The Generalized Schur Algorithm for the Superfast Solution of Toeplitz Systems *

Gregory S. Ammar
Department of Mathematical Sciences
Northern Illinois University
DeKalb, Illinois 60115

William B. Gragg †
Department of Mathematics
University of Kentucky
Lexington, Kentucky 40506

Abstract

We review the connections between fast, $O(n^2)$, Toeplitz solvers and the classical theory of Szegő polynomials and Schur's algorithm. We then give a concise classically motivated presentation of the superfast, $O(n \log_2^2 n)$, Toeplitz solver that has recently been introduced independently by deHoog and Musicus. In particular, we describe this algorithm in terms of a generalization of Schur's classical algorithm.

1 Introduction

Let $M = [\mu_{j-k}] \in \mathbb{C}^{n \times n}$ be a *Toeplitz matrix.* The problem of solving the system of linear equations $Mx = b$ is important in many areas of pure and applied mathematics: orthogonal polynomials, Padé approximation, signal processing, linear filtering, linear prediction and time series analysis. See, for instance, [1, 3, 15, 20, 21, 23, 24]. There are several *fast*, $O(n^2)$, algorithms

*This manuscript incorporates minor corrections into the paper that appears in Rational Approximation and its Applications in Mathematics and Physics (J. Gilewicz, M. Pindor and W. Siemaszko, editors), Lecture Notes in Mathematics 1237, Springer-Verlag, Berlin, 1987, pp. 315-330.

†Research supported in part by the National Science Foundation under grant DMS–8404980 and by the Seminar für Angewandte Mathematik of the ETH-Zürich.

for solving such systems. This is in contrast with the $O(n^3)$ operations normally used to solve an arbitrary $n \times n$ system, for instance, by Gaussian factorization. Asymptotically *superfast*, $O(n \log_2^2 n)$, algorithms have been proposed for solving such systems [4, 5, 10, 22] but to our knowledge these methods have not yet been implemented.

In this paper we give a classically motivated presentation of the algorithm that has recently been independently presented by deHoog [10] and Musicus [22] in the case where the Toeplitz matrix M is (Hermitian) positive definite. Our treatment is based on the relations among positive definite Toeplitz matrices, Szegő polynomials and Schur's algorithm [25]. In particular, the deHoog-Musicus algorithm is naturally explained in terms of a generalization of Schur's algorithm. An analogous treatment of the positive definite *Hankel* case, $M = [\mu_{j+k}] = M^* > 0$, generalizing the algorithm of Chebyshev [7, 12], is given in [18].

In Section 2 we review the classical foundations of fast Toeplitz solvers. We present the generalized Schur algorithm in Section 3 and describe the use of the algorithm for the superfast solution of a positive definite Toeplitz system in Section 4.

Before proceeding, we note that the restriction to positive definite Toeplitz systems is not as severe as it may seem. First, the positive definite case is of primary interest in most important applications, including discrete time Wiener filtering, autocorrelation problems, and Gaussian quadrature on the unit circle [16, 17]. Second, it is clear that most fast and superfast Toeplitz solvers are numerically unstable, and therefore unreliable, when applied to an arbitrary Toeplitz systems: [8, 9, 6]. In this connection we note that the algorithm of [5, 18], while potentially stable for positive definite *Hankel* systems, is *manifestly unstable* for positive definite Toeplitz systems. However, Cybenko has shown that the algorithms of Levinson, Durbin, and Trench are numerically stable for the class of positive definite Toeplitz matrices, and stability for this class can be expected in some superfast algorithms [6].

The implementation of the generalized Schur algorithm and the superfast (positive definite) Toeplitz solver of deHoog and Musicus is described in [2].

2 The Classical Foundations of Fast Toeplitz Solvers

2.1 Positive definite matrices and orthogonal polynomials.

Every complex (Hermitian) positive definite matrix $M = [\mu_{j,k}]$ can be factored uniquely as

$$M = LDL^*, \tag{2.1}$$

with $L = [\lambda_{j,k}]$ unit left triangular ($\lambda_{k,k} \equiv 1$) and $D = \mathrm{diag}[\delta_k]$ positive definite. The matrix $\hat{L} := LD^{1/2}$ is the *left Cholesky factor* of M. Equivalently,

$$R^*MR = D \tag{2.2}$$

and

$$M^{-1} = RD^{-1}R^*, \tag{2.3}$$

with $R = [\rho_{j,k}] = L^{-*}$ unit right triangular. By abuse of terminology, we call (2.1) the *Cholesky factorization* of M, and (2.2) or (2.3) the *inverse Cholesky factorization* of M. (Actually, (2.3) is the *reverse* Cholesky factorization of M^{-1}.) If either of these factorizations is known then the linear system $Mx = b$ can be solved directly with at most n^2 multiplicative and additive operations (n^2 *flops*). Both factorizations are represented by the formula

$$T := [\tau_{j,k}] := MR = LD. \tag{2.4}$$

In special cases, most notably when M is a *Hankel matrix*, $M = [\mu_{j+k}]$, or a *Toeplitz matrix*, $M = [\mu_{j-k}]$, it follows from *classical analysis* that these factorizations can be computed in $\mathrm{O}(n^2)$ operations. We thus obtain *fast*, $\mathrm{O}(n^2)$, algorithms for solving $Mx = b$. In the Hankel case this point is moot: positive definite Hankel matrices are notoriously severely ill-conditioned (e.g., the *Hilbert matrix* with $\mu_k = \int_0^1 \lambda^k d\lambda = 1/(k+1)$). The situation can be quite different for positive definite Toeplitz matrices (e.g., $M = I$, the identity matrix).

We now describe the *orthogonal polynomials* associated with M. It is natural to number the indicies j and k from zero, and put

$$M := M_{n+1} := [\mu_{j,k}]_{j,k=0}^{n}.$$

More generally, for later use,

$$M_k := [\mu_{j,l}]_{j,l=0}^{k-1} \in \mathbb{C}^{k\times k} \qquad (0 \le k-1 \le n)$$

is the *kth section* of M, and likewise for R_k and D_k. The positive definite matrix M determines an inner product $< \cdot, \cdot >$ for the complex vector space $\mathbb{C}_n[\lambda]$ of polynomials of degree at most n, on setting

$$< \lambda^j, \lambda^k >:= \mu_{j,k} \qquad (0 \leq j \leq n, 0 \leq k \leq n)$$

and extending $< \cdot, \cdot >$ to all of $(\mathbb{C}_n[\lambda])^2$ by requiring that it be linear in its second argument and conjugate linear in its first argument—like the Euclidean inner product y^*x for the complex vector space $\mathbb{C}^n$ of (column) n-vectors.

Now (2.2) states that the monic polynomials $\{\psi_k\}_0^n$ defined by

$$\psi_k(\lambda) := \sum_j \rho_{j,k}\lambda^j \tag{2.5}$$

are orthogonal with respect to $< \cdot, \cdot >$:

$$< \psi_j, \psi_k >= \left\{ \begin{array}{ll} 0 & j \neq k \\ \delta_k, & j = k \end{array} \right. .$$

Thus, $\{\psi_k\}_0^n$ is an orthogonal basis for $\mathbb{C}_n[\lambda]$, and the $(k+1)$th column of R contains the coefficients of the representation of ψ_k in terms of the standard basis $\{\lambda^j\}_0^n$ for $\mathbb{C}_n[\lambda]$. The columns of $\hat{R} := \hat{L}^{-*} = RD^{-1/2}$ likewise *generate* the ortho*normal* polynomials $\{\hat{\psi}_k\}_0^n$: $\hat{\psi}_k := \psi_k/\delta_k^{1/2}$. Finally, we see from (2.4) that

$$\begin{aligned} \tau_{j,k} &= \sum_l \mu_{j,l}\rho_{l,k} \\ &= \sum_l < \lambda^j, \lambda^l > \rho_{l,k} \\ &= < \lambda^j, \psi_k >= \lambda_{j,k}\delta_k \\ &= \left\{ \begin{array}{ll} 0, & j < k \\ \delta_k, & j = k \end{array} \right. . \end{aligned} \tag{2.6}$$

2.2 Inverse Cholesky Factorization and Szegő Polynomials.

We henceforth assume that $M = M^* = [\mu_{j-k}]_{j,k=0}^n > 0$ is a *Toeplitz matrix.* The orthogonal polynomials $\{\psi_k\}_0^n$ are then called the *Szegő polynomials* associated with M. They satisfy the *Szegő recurrence relation*

$$\psi_{k+1} = \lambda\psi_k + \gamma_{k+1}\tilde{\psi}_k, \tag{2.7a}$$

with $\tilde{\psi}_k(\lambda) = \lambda^k \overline{\psi}_k(1/\lambda)$ the polynomial obtained from ψ_k by conjugating and reversing the order of the coefficients,

$$\gamma_{k+1} = - < 1, \lambda\psi_k > /\delta_k, \tag{2.7b}$$

and

$$\delta_{k+1} = \delta_k(1 - |\gamma_{k+1}|^2). \tag{2.7c}$$

The numbers $\{\gamma_k\}_1^n$ are the *Schur parameters* associated with M. They determine the Szegő polynomials by (2.7a); note also that $\gamma_k \equiv \psi_k(0)$. From (2.7c) we see that $|\gamma_k| < 1$ $(1 \le k \le n)$. The Schur parameters are referred to as *reflection coefficients* in the engineering literature, and as *partial correlation coefficients* in prediction theory.

Although we shall not use the result in this paper, it is known that there is a bounded nondecreasing function $m(\theta)$ with

$$< \beta, \alpha >= \frac{1}{2\pi} \int_0^{2\pi} \beta(\lambda)^* \alpha(\lambda) dm(\theta), \qquad \lambda = e^{i\theta}.$$

See, for instance, [1]. The Szegő polynomials are thus "orthogonal on the unit circle."

Formulas (2.7b) and (2.7c) follow rather directly from (2.7a), on using the orthogonality and the *isometry relation* $< \beta, \alpha > \equiv < \beta/\lambda, \alpha/\lambda >$, valid for $\alpha, \beta \in \mathbb{C}_n[\lambda]$ with $\alpha(0) = \beta(0) = 0$. However, a matrix theoretic proof of *all* of (2.7a–2.7c), based on the persymmetry of M, seems more efficient. The matrix $A \in \mathbb{C}^{n\times n}$ is *persymmetric* if it is invariant under reflection in its antidiagonal. This means that $A = A^P := JA^TJ$, where $J := J_n$ is the $n \times n$ *reversal matrix* (obtained by reversing the columns of the $n \times n$ identity matrix).

Put

$$M_{k+1} =: \begin{bmatrix} M_k & \tilde{m}_k \\ \tilde{m}_k^* & \mu_0 \end{bmatrix} =: \begin{bmatrix} \mu_0 & m_k^* \\ m_k & M_k \end{bmatrix}$$

so that

$$m_k := [\mu_1, \mu_2, \ldots, \mu_k]^T$$

and

$$\tilde{m}_k := J_k \overline{m}_k.$$

Also put

$$R_{k+1} =: \begin{bmatrix} R_k & r_k \\ & 1 \end{bmatrix}, \qquad r_{k+1} =: \begin{bmatrix} \gamma_{k+1} \\ s_k \end{bmatrix}$$

and

$$\tilde{r}_k := J_k \overline{r}_k.$$

Equating last columns in $M_{k+1}R_{k+1} = L_{k+1}D_{k+1}$ gives

$$M_k r_k + \tilde{m}_k = 0, \tag{2.8}$$

$$\tilde{m}_k^* r_k + \mu_0 = \delta_k. \tag{2.9}$$

Now

$$M_k = J_k M_k^T J_k = J_k \overline{M}_k J_k$$

is persymmetric and Hermitian. Hence (2.8) is equivalent with the *Yule-Walker equation*

$$m_k + M_k \tilde{r}_k = 0. \tag{2.10}$$

Now increase k by unity in (2.8) and use the second partitioning of M_{k+1} to get

$$\mu_0 \gamma_{k+1} + m_k^* s_k + \mu_{k+1}^* = 0, \tag{2.11}$$

$$m_k \gamma_{k+1} + M_k s_k + \tilde{m}_k = 0. \tag{2.12}$$

Subtracting (2.8) from (2.12), and using (2.10), we obtain

$$s_k = r_k + \tilde{r}_k \gamma_{k+1}, \tag{2.13}$$

that is,

$$r_{k+1} = \begin{bmatrix} 0 \\ r_k \end{bmatrix} + \begin{bmatrix} 1 \\ \tilde{r}_k \end{bmatrix} \gamma_{k+1}. \tag{2.14a}$$

This is the Szegő recurrence relation (2.7a). Using (2.13) in (2.11) we find, on account of (2.9), that

$$\begin{aligned} \delta_k \gamma_{k+1} &= -m_{k+1}^* \begin{bmatrix} r_k \\ 1 \end{bmatrix} \\ &= -\sum_j \mu_{-j-1} \rho_{j,k} \\ &= -< 1, \lambda \psi_k > . \end{aligned} \tag{2.14b}$$

Finally, by (2.9), (2.14a) and (2.14b),

$$\begin{aligned} \delta_{k+1} &= \mu_0 + \tilde{m}_{k+1}^T \overline{r}_{k+1} \\ &= \mu_0 + \tilde{m}_k^T \overline{r}_k + m_{k+1}^* \begin{bmatrix} r_k \\ 1 \end{bmatrix} \gamma_{k+1}^* \\ &= \delta_k (1 - |\gamma_{k+1}|^2). \end{aligned} \tag{2.14c}$$

Hence the Szegő recurrence relations (2.7a–2.7c) are equivalent with the matrix formulation (2.14a–2.14c) which, in the context of Toeplitz systems, is known as the *Levinson-Durbin algorithm* [21, 11, 14].

To solve the system $Mx = b$ one puts

$$M_k x_k := b_k, \quad b_{k+1} := \begin{bmatrix} b_k \\ \beta_k \end{bmatrix}, \quad b_{n+1} := b,$$

and finds from $x_{k+1} = M_{k+1}^{-1} b_{k+1}$ that

$$x_{k+1} = \begin{bmatrix} x_k \\ 0 \end{bmatrix} + \begin{bmatrix} r_k \\ 1 \end{bmatrix} \xi_{k+1}$$

with

$$\xi_{k+1} = [r_k^*, 1] b_{k+1} / \delta_k.$$

This algorithm applies *in general* to solve $Mx = b$ when the inverse Cholesky factorization of M is known; that is it makes no use of the Toeplitz structure of M.

The work for this two-stage algorithm to solve $M_n x = b$ is about $2n^2$ flops, at most n^2 flops for each stage.

2.3 The Christoffel-Darboux-Szegő and Gohberg-Semencul Formulas.

From (2.3) we see that the generating polynomial of $M^{-1} =: [\beta_{j,k}]$ is

$$\begin{aligned} \kappa_n(\lambda, \tau) &:= \sum \beta_{j,k} \lambda^j \tau^{k*} \\ &= \sum_{k=0}^{n} \hat{\psi}_k(\lambda) \hat{\psi}_k(\tau)^*. \end{aligned}$$

It is possible to express $\kappa_n(\lambda, \tau)$ solely in terms of the normalized polynomial $\hat{\psi}_n$. The unnormalized form of this result is

$$\delta_n (1 - \lambda \tau^*) \kappa_n(\lambda, \tau) = \tilde{\psi}_n(\lambda) \tilde{\psi}_n(\tau)^* - \lambda \tau^* \psi_n(\lambda) \psi_n(\tau)^*.$$

This is *Szegő's formula* [26]. It is the analog for Szegő polynomials of the *Christoffel-Darboux formula* for polynomials orthogonal on the real line. Its inductive verification reduces to

$$\tilde{\psi}_n(\lambda) \tilde{\psi}_n(\tau)^* = \psi_n(\lambda) \psi_n(\tau)^* + (1 - |\gamma_n|^2) [\tilde{\psi}_{n-1}(\lambda) \tilde{\psi}_{n-1}(\tau)^* - \lambda \tau^* \psi_{n-1}(\lambda) \psi_{n-1}(\tau)^*],$$

which in turn is a direct consequence of (2.7a) and its equivalent:

$$\tilde{\psi}_{k+1} = \tilde{\psi}_k + \gamma^*_{k+1}\lambda\psi_k.$$

Setting now

$$\psi_n(\lambda) =: \sum \rho_k \lambda^k, \qquad \rho_k := \rho_{k,n},$$

we see that

$$\delta_n(\beta_{j,k} - \beta_{j-1,k-1}) = \rho^*_{n-j}\rho_{n-k} - \rho_{j-1}\rho^*_{k-1},$$

where elements with negative subscripts are zero. Thus

$$\delta_n\beta_{j,k} = \sum_l (\rho^*_{n+l-j}\rho_{n+l-k} - \rho_{j-l-1}\rho^*_{k-l-1})$$

and so

$$\delta_n M^{-1} = T_1^* T_1 - T_0 T_0^*$$

with *Toeplitz matrices*

$$T_0 := [\rho_{j-k-1}]^n_{j,k=0}$$

and

$$T_1 := [\rho_{n+j-k}]^n_{j,k=0}.$$

This is the *Gohberg-Semencul formula* [13]. Note that T_0 is *strictly* left triangular ($\rho_{-1} := 0$) and T_1 is unit right triangular ($\rho_n = \rho_{n,n} = 1$).

2.4 Cholesky Factorization and Schur's Algorithm.

We have seen that the Szegő recursions can be used to solve a positive definite Toepltiz system using the inverse Cholesky factorization as well as the Gohberg-Semencul formula. We now describe an algorithm for finding the Cholesky factors L_n and D_n of M_n (also see [22]). We will see that this algorithm is a manifestation of the classical algorithm of Schur. The algorithm presented below is in direct analogy with the derivation of Chebyshev's algorithm [7] for positive definite Hankel matrices, as presented, for instance, by Gautschi [12].

Extend the functional $< \cdot, \cdot >$ to *certain* pairs of Laurent polynomials by putting

$$< \lambda^j, \lambda^k > := \mu_{j-k}, |j-k| \leq n.$$

Then

$$\tau_{j,k} := < \lambda^j, \psi_k >$$

and

$$\tilde{\tau}_{j,k} :=< \lambda^j, \tilde{\psi}_k >$$

are defined for $0 \leq k \leq n$ and $-n+k < j < n$. In fact

$$\tau_{j,k} = \begin{cases} 0, & 0 \leq j < k \\ \delta_k, & j = k \\ \mu_j, & k = 0 \end{cases}$$

and moreover,

$$\begin{aligned} \tilde{\tau}_{j,k} &= \sum_{i=0}^{k} \rho_{i,k}^* < \lambda^j, \lambda^{k-i} > \\ &= (\sum_{i=0}^{k} \rho_{i,k} \mu_{k-i-j})^* \\ &= (\sum_{i=0}^{k} \rho_{i,k} < \lambda^{k-j}, \lambda^i >)^* \\ &= < \lambda^{k-j}, \psi_k >^* = \tau_{k-j,k}^*. \end{aligned}$$

Now the Szegő recursion gives

$$\begin{aligned} \tau_{j,k} &= < \lambda^j, \lambda \psi_{k-1} > + \gamma_k < \lambda^j, \tilde{\psi}_{k-1} > \\ &= < \lambda^{j-1}, \psi_{k-1} > + \gamma_k \tilde{\tau}_{j,k-1} \\ &= \tau_{j-1,k-1} + \gamma_k \tau_{k-j-1,k-1}^*, \end{aligned}$$

and letting $j = 0$ we obtain $\gamma_k = -\tau_{-1,k-1}/\delta_{k-1}$.

Since $LD = T := [\tau_{j,k}]_{j,k=0}^n$, the following algorithm can be used to obtain the Cholesky factorization of a positive definite Toeplitz matrix.

Algorithm 2.2. (Fast Cholesky Factorization).
input: $M = [\mu_{j-k}]_{j,k=0}^n > 0$,
$\tau_{0,0} = \mu_0$,
for $j = 1, 2, \ldots, n$
  $\tau_{j,0} = \mu_j, \quad \tau_{-j,0} = \mu_j^*$,
  for $k = 1, 2, \ldots, j-1$
    $\tau_{j,k} = \tau_{j-1,k-1} + \gamma_k \tau_{k-j-1,k-1}^*$,
    $\tau_{k-j,k} = \tau_{k-j-1,k-1} + \gamma_k \tau_{j-1,k-1}^*$,
  $\gamma_j = -\tau_{-1,j-1}/\tau_{j-1,j-1}$
  $\tau_{j,j} = \tau_{j-1,j-1}(1 - |\gamma_j|^2)$

We now describe the classical algorithm of Schur [25]. The fast Cholesky algorithm will then be shown to be equivalent with Schur's algorithm.

Let $D = \{\lambda : |\lambda| < 1\}$ be the open unit disk in the complex plane. A *Schur function* ϕ is holomorphic on D with $\phi(D)$ contained in the closure $\overline{D}$ of D. Schur's algorithm is a procedure that generates a (possibly finite) sequence ϕ_n of Schur functions by transforming an initial Schur function ϕ_0 by successive linear fractional transformations (LFT's). In this way a (formal) continued fraction representation of ϕ_0 is obtained.

Let $\phi = \phi_0$ be a Schur function, and put $\gamma = \gamma_1 := \phi_0(0)$. Note that $|\gamma| \leq 1$, and moreover, $|\gamma| = 1$ implies $\phi(\lambda) \equiv \gamma$ (by the maximum principle). In case $|\gamma| = 1$, Schur's algorithm terminates; if $|\gamma| < 1$ define

$$\phi_1 := \frac{1}{\lambda}\frac{\phi_0 - \gamma}{1 - \gamma^* \phi_0}.$$

To see that ϕ_1 is then a Schur function, note that the Möbius transformation $(\gamma + \tau)/(1 + \gamma^* \tau)$ maps D onto D with inverse function $(\tau - \gamma)/(1 - \gamma^* \tau)$. The function $(\phi - \gamma)/(1 - \gamma^* \phi)$ is therefore a Schur function that vanishes at $\lambda = 0$, so by Schwarz' Lemma, ϕ_1 is also a Schur function. This construction represents one step of Schur's algorithm.

Schur's Algorithm.

input: an initial Schur function ϕ_0,

$\gamma_1 = \phi_0(0)$,

for $n = 1, 2, 3, \ldots$ **while** $|\gamma_n| < 1$

$$\left\lfloor \begin{array}{l} \phi_n(\lambda) = \dfrac{1}{\lambda}\dfrac{\phi_{n-1}(\lambda) - \gamma_n}{1 - \gamma_n^* \phi_{n-1}(\lambda)}, \\ \gamma_{n+1} = \phi_n(0). \end{array} \right.$$

Thus, $\phi_{n-1} = t_n(\phi_n)$, where

$$t_n(\tau) := \frac{\gamma_n + \lambda\tau}{1 + \gamma_n^* \lambda\tau},$$

and in general, $\phi_0 = T_n(\phi_n)$ where $T_n = t_1 \circ t_2 \circ \ldots \circ t_n$. This composition of linear fractional transformations is n steps in the *Schur continued fraction* representation of ϕ_0. We refer to $T_n(0)$ and ϕ_n, respectively, as the *nth approximant* and *nth tail* of ϕ_0. The numbers γ_n are the *Schur parameters* associated with ϕ_0.

We can therefore view Schur's algorithm as producing a sequence of Schur functions ϕ_n, or equivalently, as producing a sequence of LFT's T_n.

In Section 3 we will explicitly formulate Schur's algorithm in terms of the T_n.

In order to implement Schur's algorithm, we write the functions as quotients of power series,

$$\phi_n =: \frac{\alpha_n(\lambda)}{\beta_n(\lambda)} = \frac{\sum_k \alpha_{n,k}\lambda^k}{\sum_k \beta_{n,k}\lambda^k}.$$

Schur's algorithm as formulated above involves an infinite sequence of elementary operations on infinite power series α_n and β_n. However, it is not difficult to arrange the computations so that the coefficient pairs $(\alpha_{0,n}, \beta_{0,n})$ are entered and processed in a *sequential manner.*

Algorithm 2.3 (progressive Schur algorithm).

input: the coefficients $\{\alpha_n\}_{n=0}^{\infty}$, $\{\beta_n\}_{n=0}^{\infty}$ of formal power series α and β such that $\phi_0 = \alpha/\beta$ is a Schur function.

for $n = 1, 2, 3, \ldots,$ **while** $|\gamma_n| < 1$

$\alpha_{0,n-1} = \alpha_{n-1}, \quad \beta_{0,n-1} = \beta_{n-1},$

for $k = 1, 2, 3, \ldots, n-1$

$$\begin{bmatrix} \alpha_{k,n-k-1} \\ \beta_{k,n-k} \end{bmatrix} = \begin{bmatrix} 1 & -\gamma_k \\ -\gamma_k^* & 1 \end{bmatrix} \begin{bmatrix} \alpha_{k-1,n-k} \\ \beta_{k-1,n-k} \end{bmatrix}$$

$\gamma_n = \alpha_{n-1,0}/\beta_{n-1,0},$

$\sigma_n^2 = (1 - |\gamma_n|^2),$

$\beta_{n,0} = \beta_{n-1,0}\sigma_n^2.$

Of course, in practice the first n coefficients of α_0 and β_0 will be input , and Schur's algorithm will be performed to obtain the first $n-k$ coefficients of the power series α_k and β_k (and the Schur parameter γ_k) for $k = 1, \ldots, n$.

The following proposition provides the connection between Schur's algorithm and the fast Cholesky algorithm.

Proposition 2.1 *Under the identifications $\alpha_{j,k} := -\tau_{-k-1,j}$, and $\beta_{j,k} := \tau_{j+k,j}^*$, the fast Cholesky algorithm and the progressive Schur algorithm are equivalent. In other words, Algorithm 2.2 applied to the Hermitian matrix $M_n = [\mu_{j-k}]_{j,k=0}^n$ is equivalent with n steps of Algorithm 2.3 applied to a function $\phi_0 = \alpha_0/\beta_0$ with the first n coefficients of α_0 and β_0 given by $[-\tau_{-k-1,0}]_0^{n-1}$ and $[\tau_{k,0}^*]_0^{n-1}$, respectively.*

The proof of this result follows immediately from the recursions. In particular,

$$\begin{bmatrix} \tau_{k-j,k} \\ \tau_{j,k}^* \end{bmatrix} = \begin{bmatrix} 1 & -\gamma_k \\ -\gamma_k^* & 1 \end{bmatrix} \begin{bmatrix} \tau_{k-j-1,k-1} \\ \tau_{j-1,k-1}^* \end{bmatrix}$$

for $j \geq k$. The equivalence of the two recursions implies the numerator and denominator of ϕ_0 are the first n terms in formal power series α_0 and β_0 such that α_0/β_0 is a Schur function. Thus, the computational procedure for finding the Cholesky factorization of a positive definite Toeplitz matrix is equivalent with Schur's classical algorithm.

3 The Generalized Schur Algorithm.

3.1 Schur Polynomials.

Schur's algorithm was described in the previous section as an iteration that generates the numerators and denominators of the Schur functions ϕ_n. We now reformulate Schur's algorithm in terms of the LFT's T_n defined by $\phi_0 = T_n(\phi_n)$.

Let ξ_n, η_n, $\tilde{\xi}_n$, $\tilde{\eta}_n$ be the polynomials such that

$$T_n(\tau) = \frac{\xi_n(\lambda) + \tilde{\eta}_n(\lambda)\tau}{\eta_n(\lambda) + \tilde{\xi}_n(\lambda)\tau}.$$

Since $T_0(\tau) = \tau$ and $T_n(\tau) = T_{n-1}(t_n(\tau))$ (where $t_n(\tau) = \dfrac{\gamma_n + \lambda\tau}{1 + \gamma_n^* \lambda\tau}$), we have the recurrence relations

$$\begin{bmatrix} \tilde{\eta}_n & \xi_n \\ \tilde{\xi}_n & \eta_n \end{bmatrix} = \begin{bmatrix} \tilde{\eta}_{n-1} & \xi_{n-1} \\ \tilde{\xi}_{n-1} & \eta_{n-1} \end{bmatrix} \begin{bmatrix} \lambda & \gamma_n \\ \gamma_n^*\lambda & 1 \end{bmatrix}, \qquad \begin{bmatrix} \tilde{\eta}_0 & \xi_0 \\ \tilde{\xi}_0 & \eta_0 \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}.$$

The results of the following proposition are easily obtained by induction.

Proposition 3.1 *The polynomials $\tilde{\eta}_n$ and $\tilde{\xi}_n$ satisfy*

$$\tilde{\xi}_n(\lambda) \equiv \lambda^n \overline{\xi}_n(1/\lambda), \qquad \tilde{\eta}_n(\lambda) \equiv \lambda^n \overline{\eta}_n(1/\lambda).$$

Furthermore, for all $n \geq 1$,

$$\deg(\xi_n) < n, \qquad \deg(\eta_n) < n,$$

$$\xi_n(0) = \gamma_1, \qquad \eta_n(0) \equiv 1.$$

We also have the determinant formula

$$\eta_n\tilde{\eta}_n - \xi_n\tilde{\xi}_n = \delta_n\lambda^n,$$

where $\delta_n = \sigma_1^2\sigma_2^2\cdots\sigma_n^2$.

We refer to ξ_n and η_n as the nth *Schur polynomials* associated with the Schur function ϕ_0.

Thus, we may view Schur's algorithm as generating a sequence of Schur functions, or equivalently, as generating a sequence of Schur polynomials which determine the LFT T_n. We now show that if the Schur function ϕ_0 is given as in Proposition 2.1 (where M is a positive definite Toeplitz matrix), then the Szegő polynomials are determined by the Schur polynomials. This will allow us to use Schur's algorithm to obtain the Gohberg-Semencul factorization of M^{-1}.

By transposing the recursions for the Schur polynomials and applying a diagonal scaling, we obtain

$$\begin{bmatrix} \tilde{\eta}_n & \tilde{\xi}_n/\lambda \\ \lambda\xi_n & \eta_n \end{bmatrix} = \begin{bmatrix} \lambda & \gamma_n^* \\ \lambda\gamma_n & 1 \end{bmatrix} \begin{bmatrix} \tilde{\eta}_{n-1} & \tilde{\xi}_{n-1}/\lambda \\ \lambda\xi_{n-1} & \eta_{n-1} \end{bmatrix}.$$

By the Szegő recurrence formula, we obtain

$$\begin{bmatrix} \psi_n \\ \tilde{\psi}_n \end{bmatrix} = \begin{bmatrix} \lambda & \gamma_n \\ \lambda\gamma_n^* & 1 \end{bmatrix} \begin{bmatrix} \psi_{n-1} \\ \tilde{\psi}_{n-1} \end{bmatrix}, \qquad \text{or} \qquad \overline{\begin{bmatrix} \psi_n \\ \tilde{\psi}_n \end{bmatrix}} = \begin{bmatrix} \lambda & \gamma_n^* \\ \lambda\gamma_n & 1 \end{bmatrix} \overline{\begin{bmatrix} \psi_{n-1} \\ \tilde{\psi}_{n-1} \end{bmatrix}}.$$

From these recursions, together with the initial conditions

$$\begin{bmatrix} \tilde{\eta}_1 & \tilde{\xi}_1/\lambda \\ \lambda\xi_1 & \eta_1 \end{bmatrix} = \begin{bmatrix} 1 & \gamma_1^* \\ \lambda\gamma_1 & 1 \end{bmatrix}, \qquad \overline{\begin{bmatrix} \psi_1 \\ \tilde{\psi}_1 \end{bmatrix}} \begin{bmatrix} 1 & \gamma_1^* \\ \lambda\gamma_1 & 1 \end{bmatrix} \begin{bmatrix} 1 \\ 1 \end{bmatrix},$$

we see that

$$\overline{\psi}_n = \tilde{\eta}_n + \tilde{\xi}_n/\lambda. \tag{3.1}$$

Thus, Schur's algorithm can be used to construct the Szegő polynomials.

3.2 The Generalized Schur Algorithm.

As we have seen, Schur's classical algorithm generates a sequence of linear fractional transformations T_n. We now present a generalization of Schur's algorithm that allows us to generate T_{n+k} from T_n, where $k > 1$. The following simple lemma is needed in our derivation.

Lemma 3.1 *Let $\phi_0 = \alpha_0/\beta_0$ be a Schur function, and let ξ_n and η_n be the nth Schur polynomials for ϕ_0. Then*

$$\alpha_0\eta_n - \beta_0\xi_n = \beta_{0,0}\delta_n\gamma_{n+1}\lambda^n + O(\lambda^{n+1})$$
$$\beta_0\tilde{\eta}_n - \alpha_0\tilde{\xi}_n = \beta_{0,0}\delta_n\lambda^n + O(\lambda^{n+1}).$$

Proof: Since $T_n(\tau) = \dfrac{\xi_n + \tilde{\eta}_n\tau}{\eta_n + \tilde{\xi}_n\tau}$, we have by the determinant formula

$$\frac{T_n(\tau_0) - T_n(\tau)}{\tau_0 - \tau} = \frac{\delta_n\lambda^n}{(\eta_n + \tilde{\xi}_n\tau_0)(\eta_n + \tilde{\xi}_n\tau)}.$$

Setting $\tau_0 = \phi_n$ we get

$$\phi - T_n(\tau) = \frac{\delta_n(\phi_n - \tau)\lambda^n}{(\eta_n + \phi_n\tilde{\xi}_n)(\eta_n + \tilde{\xi}_n\tau)}.$$

Now let $\tau = 0$, and note that $\xi_n/\eta_n = T_n(0)$, $\eta_n(0) = 1$ and $\tilde{\xi}_n(0) = 0$ to obtain

$$\phi\eta_n - \xi_n = \frac{\delta_n\phi_n\lambda^n}{\eta_n + \phi_n\tilde{\xi}_n} = \gamma_{n+1}\delta_n\lambda^n + O(\lambda^{n+1})$$

as $\lambda \to 0$. Similarly, setting $\tau = \infty$ we obtain

$$\tilde{\eta}_n - \phi\tilde{\xi}_n = \frac{\delta_n\lambda^n}{\eta_n + \phi_n\tilde{\xi}_n} = \delta_n\lambda^n + O(\lambda^{n+1})$$

as $\lambda \to 0$. This completes the proof.

Let $\phi_0 = \phi$ be a Schur function, and let ϕ_n be the nth tail of ϕ (i.e., ϕ_n is the result of n steps of Schur's algorithm starting at ϕ_0). Also let $\xi_{0,n} = \xi_n$ and $\eta_{0,n} = \eta_n$ be the nth Schur polynomials of ϕ_0, so that $\phi_0 = T_{0,n}(\phi_n)$, where

$$T_{0,n}(\tau) = \frac{\xi_{0,n} + \tilde{\eta}_{0,n}\tau}{\eta_{0,n} + \tilde{\xi}_{0,n}\tau}$$

In order to construct $T_{0,n+k} = T_{n+k}$ we must first obtain ϕ_n from $T_{0,n}$ and ϕ_0. We have

$$\phi_n = \frac{\alpha_n}{\beta_n} = T_{0,n}^{-1}(\frac{\alpha_0}{\beta_0}) = \frac{\alpha_0\eta_{0,n} - \beta_0\xi_{0,n}}{\beta_0\tilde{\eta}_{0,n} - \alpha_0\tilde{\xi}_{0,n}}. \tag{3.2}$$

By Lemma 3.1 both the numerator and denominator in (3.2) are divisible by λ^n. It is therefore natural to take

$$\alpha_n = (\alpha_0\eta_{0,n} - \beta_0\xi_{0,n})/\lambda^n, \qquad \beta_n = (\beta_0\tilde{\eta}_{0,n} - \alpha_0\tilde{\xi}_{0,n})/\lambda^n. \tag{3.3}$$

Thus, formula (3.3) enables us to obtain the n-th tail ϕ_n of ϕ_0 from ϕ_0 and $T_{0,n}$.

Since ϕ_n is a Schur function, we can obtain $T_{n,k}$, the LFT that results from k steps of Schur's algorithm applied to ϕ_n. We then have $\phi_n =$

$T_{n,k}(\phi_{n+k})$, (i.e., the kth tail of ϕ_n is equal to the $(n+k)$th tail of ϕ_0). Once we have $T_{n,k}$, we can construct $T_{0,n+k}$ by simply composing the LFT's. In particular,

$$\xi_{0,n+k} = \tilde{\eta}_{0,n}\xi_{n,k} + \xi_{0,n}\eta_{n,k}, \qquad \eta_{0,n+k} = \tilde{\xi}_{0,n}\xi_{n,k} + \eta_{0,n}\eta_{n,k} \tag{3.4}$$

The generalized Schur algorithm is a doubling procedure based on the recursions (3.3) and (3.4) that generates T_n for $n = 1, 2, 4, \ldots, 2^p, \ldots$. As in the case of the classical Schur algorithm, the computations are to be organized so that the coefficients of the formal power series α_0 and β_0 enter in a sequential fashion. However, instead of entering one at a time, the coefficients enter in groups, each group being twice as large as the previous one.

For the formal power series $\alpha_n = \sum_{j=0}^{\infty} \alpha_{n,j}\lambda^j$, let $\alpha_n^{(k)}$ denote the polynomial $\sum_{j=0}^{k-1} \alpha_{n,j}\lambda^j$ of degree less than k, and define $\beta_n^{(k)}$ similarly. We can describe the algorithm as follows.

Generalized Schur Algorithm.

input: $\alpha_0^{(2^p)}$ and $\beta_0^{(2^p)}$, where $\phi_0 = \alpha_0/\beta_0$ is a Schur function,
$\xi_{0,1} = \gamma_1 = \alpha_0^{(1)}/\beta_0^{(1)}$, $\eta_{0,1} = 1$,
for $n = 1, 2, 4, \ldots, 2^{p-1}$

1: Compute $\alpha_n^{(n)}, \beta_n^{(n)}$, which are respectively given by the first n coefficients of the polynomials

$$(\alpha_0^{(2n)}\eta_{0,n} - \beta_0^{(2n)}\xi_{0,n})/\lambda^n,$$
$$(\beta_0^{(2n)}\tilde{\eta}_{0,n} - \alpha_0^{(2n)}\tilde{\xi}_{0,n})/\lambda^n.$$

2: Compute $\xi_{n,n}$ and $\eta_{n,n}$ from $\alpha_n^{(n)}$ and $\beta_n^{(n)}$ as $\xi_{0,n}$ and $\eta_{0,n}$ were obtained from $\alpha_0^{(n)}$ and $\beta_0^{(n)}$. (This is the doubling step.)

3: Compute

$$\begin{aligned} \xi_{0,2n} &= \tilde{\eta}_{0,n}\xi_{n,n} + \xi_{0,n}\eta_{n,n}, \\ \eta_{0,2n} &= \tilde{\xi}_{0,n}\xi_{n,n} + \eta_{0,n}\eta_{n,n}. \end{aligned}$$

Recall that in the progressive Schur algorithm, the input polynomials $\alpha_0^{(n)}$ and $\beta_0^{(n)}$ determine $\alpha_k^{(n-k)}$ and $\beta_k^{(n-k)}$ (as well as $\gamma_k = \alpha_{k,0}/\beta_{k,0}$) for

$k = 1, \ldots, n$. By considering the doubling process of the generalized Schur algorithm, we see that, given $\alpha_0^{(n)}$ and $\beta_0^{(n)}$ where n is a power of two, the number of coefficients of α_k and β_k that are computed depends on the binary representation of the integer k. For example, the first $n/2$ coefficients of $\alpha_{n/2}$ and $\beta_{n/2}$ are calculated, while only the constant terms of α_k and β_k are calculated if k is odd. Nevertheless, *all n Schur parameters $\gamma_k = \xi_{k,1} = \alpha_k^{(1)}/\beta_k^{(1)}$ are computed in the generalized Schur algorithm.* This is important because the Schur parameters are often of significance for physical and mathematical reasons.

4 The Superfast Solution of a Positive Definite Toeplitz System

The efficient implementation of the generalized Schur algorithm is achieved by using fast Fourier transform (FFT) techniques to perform the polynomial recursions (3.3) and (3.4). A detailed description of this procedure is given in [2], where it is shown that the Schur polynomials ξ_n and η_n can be calculated using $2n \lg^2 n + O(n \lg n)$ complex multplications and $4n \lg^2 n + O(n \lg n)$ complex additions (where $\lg n \equiv \log_2 n$).

The following algorithm describes the use of the generalized Schur algorithm for the superfast solution of a positive definite Toeplitz system of equations. This algorithm is equivalent with the superfast Toeplitz solver that is presented by deHoog [10] and by Musicus [22] when applied to a positive definite matrix.

Algorithm 4.1. Let $M = [\mu_{j-k}]_{j,k=0}^{n} = M^* > 0$ where $n = 2^\nu$. The following procedure will calculate the solution of the system of equations $Mx = b$.

Set $\alpha_{0,j} := -\mu_{j+1}^*$; $\beta_{0,j} := \mu_j^*$, $(j = 0, 1, \ldots, n-1)$.

Phase 1: Use the generalized Schur algorithm to calculate ξ_n and η_n. Then obtain ψ_n from equation (3.1).

Phase 2: Solve $Mx = b$ using the Gohberg-Semencul decomposition of M^{-1} and fast Fourier transform techniques.

Phase 2 can be performed using $O(n \lg n)$ operations as described in, for example, Jain [19]. Moreover, Phase 2 can be repeated to solve $Mx = b$

for another right-hand side b. The technique of iterative improvement can therefore be efficiently implemented in this algorithm. (Of course, this is true of any algorithm that uses the Gohberg-Semencul formula in its solution phase, as in [19] and [5].)

Thus, the algorithm of deHoog and Musicus applied to a positive definite Toeplitz matrix is naturally described in terms of the generalized Schur algorithm. This algorithm therefore shares the classical roots of many of the fast and superfast algorithms.

References

[1] N. I. Akhiezer, The Classical Moment Problem, Oliver and Boyd, Edinburgh, 1965.

[2] G. S. Ammar and W. B. Gragg, *Implementation and Use of the Generalized Schur Algorithm,* in Proceedings of the 7-th International Symposium on the Mathematical Theory of Networks and Systems (MTNS-85), Stockholm, C. Byrnes and A. Lindquist, eds., North-Holland, to appear.

[3] B. D. O. Anderson and J. B. Moore, Optimal Filtering, Prentice-Hall, Englewood Cliffs, NJ, 1979.

[4] R. R. Bitmead and B. D. O. Anderson, *Asymptotically Fast Solution of Toeplitz and Related Systems of Linear Equations,* Linear Algebra Appl., 34 (1980), 103-116.

[5] R. P. Brent, F. G. Gustavson and D. Y. Y. Yun, *Fast Solution of Toeplitz Systems of Equations and Computation of Padé Approximants,* J. Algorithms, 1 (1980) 259-295.

[6] J. R. Bunch, *Stability of Methods for Solving Toeplitz Systems of Equations,* SIAM J. Sci. Statist. Comput., 6 (1985), 349-364.

[7] P. L. Chebyshev, *Sur l'Interpolation par la Méthode des Moindres Carrés,* Mém. Acad. Impér. Sci. St. Pétersbourg, 1 (1859), 1-24.

[8] G. Cybenko, Error Analysis of some Signal Processing Algorithms, Ph.D. Thesis, Princeton Univ., Princeton, NJ, 1978.

[9] G. Cybenko, *The Numerical Stability of the Levinson-Durbin Algoritm for Toeplitz Systems of Equations,* SIAM J. Sci. Statist. Comput., 1 (1980), 303-319.

[10] F. de Hoog, *On the Solution of Toeplitz Systems of Equations,* Lin. Algebra Appl., to appear.

[11] J. Durbin, *The Fitting of Time-Series Models,* Rev. Inst. Internat. Statist., 28 (1959), 229-249.

[12] W. Gautschi, *On Generating Orthogonal Polynomials,* SIAM J. Sci. Statist. Comput., 3 (1982), 289-317.

[13] I. C. Gohberg and I. A. Fel′dman, Convolution Equations and Projection Methods for their Solution, American Mathematical Society, Providence, RI, 1974.

[14] G. H. Golub and C. F. Van Loan, Matrix Computations, Johns Hopkins University Press, Baltimore, MD, 1984.

[15] W. B. Gragg, *The Padé Table and its Relation to Certain Algorithms of Numerical Analysis,* SIAM Rev., 14 (1972), 1-62.

[16] W. B. Gragg, *Positive definite Toeplitz Matrices, the Arnoldi Process for Isometric Operators, and Gaussian Quadrature on the Unit Circle (in Russian),* in Numerical Methods in Linear Algebra (E.S. Nikolaev editor), Moscow University Press, 1982, 16-32.

[17] W. B. Gragg, *The QR Algorithm for Unitary Hessenberg Matrices*, J. Comput. Appl. Math., to appear.

[18] W. B. Gragg, F. G. Gustavson, D. D. Warner and D. Y. Y. Yun, *On Fast Computation of Superdiagonal Padé Fractions*, Math. Programming Stud., 18 (1982), 39-42.

[19] J. R. Jain, *An Efficient Algorithm for a Large Toeplitz Set of Linear Equations,* IEEE Trans. Acoust. Speech Signal Process., 27 (1979), 612-615.

[20] T. Kailath, *A View of Three Decades of Linear Filtering Theory*, IEEE Trans. Inform. Theory, 20 (1974), 146-181.

[21] N. Levinson, *The Wiener RMS (Root-Mean-Square) Error Criterion in Filter Design and Prediction*, J. Math. Phys., 25 (1947), 261-278.

[22] B. R. Musicus, *Levinson and Fast Cholesky Algorithms for Toeplitz and Almost Toeplitz Matrices*, Report, Res. Lab. of Electronics, M.I.T., 1984.

[23] A. V. Oppenheim, Applications of Digital Signal Processing, Prentice-Hall, Englewood Cliffs, NJ, 1978.

[24] E. Parzen, *Autoregressive Spectral Estimation*, in Time Series in the Frequency Domain, D. Brillinger and P.R. Krishnaiah, eds., North-Holland, Amsterdam, 1983.

[25] I. Schur, *Uber Potenzreihen, die in Innern des Einheitskrises Beschränkt Sind,* J. Reine Angew. Math., 147 (1917), 205-232.

[26] G. Szegő, Orthogonal Polynomials, American Mathematical Society, Providence, RI, 1939.