (1), the algorithm of [Tr2] approximates this polynomial with an absolute error at most ε, that is, computes a value $\tilde{p}(\lambda)$ such that $|\tilde{p}(\lambda) - p(\lambda)| \leq \varepsilon$.

The sequential arithmetic cost of this algorithm is $O(k \log n + k^3 + C(k, \eta))$, where $C(k, \eta)$ is the cost of approximating to all the zeros of an auxiliary polynomial of degree k with absolute errors at most η (see the table in section 2 where the record complexity estimates from [P] are given for the root-finding problem). The dependence of η on the output error bound ε, k and n has not yet been analyzed. Such an analysis would require to study the numerical conditioning of the output as a function of the zeros of the auxiliary polynomial. Due to the stage of computing polynomial zeros, the algorithm of [Tr2] only outputs an approximate (rather than exact) value of $p(\lambda)$, does not approximate to $p(\lambda)/p'(\lambda)$ and does not apply in the case where A is a block matrix. Technically, the algorithm of [Tr2] relies on the reduction of the problem to a linear difference equation, which is solved by its reduction to the associated characteristic equation.

In [BP1], it is shown how to keep the computation in the exact model, avoiding approximation and only slightly increasing the overall cost, that is, to $O(k \log k \log n + k^3)$. The idea is to solve the difference equation associated to the original problem by powering a Frobenius matrix and by using fast polynomial arithmetic. Unlike [Tr2], this approach can be easily extended to the computation of $p(\lambda)/p'(\lambda)$ and to the case where the matrix A is a block matrix (the case of block matrices is considered also in [T] from a general but not algorithmic point of view).

In this paper, we devise a new method for the evaluation of both $p(\lambda)$ and $p(\lambda)/p'(\lambda)$, which substantially improves the known complexity results. Our improvement stems from a simple but general and rather surprising reduction of the original problem to the computations for an auxiliary $s \times s$ Toeplitz matrix; to yield such a reduction and to make use of it, we had to incorporate and modify several techniques of [BC1],[BC2],[BP1] and [Tr2].

case (here $p\prime(\lambda)$ denotes the first derivative of the polynomial $p(\lambda)$ with respect to λ). If the ratio $p(\lambda)/p\prime(\lambda)$ is easily computable, then Newton's iteration method can be applied as a root-finder, and we may take advantage of its quadratic convergence.

In this context it is very important to devise efficient algorithms for the evaluation of the characteristic polynomial

$$p(\lambda) = p(a_{-r}, \ldots, a_s, \lambda) = \det(A - \lambda I) \tag{1}$$

and of the ratio $p(\lambda)/p'(\lambda)$, where

$$p'(\lambda) = q(a_{-r}, \ldots, a_s, \lambda) = \partial p(a_{-r}, \ldots, a_s, \lambda)/\partial\lambda \tag{2}$$

given $\{a_{-r}, \ldots, a_s, \lambda\} \cup \mathbf{C}$, $\mathbf{C}$ is a field of constants (to be certain, we may let it be the complex field). In this paper we present three such algorithms (see algorithms 1, 2 and 3 below). Performing algorithms 1 and 3, we remain in the set of rational functions in the indeterminates $a_{-r}, \ldots, a_s, \lambda$ ([BM]). We estimate the cost of these two algorithms under the sequential and parallel Random Access Machine (RAM and PRAM, respectively) models of arithmetic computations ([AHU], [EG], [BGH]) with the uniform cost measure given by the number of arithmetic operations involved (in the sequential case) and by the numbers of parallel arithmetic steps and processors needed in the parallel case. In the PRAM model we assume that in each step at most p arithmetic operations can be simultaneously performed, where p is the number of processors, and we will say that pt is the total work or the sequential time of a parallel t-step algorithm performed by using p processors. We say that an algorithm computes a function $p(a_{-r}, \ldots, a_s, \lambda)$, depending on k and n, for the cost $O(f(n,k))$ if there exists a constant c such that for any n and k and for any input $a_{-r}, \ldots, a_s, \lambda$, the algorithm computes $p(a_{-r}, \ldots, a_s, \lambda)$ with cost at most $cf(n,k)$.

In [BC1],[BC2], some methods are presented for computing (1) and (2) where A is a real symmetric or a complex Hermitian matrix. Their sequential arithmetic cost (their total work) is high; they are, however, in NC, and their parallel arithmetic time is only $O(\log n + \log^2 k)$ under the PRAM model. These algorithms are based on the technique of matrix embedding, and a generalization of this approach is presented in

On the evaluation of the eigenvalues of a banded Toeplitz block matrix

Dario Bini, Dipartimento di Matematica, Universitá di Pisa (supported by NSF Grant CCR 8805782 and by the Italian MPI 40% funds);
Victor Pan, Department of Computer Science, Columbia University, New York, NY 10027; Department of Mathematics and Computer Science, Lehman College, City University of New York and Department of Computer Science, State University of New York at Albany (supported by NSF Grant CCR 8805782 and by PSC CUNY Awards 668541 and 669210).

CUCS-024-90

Abstract. Let $A = (a_{ij})$ be an $n \times n$ Toeplitz matrix with bandwidth $k+1$, $k = r + s$, that is, $a_{ij} = a_{j-i}$, $i,j = 1,\ldots,n$, $a_i = 0$ if $i > s$ and if $i < -r$. We compute $p(\lambda) = \det(A - \lambda I)$, as well as $p(\lambda)/p'(\lambda)$, where $p'(\lambda)$ is the first derivative of $p(\lambda)$, by using $O(k \log k \log n)$ arithmetic operations. Moreover, if a_i are $m \times m$ matrices, so that A is a banded Toeplitz block matrix, then we compute $p(\lambda)$, as well as $p(\lambda)/p'(\lambda)$, by using $O(m^3 k(\log^2 k + \log n) + m^2 k \log k \log n)$ arithmetic operations. The algorithms can be extended to the computation of $\det(A - \lambda B)$ and of its first derivative, where both A and B are banded Toeplitz matrices. The algorithms may be used as a basis for iterative solution of the eigenvalue problem for the matrix A and of the generalized eigenvalue problem for A and B.

1. Introduction. The problem of computing the eigenvalues of an $n \times n$ banded (block) Toeplitz matrix $A = (a_{j-i})$ with bandwidth $k+1$, where $k = r + s$, $a_i = 0$ if $i > s$ and if $i < -r$, $a_s a_{-r} \neq 0$, has been recently considered by several authors. This problem can be numerically solved by applying any root-finding method to the characteristic equation $p(\lambda) = \det(A - \lambda I) = 0$, so that efficient algorithms for the evaluation of $p(\lambda)$ or for both $p(\lambda)$ and $p(\lambda)/p\prime(\lambda)$ are particularly useful in this