Computing 49, 339–347 (1993)

Computing
© Springer-Verlag 1993
Printed in Austria

Iterative Toeplitz Solvers with Local Quadratic Convergence

E. Linzer, Yorktown Heights and **M. Vetterli,** New York

Received July 22, 1991; revised October 13, 1992

Abstract — Zusammenfassung

Iterative Toeplitz Solvers with Local Quadratic Convergence. We study an iterative, locally quadratically convergent algorithm for solving Toeplitz systems of equations from [R. P. Brent, F. G. Gustavson and D. Y. Yun. "Fast solution of Toeplitz systems of equations and computation of Padé approximations", *J. Algorithms*, 1: 259–295, 1980]. We introduce a new iterative algorithm that is locally quadratically convergent when used to solve symmetric positive definite Toeplitz systems. We present a set of numerical experiments on randomly generated symmetric positive definite Toeplitz matrices. In these experiments, our algorithm performed significantly better than the previously proposed algorithm.

AMS (MOS) Subject Classifications: 65F10, 65B99

Key words: Toeplitz, iterative methods, steepest descent, quadratic convergence.

Iterative Toeplitz Solver mit lokal quadratischer Konvergenz. Wir studieren einen iterativen, lokal quadratisch konvergenten Algorithmus für die Lösung von Toeplitz-Systemen von Gleichungen von [R. P. Brent, F. G. Gustavson und D. Y. Y. Yun, "Fast solution of Toeplitz systems of equations and computation of Padé approximations", *J. Algorithms*, 1: 259–295, 1980]. Wir führen einen neuen iterativen Algorithmus ein, der lokal quadratisch konvergent ist, wenn er für positiv definite Toeplitz-Systeme gebraucht wird. Wir präsentieren eine Anzahl von numerischen Experimenten mit zufallsgenerierten, symmetrischen, positiv definiten Toeplitz-Matrizen. In diesen Experimenten ist unser Algorithmus entscheidend besser als der früher vorgeschlagene Algorithmus.

1. Introduction

An $n \times n$ matrix $\mathbf{A}$ is Toeplitz if it is constant along its diagonals; that is, if $\mathbf{A}_{i,j} = \mathbf{A}_{i-1,j-1}$. In this paper, we consider the case when $\mathbf{A}$ is symmetric and positive definite (hereafter, s.p.d.).

The problem of solving a system of linear equations of the form $\mathbf{Ax} = \mathbf{y}$, where $\mathbf{A}$ is an s.p.d. Toeplitz matrix, is ubiquitous in engineering and physics [1, 2]. The structure of $\mathbf{A}$ allows one to find $\mathbf{x}$ with direct methods that require many fewer operations than the $O(n^3)$ used in Gaussian elimination. So called "fast", or $O(n^2)$ [2, 3, 4, 5, 6], algorithms, most notably the Levinson recursion [4], exist for this problem. More recently, several "superfast" algorithms have been introduced that reduce the operation count to $O(n \log^2 n)$ [7, 8, 9, 10].

Strang [11] has suggested using iterative methods for the solution of $\mathbf{Ax} = \mathbf{y}$. (His specific suggestion was to use the preconditioned conjugate gradient method. See

[12] for an early mention of an iterative representation of $\mathbf{A}^{-1}$.) In the "ordinary" iterative method, we repeatedly solve

$$\mathbf{M}\mathbf{x}^{(k)} = (\mathbf{M} - \mathbf{A})\mathbf{x}^{(k-1)} + \mathbf{y}$$

for some easily invertible matrix $\mathbf{M}$ that is "close" to $\mathbf{A}$. If it is close enough, the vectors $\mathbf{x}^{(k)}$ will converge to $\mathbf{x}$. If $\mathbf{A}$ is symmetric and positive definite, the ordinary iteration can be accelerated in many ways; for example, with the use of the steepest descent or conjugage gradient algorithms. See [13, 14, 15, 16, 17] for some new iterative methods for solving Toeplitz systems of equations.

In standard iterative algorithms and the algorithms discussed in this paper, each iteration requires the multiplication of a vector by $\mathbf{A}$, the inversion of the "$\mathbf{M}$" system, and, possibly, some inner products. The most common application of iterative algorithms is to sparse systems, where a matrix-vector multiplication is inexpensive. A similar situation prevails with dense Toeplitz matrices, because the matrix-vector product $\mathbf{Aw}$ can be computed quickly with fast Fourier transforms (FFT's) in $O(n \log n)$ operations [2, 18].

Let $L(\mathbf{x})$ be a lower triangular Toeplitz matrix with first column $\mathbf{x}$. Let $\tilde{\mathbf{x}}$ be a vector with the components of $\mathbf{x}$ listed in reverse order. Also, let the first column of $\mathbf{A}^{-1}$ be $\mathbf{c} = (b_0, \mathbf{b}^t)^t$. The *Gohberg-Semencul* factorization expresses the inverse of a Toeplitz matrix in terms of the first row and first column of the inverse [19, 20]. Because $\mathbf{A}^{-1}$ is symmetric, we can write $\mathbf{A}^{-1}$ as a function of $\mathbf{c}$:

$$\mathbf{A}^{-1} = \frac{1}{b_0}\left(L\begin{pmatrix} b_0 \\ \mathbf{b} \end{pmatrix} L^t\begin{pmatrix} b_0 \\ \mathbf{b} \end{pmatrix} - L\begin{pmatrix} 0 \\ \tilde{\mathbf{b}} \end{pmatrix} L^t\begin{pmatrix} 0 \\ \tilde{\mathbf{b}} \end{pmatrix}\right). \tag{1}$$

The Gohberg-Semencul factorization is particularly convenient because it shows how to compute $\mathbf{A}^{-1}\mathbf{w}$ from $\mathbf{c}$ with $O(n \log n)$ operations using FFT's. Note that because $\mathbf{A}$ is s.p.d., we must have $b_0 > 0$.

Many direct algorithms first compute $\mathbf{c}$ and then use the Gohberg-Semencul factorization to compute $\mathbf{x}$. In the first paper that presented an $O(n \log^2 n)$ algorithm to solve Toeplitz systems, the authors also presented a method based on iterative refinement to improve the computed value of the first row and first column of the inverse of Toeplitz matrix [10]. We will call the symmetric version of that method Dynamic Iterative Improvement (hereafter, DII). As stated in [10], DII is locally quadratically convergent. V. Pan modified DII to develop a fast version of Newton's iteration for Toeplitz-like matrices [15].

The renewed interest in iterative Toeplitz solvers warrants a second look at DII, which we review in the next section. In Section 3, we present a new algorithm, Dynamically Conditioned Steepest Descent (hereafter, DCSD), to compute the Gohberg-Semencul factorization of $\mathbf{A}^{-1}$. Like DII, DCSD is locally quadratically convergent.

In section 4, we present a set of numerical experiments that is designed to compare the performance of DII and DCSD. In these experiments, DCSD behaved significantly better than DII.

2. Iterative Improvement

DII is conceptually simple. We wish to iteratively compute $\mathbf{c}$. Denote the approximate solution at the k^{th} step of the algorithm as $\mathbf{c}^{(k)}$. We begin with the approximation $\mathbf{c}^{(0)}$. This means that we have an approximate solution to the equation

$$\mathbf{A}\mathbf{c} = \mathbf{e}_1 , \tag{2}$$

where $\mathbf{e}_1 = (1, 0, \ldots, 0)^t$.

Let

$$\mathbf{N}((u_0, \mathbf{u})^t) \equiv \frac{1}{u_0}\left[L\begin{pmatrix} u_0 \\ \mathbf{u} \end{pmatrix} L^t\begin{pmatrix} u_0 \\ \mathbf{u} \end{pmatrix} - L\begin{pmatrix} 0 \\ \tilde{\mathbf{u}} \end{pmatrix} L^t\begin{pmatrix} 0 \\ \tilde{\mathbf{u}} \end{pmatrix}\right].$$

The approximation $\mathbf{c}^{(0)}$ can be substituted into the Gohberg-Semencul formula, (1), to obtain $\mathbf{N}(\mathbf{c}^{(0)})$, an approximation to $\mathbf{A}^{-1}$. Iterative improvement is then used to obtain a better estimate of $\mathbf{c}$, which in turn gives a better estimate for $\mathbf{A}^{-1}$. The process is repeated until the residue (measured in some norm) is sufficiently small. The algorithm is summarized below. It is the same algorithm as the algorithm given in [10], except that the we have removed computations that are made redundant by the fact that $\mathbf{A}$ is symmetric (the algorithm given in [10] is for general Toeplitz matrices).

Algorithm 1 (Dynamic iterative improvement)

```
Input $\mathbf{c}^{(0)}$
$k = 1$
$\mathbf{r}^{(0)} = \mathbf{e}_1 - \mathbf{A}\mathbf{c}^{(0)}$
While $\|\mathbf{r}^{(k-1)}\|$ is large
    $\mathbf{c}^{(k)} = \mathbf{c}^{(k-1)} + \mathbf{N}(\mathbf{c}^{(k-1)})\mathbf{r}^{(k-1)}$
    $\mathbf{r}^{(k)} = \mathbf{e}_1 - \mathbf{A}\mathbf{c}^{(k)}$
    $k = k + 1$
```

In standard iterative solution methods [3, 18], we estimate $\mathbf{A}^{-1}$ once and then do not improve this estimate. When these algorithms converge, they converge linearly; the norm of the error is multiplied by a constant (less than unity) during each iteration. The authors of [10] note that the fact that in DII the estimate for $\mathbf{A}^{-1}$ improves at about the same rate as the estimate for the solution vector implies that we will have quadratic convergence. We now give a formal statement and proof of quadratic convergence.

Theorem 2.1. *Define the error at the k^{th} iteration of DII as $\mathbf{s}^{(k)} = \mathbf{c}^{(k)} - \mathbf{c}$. If $\mathbf{A}$ is s.p.d., then $\exists\delta > 0$ and $M < \infty$ s.t. if $\|\mathbf{s}^{(0)}\|_1 < \delta$ then $\|\mathbf{s}^{(k)}\|_1 \to 0$ as*

$$\|\mathbf{s}^{(k+1)}\|_1 < M\|\mathbf{s}^{(k)}\|_1^2 .$$

Proof summary: With the identity,

$$\mathbf{A}\mathbf{c} = \mathbf{e}_1 ,$$

and (1) we can obtain

$$\|\mathbf{s}^{(k+1)}\|_1 \leq 6\|\mathbf{A}\|_1 \tau^2 \|\mathbf{s}^{(k)}\|_1^2 + O(\|\mathbf{s}^{(k)}\|_1^3), \tag{3}$$

where

$$\tau \equiv \frac{\|\mathbf{c}\|_1}{b_0}.$$

(See [17] for details.) Choosing $M > 6\|\mathbf{A}\|_1 \tau^2$ proves the theorem. □

The parameter τ can be bounded in terms of the reflection coefficients generated by the Levinson algorithm; see [21, lemma 2.4].

3. Steepest Descent

The iterative solution of a symmetric positive definite system may be obtained by the use of the *steepest descent* algorithm. The strategy in steepest descent is to minimize the quadratic form $\phi(\mathbf{x}) \equiv (\mathbf{x} - \mathbf{A}^{-1}\mathbf{y})^t\mathbf{A}(\mathbf{x} - \mathbf{A}^{-1}\mathbf{y})$, which yields the minimum value of 0 at $\mathbf{x} = \mathbf{A}^{-1}\mathbf{y}$. If we find $\mathbf{x}^{(k)}$ by moving from $\mathbf{x}^{(k-1)}$ along $-\nabla\phi(\mathbf{x}^{(k-1)})$, we obtain the steepest descent algorithm [3]:

Algorithm 2 (Steepest descent)

```
Input $\mathbf{x}^{(0)}$
$k = 1$
$\mathbf{r}^{(0)} = \mathbf{y} - \mathbf{A}\mathbf{x}^{(0)}$
While $\|\mathbf{r}^{(k-1)}\|$ is large
    $\alpha^{(k)} = \mathbf{r}^{(k-1)t}\mathbf{r}^{(k-1)}/\mathbf{r}^{(k-1)t}\mathbf{A}\mathbf{r}^{(k-1)}$
    $\mathbf{x}^{(k)} = \mathbf{x}^{(k-1)} + \alpha^{(k)}\mathbf{r}^{(k-1)}$
    $\mathbf{r}^{(k)} = \mathbf{r}^{(k-1)} - \alpha^{(k)}\mathbf{A}\mathbf{r}^{(k-1)}$
    $k = k + 1.$
```

Let $\lambda_{\max}(\mathbf{P})$ (resp., $\lambda_{\min}(\mathbf{P})$) be the largest (resp., smallest) eigenvalue value of $\mathbf{P}$. The iterates $\mathbf{x}^{(k)}$ will satisfy

$$\sqrt{\phi(\mathbf{x}^{(k)})} \leq 2\frac{\sqrt{\kappa_2(\mathbf{A})} - 1}{\sqrt{\kappa_2(\mathbf{A})} + 1}\sqrt{\phi(\mathbf{x}^{(k-1)})} \tag{4}$$

where $\kappa_2(\mathbf{A}) = \|\mathbf{A}\|_2\|\mathbf{A}^{-1}\|_2 = \lambda_{\max}(\mathbf{A})/\lambda_{\min}(\mathbf{A})$ is the l_2 norm condition number of $\mathbf{A}$. Although (4) does not guarrantee convergence, it can be shown that steepest descent (unlike iterative improvement) converges globally when used to solve s.p.d. systems; see [3; page 363].

We now borrow the idea of preconditioning from the preconditioned conjugate gradient method [3, 18]. The idea is to compute instead

Algorithm 3 (Preconditioned steepest descent)

```
Input $\mathbf{x}^{(0)}$
$k = 1$
$\mathbf{r}^{(0)} = \mathbf{y} - \mathbf{A}\mathbf{x}^{(0)}$
While $\|\mathbf{r}^{(k-1)}\|$ is large
    Solve $\mathbf{M}\mathbf{z}^{(k-1)} = \mathbf{r}^{(k-1)}$
```

```
        $\alpha^{(k)} = \mathbf{z}^{(k-1)t}\mathbf{r}^{(k-1)}/\mathbf{z}^{(k-1)t}\mathbf{A}\mathbf{z}^{(k-1)}$
        $\mathbf{x}^{(k)} = \mathbf{x}^{(k-1)} + \alpha^{(k)}\mathbf{z}^{(k-1)}$
        $\mathbf{r}^{(k)} = \mathbf{r}^{(k-1)} - \alpha^{(k)}\mathbf{A}\mathbf{z}^{(k-1)}$
        $k = k + 1$
```

for some easily invertible symmetric positive definite matrix $\mathbf{M}$ that approximates $\mathbf{A}$. This is mathematically equivalent to solving the transformed system $\hat{\mathbf{A}}\hat{\mathbf{x}} = \hat{\mathbf{y}}$, where $\hat{\mathbf{A}} \equiv \mathbf{M}^{-1/2}\mathbf{A}\mathbf{M}^{-1/2}$, $\hat{\mathbf{x}} = \mathbf{M}^{1/2}\mathbf{x}$, and $\hat{\mathbf{b}} = \mathbf{M}^{-1/2}\mathbf{b}$. It is easy to see that $\hat{\phi}(\hat{\mathbf{x}}) = \phi(\mathbf{x})$. If $\mathbf{A}$ is chosen so that $\kappa_2(\hat{\mathbf{A}}) < \kappa_2(\mathbf{A})$, the constant in (4) is reduced.

The ideas of the previous section can be used to speed up the convergence of the steepest descent algorithm. We use steepest descent to compute the first column of $\mathbf{A}^{-1}$. As this gives us an approximation to $\mathbf{A}^{-1}$, we can use $\mathbf{N}(\mathbf{c}^{(k-1)})$ instead of $\mathbf{M}^{-1}$. The resulting algorithm is

Algorithm 4 (Dynamically-conditioned steepest descent)

```
Input $\mathbf{c}^{(0)}$
$\mathbf{r}^{(0)} = \mathbf{e}_1 - \mathbf{A}\mathbf{c}^{(0)}$
$k = 1$
While $\|\mathbf{r}^{(k-1)}\|$ is large
        $\mathbf{z}^{(k-1)} = \mathbf{N}(\mathbf{c}^{(k-1)})\mathbf{r}^{(k-1)}$
        $\alpha^{(k)} = \mathbf{z}^{(k-1)t}\mathbf{r}^{(k-1)}/\mathbf{z}^{(k-1)t}\mathbf{A}\mathbf{z}^{(k-1)}$
        $\mathbf{c}^{(k)} = \mathbf{c}^{(k-1)} + \alpha^{(k)}\mathbf{z}^{(k-1)}$
        $\mathbf{r}^{(k)} = \mathbf{r}^{(k-1)} - \alpha^{(k)}\mathbf{A}\mathbf{z}^{(k-1)}$
        $k = k + 1.$
```

As with DII, we define the error at the kth iteration as $\mathbf{s}^{(k)} = \mathbf{c}^{(k)} - \mathbf{c}$. Define the vector $\mathbf{A}$-norm as $\|\mathbf{x}\|_{\mathbf{A}} = \sqrt{\mathbf{x}^t\mathbf{A}\mathbf{x}}$. Then $\mathbf{s}^{(k)}$ and $\mathbf{c}^{(k)}$ are related by

$$\|\mathbf{s}^{(k)}\|_{\mathbf{A}} = \sqrt{\phi(\mathbf{c}^{(k)})}\,, \tag{5}$$

so (4) can be used to bound the $\mathbf{A}$-norm of the error at each step. As with Algorithm 1, DCSD is locally quadratically convergent.

Theorem 3.1. *If* $\mathbf{A}$ *is s.p.d., then* $\exists\delta > 0$ *and* $M < \infty$ *s.t. if* $\|\mathbf{s}^{(0)}\|_{\mathbf{A}} < \delta$ *then* $\|\mathbf{s}^{(k)}\|_{\mathbf{A}} \to 0$ *as*

$$\|\mathbf{s}^{(k+1)}\|_{\mathbf{A}} < M\|\mathbf{s}^{(k)}\|_{\mathbf{A}}^2.$$

Proof summary: By using (1), (4), and (5), it can be shown that

$$\|\mathbf{s}^{(k)}\|_{\mathbf{A}} \leq 4\sqrt{n}\,\tau\|\mathbf{A}\|_2\|\mathbf{A}^{-1/2}\|_1\kappa_2(\mathbf{A})\|\mathbf{s}^{(k-1)}\|_{\mathbf{A}}^2 + O(\|\mathbf{s}^{(k-1)}\|^3) \tag{6}$$

(See [17] for details.) Equation (6) shows that DCSD is locally quadratically convergent. □

If the parameter $\alpha^{(k)}$, taken as

$$\alpha^{(k)} = \mathbf{z}^{(k-1)t}\mathbf{r}^{(k-1)}/\mathbf{z}^{(k-1)t}\mathbf{A}\mathbf{z}^{(k-1)}, \tag{7}$$

is set to "1", then DCSD becomes the same algorithm as DII. If $\mathbf{N}(\mathbf{c}^{(k-1)})$ is a positive definite matrix, then the choice of $\alpha^{(k)}$ given by (7) is optimal, in the sense that this choice yields the smallest $\mathbf{s}^{(k)}$ in the $\mathbf{A}$-norm.

Nonetheless, when comparing the error bounds for DII, (3), and DCSD, (6), it is not clear, in general, which bound is stronger. As these equations are also only upper bounds, we conducted some numerical experiments to compare DCSD and DII.

4. Numerical Experiments

In this section, we present some numerical experiments on random s.p.d. Toeplitz matrices to judge the performance of DII and DCSD.

Denote by **I** the $n \times n$ identity matrix. The first type of matrix that we considered is $\mathbf{A}(\kappa)$. To generate this matrix, we first compute n random numbers, $a_0, \ldots, a_{n-1}$, with a uniform distribution on $[-1, 1)$. We then generate a symmetric Toeplitz matrix $\mathbf{A}$ with first row $(a_0, a_1, \ldots, a_{n-1})$. The matrix $\mathbf{A}(\kappa)$ is defined by

$$\mathbf{A}(\kappa) \equiv \alpha(\mathbf{A} + \beta\mathbf{I}),$$

where α and β are chosen so that $\kappa_2(\mathbf{A}(\kappa)) = \kappa$ and the main diagonal elements of $\mathbf{A}(\kappa)$ are equal to "1".

The second type of matrix that we considered is $\mathbf{B}(\kappa)$. This matrix is similar to $\mathbf{A}(\kappa)$, except that the values of $a_0, \ldots, a_{n-1}$ are chosen randomly on $[0, 1)$.

The experiments were performed as follows. We calculated the exact solution to (2) and a random vector with l_2 norm γ. The random vector was added to the exact solution to obtain the initial approximate solution. The experiments were performed for $\kappa \in \{10, 100, 1000, 10000, 100000\}$ and $\gamma \in \{.001, .003, .1, .3, 1, 3, 10\}$.

Table 1. Number of iterations needed by DII and DCSD for the matrices $\mathbf{A}(\kappa)$. The initial guess is the correct solution plus an error vector with l_2 norm γ

	$\kappa = 10$		$\kappa = 100$		$\kappa = 10^3$		$\kappa = 10^4$		$\kappa = 10^5$	
γ	DCSD	DII	DCSD	DII	DCSD	DII	DCSD	DII	DCSD	DII
.001	2	3	2	3	3	3	2	3	2	2
.003	3	5	3	4	3	4	3	4	3	*
.01	3	*	3	9	4	10	4	6	3	4
.03	4	*	4	*	4	*	4	7	3	4
.1	5	*	5	*	18	*	5	*	7	*
.3	6	*	7	*	6	*	16	*	26	*
1	15	*	15	*	12	*	24	*	42	*
3	18	*	19	*	25	*	33	*	7	*
10	29	*	29	*	25	*	40	*	80	*

* The errors were so large that the algorithm resulted in overflows.

For both $\mathbf{A}(\kappa)$ and $\mathbf{B}(\kappa)$, a new random matrix and random vector were calculated for each combination of κ and γ.

The experiments were performed on matrices of size 8192×8192. The algorithms were allowed to run until either the l_2 norm of the residual was less then 10^{-9} or the errors were so large as to cause overflows.

The results of the experiments on the matrices $\mathbf{A}(\kappa)$ as shown in Table 1. For large γ, DII resulted in overflows, but DCSD never did. When DII converged, it used the same number of iterations as DCSD on two occasions. Otherwise, DII needed more iterations than DCSD to reduce the norm of the residual to 10^{-9}.

The results of the experiments on the matrices $\mathbf{B}(\kappa)$ are shown in Table 2. Here, the performance of DCSD is slightly improved but the performance of DII is much worse. DII converged only once, and then it needed $3\frac{1}{2}$ times as many iterations as DCSD.

Table 2. Number of iterations needed by DII and DCSD for the matrices $\mathbf{B}(\kappa)$. The initial guess is the correct solution plus an error vector with l_2 norm γ

	$\kappa = 10$		$\kappa = 100$		$\kappa = 10^3$		$\kappa = 10^4$		$\kappa = 10^5$	
γ	DCSD	DII	DCSD	DII	DCSD	DII	DCSD	DII	DCSD	DII
.001	2	7	3	*	3	*	3	*	3	*
.003	3	*	3	*	3	*	3	*	3	*
.01	3	*	3	*	4	*	4	*	4	*
.03	3	*	4	*	4	*	4	*	4	*
.1	4	*	5	*	4	*	5	*	9	*
.3	5	*	6	*	9	*	8	*	10	*
1	6	*	16	*	9	*	11	*	14	*
3	6	*	18	*	21	*	27	*	31	*
10	15	*	29	*	31	*	29	*	27	*

* The errors were so large that the algorithm resulted in overflows.

5. Discussion

In the numerical experiments reported on in the last section, DCSD performed better than DII. While in these experiments DCSD never diverged, we cannot prove that DCSD is gloabally convergent so it should therefore be combined with a globally convergent algorithm.

A method for combining DCSD and a globally convergent algorithm can be found in [17]. When this method was used to combine DCSD with the preconditioned conjugate gradient method of Strang [11], the algorithm that resulted performed about as well as Strang's algorithm alone. The reason that the combination did not work better than Strang's algorithm alone is that Strang's algorithm converges slowly during the first few iterations, but as the algorithm progresses the rate of convergence increases [11, 22]. Therefore, by the time the errors are small enough for DCSD to be an effective algorithm, Strang's algorithm can anyway produce a very accurate solution quickly.

To obtain a fast iterative algorithm to solve Toeplitz equations with DCSD, it is desirable to have globally convergent algorithms for Toeplitz systems that produce fairly accurate solutions after a very small number of iterations. Algorithms with this property, as well as their use with DCSD, are described in [17] and will be reported on in a future paper by the first author.

References

[1] Bunch, J. R.: Stability of methods for solving Toeplitz systems of equations. SIAM J. Sci. Stat. Comput. *6*, 349–364 (1985).
[2] Blahut, R. E.: Fast algorithms for digital signal processing. Reading, MA: Addison-Wesley 1986.
[3] Golub, G. H., Van Loan, C. F.: Matrix computations. Baltimore, MD: Johns Hopkins 1983.
[4] Levinson, N.: The Wiener rms error criterion in filter design and prediction. J. Math. Phys. *25*, 261–278 (1947).
[5] Trench, W. F.: An algorithm for the inversion of finite Toeplitz matrices. J. SIAM *12*, 512–522 (1964).
[6] Cybenko, G., Berry, M.: Hyperbolic Housholder algorithm for factoring structured matrices. SIAM J. Matrix Anal. Appl. *11*, 499–520 (1990).
[7] Ammar, G. S., Gragg, W. B.: Superfast solution of real positive definite Toeplitz systems. *9*, 61–76 (1988).
[8] Ammar, G. S., Gragg, W. B.: The generalized Schur algorithm for the superfast solution of Toeplitz systems. In: Pindor, M., Gilewicz, J., Siemaszko, W. (eds.) Rational approximation and its application in mathematics and physics. Springer 1986.
[9] Bitmead, R. R., Anderson, B. D. O.: Asymptotically fast solution of Toeplitz and related systems of linear equations. Linear Algebera Appl. *34*, 103–116 (1980).
[10] Brent, R. P., Gustavson, F. G., Yun, D. Y. Y.: Fast solution of Toeplitz systems equations and computation of Padé approximations. J. Algorithms *1*, 259–295 (1980).
[11] Strang, G.: A proposal for Toeplitz matrix calculations. Stud. Appl. Math. *74*, 171–176 (1986).
[12] Rino, C.: The inversion of covariance matrices by finite Fourier transformations. IEEE Trans. Inform. Theory *16*, 230–232 (1970).
[13] Chan, R. H.: The spectrum of a family of circulant preconditioned Toeplitz systems. SIAM J. Numer. Anal. *26*, 503–506 (1989).
[14] Ku, T., Kuo, J.: Design and analysis of Toeplitz preconditioners. Proc. IEEE Int. Conf. Acoust. Speech Sig. Proc., pp. 1811–1814 (1990).
[15] Pan, V.: Fast and efficient parallel inversion of Toeplitz and block Toeplitz matrices. Operator Theory: Adv. Appl. *40*, 359–389 (1989).
[16] Pan, V., Schrieber, R.: A fast, preconditioned conjugate gradient Toeplitz solver. Technical report 89.14, RIACS, NASA Ames Research Center, March 1989.
[17] Linzer, E.: Arithmetic complexity and numerical properties of algorithms involving Toeplitz matrices. PhD thesis, Columbia University, New York, NY, October 1990.
[18] Strang, G.: Introduction to applied mathematics. Wellesley, MA: Wellesley-Cambridge 1986.
[19] Gohberg, I. C., Fel'dman, I. A.: Convolution equations and projection methods for their solution. Providence, RI: American Mathematical Society 1974.
[20] Iohvidov, I. S.: Hankel and Toeplitz matrices and forms. Boston, MA: Birkhauser 1982.
[21] Cybenko, G.: Error analysis of some signal processing algorithms. Princeton, NJ: PhD thesis, Princeton University 1978.

[22] Chan, R. H., Strang, G.: Toeplitz equations by conjugate gradients with circulant preconditioner. SIAM J. Sci. Stat. Comput. *10*, 104–119 (1989).

E. Linzer
IBM Research
P.O. Box 218
Yorktown Heights, NY 10598

M. Vetterli
Department of Electrical Engineering
and Center for Telecommunications Research
Columbia University
New York
NY 10027