CONTINUATION METHODS FOR THE COMPUTATION OF ZEROS OF SZEGŐ POLYNOMIALS*

G. S. AMMAR[†], D. CALVETTI[‡], AND L. REICHEL[§]

Abstract. Let $\{\phi_j\}_{j=0}^{\infty}$ be a family of monic polynomials that are orthogonal with respect to an inner product on the unit circle. The polynomials ϕ_j arise in time series analysis and are often referred to as Szegő polynomials or Levinson polynomials. Knowledge about the location of their zeros is important for frequency analysis of time series and for filter implementation. We present fast algorithms for computing the zeros of the polynomials ϕ_n based on the observation that the zeros are eigenvalues of a rank-one modification of a unitary upper Hessenberg matrix $H_n(0)$ of order n. The algorithms first determine the spectrum of $H_n(0)$ by one of several available schemes that require only $O(n^2)$ arithmetic operations. The eigenvalues of the rank-one perturbation are then determined from the eigenvalues of $H_n(0)$ by a continuation method. The computation of the n zeros of ϕ_n in this manner typically requires only $O(n^2)$ arithmetic operations. The algorithms have a structure that lends itself well to parallel computation. The latter is of significance in real-time applications.

AMS subject classifications. 30C15, 62M10, 65E05, 65F15, 65H17, 65H20, 65L05

Key words. Szegő polynomial, root-finder, continuation method, linear predictor, reflection coefficient, eigenvalue problem, unitary Hessenberg matrix, parallel computation.

1. Introduction.

Let $\alpha(t)$ be a distribution function with infinitely many points of increase in the interval $[-\pi, \pi]$, and assume that the moments associated with $\alpha(t)$,

$$\mu_j = \frac{1}{2\pi} \int_{-\pi}^{\pi} e^{-ijt} d\alpha(t), \qquad j = 0, \pm 1, \pm 2, \dots\ , \tag{1.1}$$

exist, where $i = \sqrt{-1}$. Then there is a family of monic polynomials $\{\phi_j\}_{j=0}^{\infty}$, with $\deg(\phi_j) = j$, that are orthogonal with respect to the inner product

$$(f, g) = \frac{1}{2\pi} \int_{-\pi}^{\pi} \overline{f(z)} g(z) d\alpha(t), \qquad z = e^{it}, \tag{1.2}$$

where the bar denotes complex conjugation. The ϕ_j are known as *Szegő polynomials.* Some of their properties are discussed in [23].

Of particular importance is the fact that the monic Szegő polynomials satisfy a recursion relation of the form

$$\begin{aligned} &\phi_0(z) = \tilde{\phi}_0(z) = 1, \\ &\phi_{j+1}(z) = z\phi_j(z) + \gamma_{j+1}\tilde{\phi}_j(z), \qquad j = 0, 1, 2 \dots\ , \\ &\tilde{\phi}_{j+1}(z) = \bar{\gamma}_{j+1} z\phi_j(z) + \tilde{\phi}_j(z). \end{aligned} \tag{1.3}$$

The recursion coefficients $\gamma_{j+1} \in \mathbb{C}$ are determined by

$$\begin{aligned} &\gamma_{j+1} = -(1, z\phi_j)/\rho_j, \qquad j = 0, 1, 2, \dots\ , \\ &\sigma_{j+1} = (1 - |\gamma_{j+1}|^2)^{1/2}, \\ &\rho_{j+1} = \sigma_{j+1}^2 \rho_j, \qquad \rho_0 = \mu_0; \end{aligned} \tag{1.4}$$

* Preprint, Feb. 1995. To appear in *Linear Algebra and Its Applications.*

[†] Department of Mathematical Sciences, Northern Illinois University, DeKalb, IL 60115. E-mail: ammar@math.niu.edu.

[‡] Department of Pure and Applied Mathematics, Stevens Institute of Technology, Hoboken, NJ 07030. E-mail: calvetti@mathrs1.stevens-tech.edu. Research supported in part by NSF grant DMS 9409422.

[§] Department of Mathematics and Computer Science, Kent State University, Kent, OH 44242. E-Mail: reichel@mcs.kent.edu. Research supported in part by NSF grant DMS 9205531.

see [23]. It can be verified by induction that the auxiliary polynomials $\tilde{\phi}_j$ satisfy

$$\tilde{\phi}_j(z) = z^j \bar{\phi}_j(1/z). \tag{1.5}$$

The recursion coefficients γ_j are in the literature referred to as *reflection coefficients*, *partial correlation coefficients* or *Schur parameters*, the choice of name depending on the application in which the ϕ_j are used. The assumption that $\alpha(t)$ has infinitely many points of increase implies that (i) $|\gamma_j| < 1$ for all j, (ii) the equations (1.3)-(1.4) hold for all j, and (iii) the zeros of each ϕ_j are of moduli less than one; see [39, Chapter 11] or [23].

Szegő polynomials arise in applications such as signal processing and time series analysis because of their connection with stationary time series. In these applications the Szegő polynomials are sometimes referred to as *Levinson polynomials.*

The present paper considers the computation of the zeros of Szegő polynomials. Knowledge of the location of the zeros is important in time series analysis and in the design and implementation of digital filters. A few applications of zeros of Szegő polynomials are described in Section 2. Some of these applications require real-time computation, and it is therefore important that an algorithm be available that has a low count of arithmetic operations and that can be implemented efficiently on a parallel computer.

The most straightforward approach to computing the zeros of the polynomials ϕ_n is to use a polynomial root-finder; see [38, Sections 5.5-5.10] for a description of a few methods. For several of these methods it is important that the zeros be determined in an appropriate order; otherwise severe loss of accuracy in the computed roots may result. Moreover, in order to obtain globally and quickly converging methods, several iterative techniques may have to be employed. These considerations can make software for reliable and quickly converging root-finders fairly complicated. Therefore the problem of finding zeros of a polynomial should be replaced by the problem of computing eigenvalues of a matrix when possible; see, e.g., the discussion in [38, p. 270].

If ϕ_n is represented in terms of the power basis, then the polynomial root-finding problem can be replaced by determining the eigenvalues of the companion matrix associated with the polynomial. The companion matrix is a Hessenberg matrix and its eigenvalues can be computed by the QR algorithm in $O(n^3)$ arithmetic operations; see [18, Chapter 7]. Experimental results reported by Goedecker [17] show that this approach performs favorably in several respects compared with polynomial root-finding algorithms in some commercially available software libraries. See also Toh and Trefethen [40] for related discussions.

We use the recursion relations (1.3) to construct an upper Hessenberg matrix, whose eigenvalues are the zeros of ϕ_n. Let

$$H_n = G_1(\gamma_1)G_2(\gamma_2)\cdots G_{n-1}(\gamma_{n-1})\hat{G}_n(\gamma_n), \tag{1.6}$$

where

$$G_j(\gamma_j) = \begin{bmatrix} I_{j-1} & & \\ & \begin{matrix} -\gamma_j & \sigma_j \\ \sigma_j & \bar{\gamma}_j \end{matrix} & \\ & & I_{n-j-1} \end{bmatrix} \in \mathbb{C}^{n\times n}, \qquad 1 \leq j < n, \tag{1.7}$$

is a unitary transformation in the $(j, j+1)$ coordinate plane defined by the recursion coefficient γ_j, and σ_j is defined by (1.4). Here I_k denotes the $k \times k$ identity matrix. Further,

$$\hat{G}_n(\gamma_n) = \begin{bmatrix} I_{n-1} & \\ & -\gamma_n \end{bmatrix}. \tag{1.8}$$

We refer to the matrices (1.7) as Givens matrices, and the matrix (1.8) as a truncated Givens matrix.

The connection between upper Hessenberg matrices of the form (1.6) and Szegő polynomials was established by Gragg [19], who showed that

$$\phi_j(z) = \det(zI_j - H_j), \qquad j \geq 1.$$

A short induction proof of this identity is presented in Section 3.

The computation of the eigenvalues of H_n by the QR algorithm for nonsymmetric matrices would require $O(n^3)$ arithmetic operations; see, e.g., Golub and Van Loan [18]. It is the purpose of this paper to present a new algorithm that requires only $O(n^2)$ arithmetic operations, and, moreover, has a structure that lends itself well to parallel computation. Our method is based on the observation that the matrix H_n is a rank-one modification of a unitary upper Hessenberg matrix. Since the Givens matrices $G_j(\gamma_j)$ are unitary, we obtain that

$$H_n^* H_n = \begin{bmatrix} I_{n-1} & \\ & |\gamma_n|^2 \end{bmatrix},$$

where * denotes transposition and complex conjugation. Thus, if γ_n were unimodular, then H_n would be unitary.

Introduce the one-parameter family of matrices

$$H_n(t) = G_1(\gamma_1)G_2(\gamma_2)\cdots G_{n-1}(\gamma_{n-1})\hat{G}_n(w(t)), \qquad 0 \leq t \leq 1, \tag{1.9}$$

where $w(t)$ is a is complex-valued differentiable function such that $w(0) = \gamma_n/|\gamma_n|$ and $w(1) = \gamma_n$. Then $H_n = H_n(1)$, and $H_n(0)$ is a closest unitary matrix to H_n; see Section 3 for details. Let $\{\lambda_j(t)\}_{j=1}^n$ denote the eigenvalues of $H_n(t)$. Since the unitary upper Hessenberg matrix $H_n(0)$ has positive subdiagonal elements, $\sigma_1, \sigma_2, \ldots, \sigma_{n-1}$, its eigenvalues $\{\lambda_j(0)\}_{j=1}^n$ are unimodular and pairwise distinct. There are several algorithms available for computing the spectrum of $H_n(0)$ in $O(n^2)$ arithmetic operations; see [5, 8, 9, 11, 20, 21, 22]. Having determined the eigenvalues $\{\lambda_j(0)\}_{j=1}^n$, we apply continuation methods, presented in Section 3, to track the eigenvalues of $H_n(t)$ as t increases from 0 to 1, in order to determine the spectrum of $H_n(1)$.

For fixed t, $0 \leq t \leq 1$, let

$$f(z,t) = \det(zI_n - H(t))$$

denote the characteristic polynomial of $H(t)$, so that $\phi_n(z) = f(z,1)$. Then (1.3) yields that, for $0 \leq t \leq 1$,

$$f(z,t) = z\phi_{n-1}(z) + w(t)\tilde{\phi}_{n-1}(z). \tag{1.10}$$

The continuation methods we consider are based on the identity

$$f(\lambda_j(t),t) = \det(\lambda_j(t)I_n - H_n(t)) = 0, \qquad 0 \le t \le 1, \qquad 1 \le j \le n. \tag{1.11}$$

We obtain from (1.11) and (1.10) that, for $1 \le j \le n$,

$$0 = \frac{d}{dt} f(\lambda_j(t),t) = f_z(\lambda_j(t),t)\frac{d\lambda_j(t)}{dt} - \tilde{\phi}_{n-1}(\lambda_j(t))w'(t). \tag{1.12}$$

If all eigenvalues $\{\lambda_j(t)\}_{j=1}^{n}$ of $H_n(t)$ are simple, then (1.12) yields the differential equations

$$\frac{d}{dt}\lambda_j(t) = w'(t)\frac{\tilde{\phi}_{n-1}(\lambda_j(t))}{f_z(\lambda_j(t),t)}, \qquad 1 \le j \le n. \tag{1.13}$$

The continuation methods described in Section 3 are prediction-correction schemes, in which the predictor is Euler's method applied to a modification of the differential equations (1.13), and the corrector is Newton's method applied to equations (1.11). The computed examples of Section 5 show that the continuation methods rarely fail to determine all zeros of a Szegő polynomials ϕ_n of moderate degree. Section 4 describes a deflation technique applicable when the continuation methods fail to determine all zeros. Deflation removes the known factors from ϕ_n and gives an upper Hessenberg matrix associated with the remainder.

Assuming that the number of arithmetic operations required to determine $\lambda_j(1)$ from $\lambda_j(0)$ for each j grows only linearly with n, the spectrum of H_n can be computed in only $O(n^2)$ arithmetic operations. Computed examples presented in Section 5 support this assumption. Note that, in general, continuation methods allow each eigenvalue $\lambda_j(1)$ to be determined from $\lambda_j(0)$ independently of the other eigenvalues. This makes continuation methods well suited for parallel computation.

Recently several continuation methods have been proposed for the computation of eigenvalues of symmetric and nonsymmetric matrices; see [15, 30, 31, 32] and references therein. These methods can be competitive when implemented on sequential as well as on parallel computers. They are usually based on a split-and-merge technique, in which the eigenvalues of certain subproblems are computed and used as starting points for a continuation method for computing the eigenvalues of the larger eigenproblem. Our approach to computing the zeros of ϕ_n can be combined with any of these continuation methods, using the eigenvalues of the unitary matrix as starting points.

Results on the distribution of zeros of Szegő polynomials are presented in [25, 26, 34], where it is shown that, under suitable conditions on the distribution function α, many of the zeros have nearly equidistant arguments. Plots of zeros of Szegő polynomials that arise from linear predictive coding of speech are shown by Brumme [10]. These plots indicate that many of the Szegő polynomials that arise in this application have their zeros close to the unit circle. They should therefore be quite easy to determine with the algorithms of the present paper. Some computational aspects of Szegő polynomials are considered in [7, 24].

2. Szegő polynomials in signal processing applications.

In this section we review some fundamental concepts of time series analysis, and describe how Szegő polynomials arise naturally in some important time series and signal processing applications. For more details and precise statements of the results mentioned, we refer to the signal processing literature, e.g., Cadzow [12], Kailath [27, 28] and Lim [33].

A real *discrete-time signal*, or *time series* is a sequence of real random variables $\{x_k\}_{k=-\infty}^{\infty}$ ordered by a discrete time variable n. The time series is often assumed to have certain statistical properties. A *signal operator* is a systematic procedure that converts an *input signal* or *excitation* $\{x_k\}_{k=-\infty}^{\infty}$ to an *output signal* or *response* $\{y_k\}_{k=-\infty}^{\infty}$. A *linear time invariant* (LTI) *operator* can be expressed as a convolution,

$$y_k = \sum_{j=-\infty}^{\infty} h_j x_{k-j}. \tag{2.1}$$

Thus, the dynamics of an LTI operator are completely determined by the sequence $\{h_j\}_{j=-\infty}^{\infty}$, which is known as the *impulse response* of the operator. An LTI operator is said to be *causal* if $h_j = 0$ for all $j < 0$. We will only consider causal LTI operators. They can be studied by mapping the signal $\{x_k\}_{k=-\infty}^{\infty}$ to a function of a complex variable by the z-transform. The z-transform of $\{x_k\}_{k=-\infty}^{\infty}$ is defined as

$$\mathcal{X}(z) = \sum_{k=-\infty}^{\infty} x_k z^{-k}, \qquad z \in \Omega_{\mathcal{X}}, \tag{2.2}$$

where $\Omega_{\mathcal{X}} \subset \mathbb{C}$ denotes the region of convergence of the right-hand side of (2.2). Let $\mathcal{X}(z)$, $\mathcal{Y}(z)$ and $\mathcal{H}(z)$ be the z-transforms of $\{x_k\}_{k=-\infty}^{\infty}$, $\{y_k\}_{k=-\infty}^{\infty}$ and $\{h_k\}_{k=0}^{\infty}$, respectively. Then (2.1) is equivalent to

$$\mathcal{Y}(z) = \mathcal{H}(z)\mathcal{X}(z), \qquad z \in \Omega_{\mathcal{H}} \cap \Omega_{\mathcal{X}}. \tag{2.3}$$

The function $\mathcal{H}(z)$ is called the *transfer function* of the operator.

A time series $\{x_k\}_{k=-\infty}^{\infty}$ is said to be *wide sense stationary* (WSS) if (i) all x_k have the same expected value $E[x_0]$, (ii) the autocorrelation of the signal is stationary,

$$\mu_{j-k} = E[x_j x_k], \tag{2.4}$$

i.e., $E[x_j x_k]$ depends only on the difference of the indices j and k, and (iii) the variance $\mu_0 - (E[x_0])^2$ of each x_k is finite. Note that

$$\mu_{-j} = \mu_j. \tag{2.5}$$

Signals that are WSS are simpler to analyze than those that are not. Therefore one often seeks to manipulate a signal so that it can be effectively treated as if it were WSS, even when the given signal is not. For instance, speech signals are not WSS. However, after a speech signal is divided into short subsequences, each one corresponding to a few milliseconds of "speech", the subsequences can be treated as if they were WSS.

A key property of WSS signals is that for some coefficients $\{\alpha_j\}_{j=1}^{m}$, we have

$$w_k = x_k + \sum_{j=1}^{m} \alpha_j x_{k-j}, \tag{2.6}$$

where $\{w_k\}_{k=-\infty}^{\infty}$ denotes a white noise signal. The number of terms in (2.6) may be infinite. The operator defined by (2.6) converts the input signal $\{x_k\}_{k=-\infty}^{\infty}$ to white noise, and can be thought of as a filter that captures all interesting statistical properties of the signal.

A linear predictor of order n predicts the next signal element by forming a linear combination of the n most recent signal elements. Assume for the moment that m in (2.6) is finite, and that all coefficients α_j are known. Define the linear predictor of order m,

$$\hat{x}_k = -\sum_{j=1}^{m} \alpha_j x_{k-j}.$$

The prediction error $\epsilon_k = x_k - \hat{x}_k$ is then the white noise signal. In general, neither the value m nor the coefficients $\{\alpha_j\}_{j=1}^m$ in (2.6) are explicitly known. When determining a linear predictor filter, one therefore typically first selects its order n, and then seeks to find coefficients $\{\alpha_j\}_{j=1}^n$ that minimize the mean square of the predictor error, given by

$$E[|\epsilon_k|^2] = E[(\sum_{j=0}^{n} \alpha_j x_{k-j})(\sum_{\ell=0}^{n} \alpha_\ell x_{k-\ell})] = \sum_{j=0}^{n}\sum_{\ell=0}^{n} \mu_{\ell-j}\alpha_j\alpha_\ell, \tag{2.7}$$

where we have defined $\alpha_0 = 1$. The last equality of (2.7) follows from the assumption that $\{x_k\}_{k=-\infty}^{\infty}$ is WSS. Requiring that the partial derivatives of the expression (2.7) with respect to the coefficients α_j vanish yields the linear system of equations

$$\sum_{j=1}^{n} \mu_{k-j}\alpha_j = -\mu_k, \qquad 1 \le k \le n, \tag{2.8}$$

for the coefficients α_j. This system is known as the *Yule-Walker equations.* Consider the Toeplitz matrix

$$T_{n+1} = \begin{bmatrix} \mu_0 & \mu_1 & & \cdots & & \mu_n \\ \mu_{-1} & \mu_0 & \mu_1 & & & \\ & \mu_{-1} & \ddots & \ddots & & \vdots \\ \vdots & & \ddots & & & \\ & & & & & \mu_1 \\ \mu_{-n} & & \cdots & & \mu_{-1} & \mu_0 \end{bmatrix} \tag{2.9}$$

defined by the autocorrelation lags μ_j in (2.8). The matrix T_{n+1} is symmetric and, in general, nonnegative definite. For ease of discussion, we will assume that T_{n+1} is positive definite. The entries $\{\mu_j\}_{j=-n}^n$ can be thought of as moments associated with a distribution function $\alpha(t)$ on the unit circle, as defined by (1.1). In fact, the moments $\{\mu_j\}_{j=-n}^n$ define a piecewise constant distribution function on the unit circle with at most n points of increase; see [23, Chapter 4]. This distribution function is unique up to an arbitrary rotation on the unit circle. It defines an inner product on the set of all polynomials of degree at most n by (1.2). We can define a family of monic orthogonal polynomials $\{\phi_j\}_{j=0}^n$ with respect to this inner product. These polynomials are uniquely determined by the moments $\{\mu_j\}_{j=-n}^n$.

Several fast algorithms are available for the solution of the Yule-Walker equations, such as Levinson's algorithm, Schur's algorithm and superfast Toeplitz solvers; see [3, 4, 12, 18, 27, 28] and references therein. These algorithms do not only provide the solution $\{\alpha_j\}_{j=1}^n$ of the Yule-Walker equations, but also the recursion coefficients

$\{\gamma_j\}_{j=1}^n$ and auxiliary coefficients $\{\sigma_j\}_{j=1}^n$ for the Szegő polynomials associated with the moments $\{\mu_j\}_{j=-n}^n$ defined by (2.4). We illustrate this with the Levinson algorithm. In view of (2.5), only moments with nonnegative index are required as input.

Algorithm 1. Levinson's algorithm for the solution of (2.8)

```
Input: $\{\mu_j\}_{j=0}^n$; Output: $\{\alpha_j\}_{j=1}^n$, $\{\gamma_j\}_{j=1}^n$, $\{\sigma_j\}_{j=1}^n$;
$\alpha_0 := 1$; $\rho_0 := \mu_0$;
for $m := 1, 2, \ldots, n$ do
    $\gamma_m := -\sum_{k=0}^{m-1} \mu_{m-k}\alpha_k/\rho_{m-1}$;
    $\sigma_m^2 := (1-\gamma_m)(1+\gamma_m)$;
    $\rho_m := \sigma_m^2\rho_{m-1}$;
    $\alpha_m := \gamma_m$;
    for $k := 1, 2, \ldots, m-1$ do
        $\zeta_k := \alpha_k + \gamma_m\alpha_{m-k}$
    end $k$;
    for $k := 1, 2, \ldots, m-1$ do $\alpha_k := \zeta_k$ end $k$
end $m$
```

□

The mean square predictor error $E[|\epsilon_k|^2]$ of the filter determined by Algorithm 1 is ρ_n. The coefficients $\{\alpha_j\}_{j=1}^n$ determined by the algorithm not only solve the Yule-Walker equations (2.8), but also are coefficients in the power basis of the auxiliary polynomials $\tilde{\phi}_n$ given by (1.5), i.e.,

$$\tilde{\phi}_n(z) = 1 + \sum_{j=1}^{n} \alpha_j z^j;$$

see, e.g., [3, 27] for details.

Assume for the moment that $m = n$ in (2.6). Then the transfer function associated with the operator (2.6) is given by

$$\mathcal{A}(z) = 1 + \sum_{j=1}^{n} \alpha_j z^{-j}.$$

Often one is interested in the transfer function associated with the inverse of the operator (2.6). It is given by

$$\mathcal{H}(z) = \frac{1}{\mathcal{A}(z)} = \frac{z^n}{\phi_n(z)}, \tag{2.10}$$

where the last equality follows from (1.5). Thus, the poles of $\mathcal{H}(z)$ are the zeros of the Szegő polynomial ϕ_n. The positive definiteness of the matrix (2.9) implies that all zeros of ϕ_n are strictly inside the unit circle. Operators associated with such transfer functions are stable, and are often used in applications where forecasting is of interest, e.g., in economics, seismology and control.

The location of the poles of the transfer function of a stable linear system determines the frequency response of the associated operator. Explicit knowledge of the poles is helpful both when manipulating $\mathcal{H}$ and when implementing filters. For instance, the implementation of the filter associated with $\mathcal{H}$ in cascade form requires the denominator of (2.10) in factored form. This implementation enables parallel processing. Moreover, in order to simplify a filter, one may wish to remove factors of the denominator of (2.10) corresponding to zeros that are not close to the unit circle.

Finally, the inverse z-transform of $\mathcal{H}(z)$ of filters can be easily computed from the partial fraction representation of $\mathcal{H}$. The computation of the partial fraction representation of $\mathcal{H}$ requires knowledge of the poles of transfer function, i.e., of the zeros of ϕ_n. The discussion of the present section extends in a straightforward manner to complex times series.

3. Continuation methods. This section presents continuation methods for the computation of zeros of Szegő polynomials ϕ_n, or equivalently, for the computation of eigenvalues of $n \times n$ matrices H_n of the form (1.6). The first step in these methods is to approximate H_n by an $n \times n$ unitary upper Hessenberg matrix $H_n(0)$ and determine its spectrum $\{\lambda_j(0)\}_{j=1}^{n}$. Any $n \times n$ unitary upper Hessenberg matrix with positive subdiagonal elements can be written in the factored form (1.9) with $t = 0$, i.e., as a product of $n-1$ Givens matrices and a truncated unitary Givens matrix. In the examples of Section 5, we compute the spectrum of $H_n(0)$ by the divide-and-conquer algorithm described in [21, 22, 8], because software is available [9]. This algorithm requires $O(n^2)$ arithmetic operations and is well suited for implementation on a parallel computer.

The matrix H_n can be approximated by a unitary upper Hessenberg matrix in many ways. Our choice of approximant $H_n(0)$ defined by (1.9) is motivated by the following result.

THEOREM 3.1. *Let H_n be an upper Hessenberg matrix of the form (1.6), and let U denote the set of all unitary upper Hessenberg matrices of order n. Then the matrix $H_n(0)$ defined by (1.9) satisfies*

$$\min_{H \in U} \|H_n - H\| = \|H_n - H_n(0)\|, \tag{3.1}$$

where $\|\cdot\|$ denotes any unitarily invariant matrix norm. The solution of the minimization problem (3.1) is unique if $\gamma_n \neq 0$. Moreover, $\|H_n - H_n(0)\|_2 = 1 - |\gamma_n|$, where $\|\cdot\|_2$ denotes the spectral norm.

Proof. The result follows from [16, Theorem 1]; see [36] for details. □

The eigenvalues $\{\lambda_j(t)\}_{j=1}^{n}$ of $H_n(t)$ are continuous functions of t. Moreover, each $\lambda_j(t)$ is a piecewise analytic function of t, for $0 \leq t \leq 1$, whose only singularities are algebraic; see Kato [29, Chapter 2]. A discussion on properties of the map $t \to \lambda_j(t)$ is also presented by Li and Zeng [31]. The purpose of the continuation methods is to track the eigenvalue paths

$$\{\lambda_j(t),\ 0 \leq t \leq 1\}, \qquad 1 \leq j \leq n,$$

in order to determine the eigenvalues $\lambda_j(1)$ of $H_n = H_n(1)$ from the eigenvalues $\lambda_j(0)$ of $H_n(0)$.

Example 1. Let $\gamma_j = 0$ for $1 \leq j \leq n$. Then $\phi_n(z) = z^n$, and the associated matrix H_n is the downshift matrix,

$$H_n = \begin{bmatrix} 0 & & & \\ 1 & \ddots & & \\ & \ddots & \ddots & \\ & & 1 & 0 \end{bmatrix}.$$

Define the family of matrices

$$H_n(t) = G_1(0)G_2(0)\cdots G_{n-1}(0)\hat{G}_n(t-1), \qquad 0 \leq t \leq 1.$$

Then $H_n(1) = H_n$ and $H_n(0)$ is unitary. The eigenvalues of $H_n(t)$ are given by

$$\lambda_j(t) = (1-t)^{1/n} e^{2\pi i(j-1)/n}, \qquad 1 \leq j \leq n, \qquad 0 \leq t \leq 1.$$

Thus, $z = 0$ is a bifurcation point of all eigenvalue paths at $t = 1$. We note that the geometric multiplicity of the eigenvalues $\lambda_j(t)$ may change with t. In the present example, $H_n(1)$ has only one linearly independent eigenvector, while the unitary matrix $H_n(0)$ has n orthogonal eigenvectors. $\square$

Tracking the eigenvalue paths is easy for many matrices $H_n(t)$, in the sense that eigenvalues $\lambda_j(t)$ only have to be determined for a few values of t in order to compute $\lambda_j(1)$. However, for some matrices $H_n(t)$ it is a numerically challenging task to follow the eigenvalue paths, and sophisticated software for the handling of bifurcation points and the determination of certain eigenvalues $\lambda_j(t)$ for increasing values of t may be required. We remark that the use of sophisticated software for path-following might not always yield satisfactory speed-ups on parallel computers due to inter-processor communication necessary close to bifurcation points. These difficulties suggest two approaches to writing reliable software for computing the spectrum of H_n based on path-following: Compute the spectrum of $H_n(0)$ by a fast algorithm, and then apply

(i) a sophisticated continuation method with step size control that can track eigenvalue paths with large and small curvature and that can handle bifurcation points.

(ii) a simple continuation method, that is easy to implement efficiently on a variety of computers and that rarely fails to determine all the eigenvalues $\{\lambda_j\}_{j=1}^n$ of $H_n = H_n(1)$ accurately. Check the accuracy of the computed eigenvalues, e.g., by evaluating ϕ_n at the computed eigenvalues. Assume that k eigenvalues are not accurate enough. Determine these k eigenvalues by applying the QR algorithm to a certain $k \times k$ nonsymmetric Hessenberg matrix.

Approach (ii) is an application of a paradigm for the development of parallel algorithms formulated by Demmel [13, p. 50]. Algorithms 2 and 3 below are based on this approach.

Surveys of continuation methods have recently been presented by Allgower and Georg [1, 2]. We apply a scheme described by Deuflhard and Hohmann [14, Section 4.4.2], which uses Euler's method as a predictor and Newton's method as a corrector.

Assume for the moment that $\lambda_j(t)$ is a simple eigenvalue of $H_n(t)$ for $0 \leq t \leq 1$, and let $\lambda_j(t_k)$ be explicitly known for the parameter value $t_k \in [0, 1)$. We would like to determine $\lambda_j(t_{k+1})$ for some t_{k+1}, $t_k < t_{k+1} \leq 1$. First we choose t_{k+1} and compute an approximation $\lambda_j^{(1)}$ of $\lambda_j(t_{k+1})$, called the *prediction*, by Euler's method in the following manner. Consider the path being followed as a curve $(\lambda_j(t), t)$. Introduce the arc length parameter s of this curve. Then

$$|\dot{\lambda}|^2 + (\dot{t})^2 = 1, \qquad \dot{t} > 0, \tag{3.2}$$

where $\dot{\lambda}$ and $\dot{t}$ denote the derivatives of λ and t with respect to the arc length s. The analogue of the differential equation (1.12) is given by

$$f_z \dot{\lambda} + f_t \dot{t} = 0. \tag{3.3}$$

Apply one step of Euler's method to the differential equation (3.3) with positive step size $h \leq 1$. The functions f_t and f_z are easily evaluated by using the Szegő recursions as described in Algorithm 4 below. In view of (3.2), the pair $(\dot{\lambda}, \dot{t})$ is then obtained by normalizing the vector $[-f_t/f_z, 1]$ to have unit length, with positive last component. We then set $h := \min\{h, (1-t_k)/\dot{t}\}$, $t_{k+1} := t_k + h\dot{t}$, and the prediction at t_{k+1} is taken to be $\lambda_j^{(1)} := \lambda_j(t_k) + h\dot{\lambda}$.

Having computed a prediction $\lambda_j^{(1)}$ for $\lambda_j(t_{k+1})$, a sequence of improved approximations $\lambda_j^{(2)}$, $\lambda_j^{(3)}, \ldots$ of $\lambda_j(t_{k+1})$ is obtained by applying Newton's method to the equation $f(z, t_{k+1}) = 0$ with starting point $z = \lambda_j^{(1)}$. This is the *correction phase* of the algorithm.

The choice of the step size h for Euler's method can dramatically affect the performance of a continuation algorithm. If h is too large, the continuation algorithm might fail to track the desired path, producing a terminal point that is actually the terminal point of another path. On the other hand, if h is too small, then the algorithm would require an unnecessarily large number of integration steps in order to reach the terminal point. Furthermore, different eigenvalue paths in the same problem may call for different step sizes. It is clear that adaptive control of the step size h is desirable.

A simple approach to step size control is to monitor the sizes of the Newton steps performed during the correction phase. The degree of stringency applied to this process is controlled by a positive parameter κ. Following Deuflhard and Hohmann [14], we consider the current step size h for Euler's method to be small enough if the Newton corrections

$$h^{(\ell)} = \frac{f(\lambda_j^{(\ell)}, t_{k+1})}{f_z(\lambda_j^{(\ell)}, t_{k+1})}$$

satisfy

$$\kappa |h^{(\ell+1)}| \le |h^{(\ell)}|, \qquad \ell \ge 1. \tag{3.4}$$

If condition (3.4) is violated for some value of $\ell \ge 1$, then h is reduced according to

$$h := 2^{-1/2} h,$$

and we perform another prediction-correction step at $\lambda(t_k)$ with the reduced step size h. Observe that if $\kappa = 0$, then the step size will never be decreased. A common choice is $\kappa = 2$. We use this value in the computed examples of Section 5.

We increase the current step size h when (3.4) is satisfied for all Newton corrections computed, and in addition,

$$\left| \frac{h^{(2)}}{h^{(1)}} \right| \le \frac{1}{8}.$$

In this case, having determined $\lambda(t_{k+1})$, we increase, if possible, the step size h by a factor $2^{1/2}$ according to

$$h := \min\{2^{1/2} h,\ 1 - t_{k+1}\}$$

and begin a new prediction-correction step at $\lambda(t_{k+1})$ with the new step size. We summarize the above discussion with the following algorithm.

TABLE 1
Input for Algorithms 2 and 3.

Variable	Description
n	degree of Szegő polynomial ϕ_n
$\{\gamma_j\}_{j=1}^n$	reflection coefficients defining ϕ_n
h_0	initial step size
$h_{\min}$	minimum step size permitted
κ	step size control parameter
maxit	maximum number of predictor-corrector steps for each eigenvalue and maximum number of Newton iterations for one eigenvalue
ϵ	tolerance
δ	tolerance for Algorithm 3 only

Algorithm 2. Simple Continuation Method

```
    Input: See Table 1;  Output: $\{\lambda_j\}_{j=1}^n$ (spectrum of $H_n$);
    Compute the spectrum $\{\lambda_j^{(0)}\}_{j=1}^n$ of $H_n(0)$ by a fast algorithm;
    for $j := 1, 2, \ldots, n$ do
        $k := 0$; $t_0 := 0$; $h := h_0$; $L := 0$; $\lambda_j(t_0) := \lambda_j^{(0)}$; $\alpha := \gamma_n/|\gamma_n|$; $w' := \gamma_n - \alpha$;
(*)     $\ell := 0$;   $\lambda_j^{(0)} := \lambda_j(t_k)$;   $L := L + 1$;
        $\tau := \dfrac{-w'\tilde{\phi}_{n-1}(\lambda_j^{(0)})}{f_z(\lambda_j^{(0)}, t_k)}$;   $\dot{t} := (|\tau|^2 + 1)^{-1/2}$;   $\dot{\lambda} := \dot{t}\tau$;
        $h := \min\{h, (1 - t_k)/\dot{t}\}$;   $\lambda_j^{(1)} := \lambda_j^{(0)} + h\dot{\lambda}$;   $t_{k+1} := t_k + h\dot{t}$;
        while $|\lambda_j^{(\ell+1)} - \lambda_j^{(\ell)}| > \epsilon$ and $\ell \le$ maxit do
            $\ell := \ell + 1$; $h^{(\ell)} := f(\lambda_j^{(\ell)}, t_{k+1})/f_z(\lambda_j^{(\ell)}, t_{k+1})$;
            if $\ell \ge 2$ and $\kappa|h^{(\ell)}| \ge |h^{(\ell-1)}|$ then
                $h := 2^{-1/2}h$;
                if $h > h_{\min}$ then goto (*) else goto no_convergence endif
            endif;
            $\lambda_j^{(\ell+1)} := \lambda_j^{(\ell)} - h^{(\ell)}$;
        endwhile;
        if $L >$ maxit then goto no_convergence endif;
        $\lambda_j(t_{k+1}) := \lambda_j^{(\ell+1)}$; $k := k + 1$;
        if $\ell \le 1$ or $8|h^{(2)}| \le |h^{(1)}|$ then $h := 2^{1/2}h$ endif;
        if $|t_k - 1| > \epsilon$ then goto (*) endif;
        $\lambda_j := \lambda_j^{(\ell+1)}$
    end $j$;
```
□

Algorithm 2 adheres to approach (ii) for designing a continuation method: the algorithm is fairly simple and is able to determine all the zeros of many Szegő polynomials. Failure to find all the zeros generally occurs because the initial step size h_0 was chosen too large, or the step size control parameter κ was chosen too small, so that computed points for two eigenvalue paths, beginning at $\lambda_j(0)$ and $\lambda_k(0)$, converge to the same terminal point; i.e., the computed $\lambda_j(1) = \lambda_k(1)$. In this case, we can perform a *retry:* paths j and k are re-followed, with $h_0 := h_0/5$ and $\kappa := \max\{2, 2\kappa\}$. Algorithm 2, together with this retry mechanism, successfully finds all the zeros of the Szegő polynomials in our experiments with randomly generated complex Schur parameters.

The algorithm may experience difficulty in computing all the zeros of the Szegő polynomial $\phi_n(z) = f(z, 1)$ if one of the matrices $H_n(t)$, $0 \leq t \leq 1$, has multiple eigenvalues. In this case, bifurcations exist among the eigenvalue paths. For generic choices of *complex* reflection coefficients $\{\gamma_j\}_{j=1}^n$, there are no bifurcations of eigenvalue paths. However, if all reflection coefficients are real, then the intermediate matrices $H_n(t)$, $0 \leq t \leq 1$, are also real, and one cannot rule out the generic possibility that complex conjugate eigenvalue paths bifurcate on the real axis. Two real eigenvalue paths may also bifurcate into a complex conjugate pair. In these situations, Algorithm 2 will generally fail to find all the eigenvalues. We must therefore modify the algorithm to accommodate bifurcations for the real problem.

In order to extend the simple continuation method of Algorithm 2 to the case that all reflection coefficients are real, we propose a modification that is capable of handling most bifurcations occurring on the real axis. First, the eigenvalues of the orthogonal Hessenberg matrix $H_n(0)$ are computed and sorted to identify the real and complex conjugate starting points for the continuation method. Those paths with initial points having nonnegative imaginary parts are then followed as in Algorithm 2 (i.e., with $w(t) \in \mathbb{R}$) until a possible bifurcation is detected. For a path beginning at $\lambda_j(0) \notin \mathbb{R}$, a bifurcation is detected if a computed $\lambda_j(t_k)$ is too close to the real axis. A bifurcation on a path beginning at ± 1 is detected if it cannot be followed to termination.

If the path beginning at $\lambda_j(0)$ is suspected of bifurcating, we attempt to follow a new path beginning at $\lambda_j(0)$, where the last Schur parameter $w(t)$ varies along a parabolic arc in the complex plane. If $\lambda_j(0) \notin \mathbb{R}$, then we also follow the path beginning at $\bar{\lambda}_j(0)$ with complex perturbations of the last Schur parameter as well. By introducing complex perturbations in this manner, we can usually avoid the bifurcation present in the real problem. In our experiments, the last reflection coefficient follows the arc belonging to the parabola intersecting the real axis at $\gamma_n(0) = \alpha$ and $\gamma_n(1) = \gamma_n$ with vertex at $\gamma_n + \beta/2 + i|\beta|/2$, where $\alpha = \gamma_n/|\gamma_n|$, $\beta = \alpha - \gamma_n$ and $i = \sqrt{-1}$.

We also introduce complex perturbations in the retry mechanism. If two paths have a common termination point, and this point is not a multiple eigenvalue of $H(1)$, we then re-follow the two paths in question using complex perturbations in the same manner.

Computed examples indicate that this scheme is almost always successful in computing all eigenvalues of H_n. In a few cases, however, bifurcation occurs at such an early stage that the described complex perturbation of the last reflection coefficient does not successfully separate the eigenvalue paths. When this happens, we apply the deflation scheme described in Section 4, in order to produce a smaller upper Hessenberg matrix H_k whose spectrum are the eigenvalues of H_n that remain to be determined. The eigenvalues of H_k are then computed using the QR algorithm for nonsymmetric Hessenberg matrices described in [18]. We summarize the discussion above with the following algorithm.

Algorithm 3. A Continuation Method for the Real Problem

```
Input: See Table 1;  Output: $\{\lambda_j\}_{j=1}^n$ (spectrum of $H_n$);
Compute the spectrum $\{\lambda_j^{(0)}\}_{j=1}^n$ of $H_n(0)$ by a fast algorithm;
Sort $\{\lambda_k^{(0)}\}_{k=1}^n$ so that if $\mathrm{Im}(\lambda_k^{(0)}) > 0$, then $\lambda_{k+1}^{(0)} := \bar{\lambda}_k^{(0)}$ ;
for $j := 1, 2, \ldots, n$ do $\nu_j := 1 + \mathrm{sign}\big(\mathrm{Im}(\lambda_j^{(0)})\big)$; endif;
for $j := 1, 2, \ldots, n$ do
        if $\nu_j = 0$ then
              if $\nu_{j-1} = 2$ then
                    $\lambda_j := \bar{\lambda}_{j-1}$; goto $(***)$
              else
                    $\nu_j := -1$;
              endif;
        endif;
(**)    $k := 0$; $t_0 := 0$; $h := h_0$; $L := 0$; $\lambda_j(t_0) := \lambda_j^{(0)}$;
        if $\nu_j = -1$ then $q_{\mathrm{bif}} := 1$; else $q_{\mathrm{bif}} := 0$; endif;
        $\alpha := \gamma_n/|\gamma_n|$; $\beta = \gamma_n - \alpha$; $w = \alpha$; $w' := \beta + q_{\mathrm{bif}}|\beta|i$;
(*)     $\ell := 0$;   $L := L + 1$;   $\lambda_j^{(0)} := \lambda_j(t_k)$;
        $\tau := \dfrac{-w'\tilde{\phi}_{n-1}(\lambda_j^{(0)})}{f_z(\lambda_j^{(0)}, t_k)}$;   $\dot{t} := (|\tau|^2 + 1)^{-1/2}$;   $\dot{\lambda} := \dot{t}\tau$;
        $h := \min\{h, (1 - t_k)/\dot{t}\}$;   $\lambda_j^{(1)} := \lambda_j^{(0)} + h\dot{\lambda}$;   $t_{k+1} := t_k + h\dot{t}$;
        $w' := \beta + q_{\mathrm{bif}}(1 - 2t_{k+1})|\beta|i$;   $w := \alpha + \beta t_{k+1} + q_{\mathrm{bif}} t_{k+1}(1 - t_{k+1})|\beta|i$;
        while $|\lambda_j^{(\ell+1)} - \lambda_j^{(\ell)}| > \epsilon$ and $\ell \le$ maxit do
              $\ell := \ell + 1$; $h^{(\ell)} := f(\lambda_j^{(\ell)}, t_{k+1})/f_z(\lambda_j^{(\ell)}, t_{k+1})$;
              if $\ell \ge 2$ and $\kappa|h^{(\ell)}| > |h^{(\ell-1)}|$ then
                    $h := 2^{-1/2}h$;
                    if $h > h_{min}$ then goto $(*)$ endif;
                    if $\nu_j = 1$ then $\nu_j := -1$; goto $(**)$ endif;
                    goto no_convergence;
              endif;
              $\lambda_j^{(\ell+1)} := \lambda_j^{(\ell)} - h^{(\ell)}$
        endwhile;
        if $\nu_j = 2$ and $\mathrm{Im}(\lambda_j^{(\ell+1)}) < \delta$ then
              Bifurcation likely. Restart with complex perturbation of $\gamma_n$;
              $\nu_j := -1$; go to $(**)$
        endif;
        if $L >$ maxit then goto no_convergence endif;
        if $\ell \le 1$ or $8|h^{(2)}| \le |h^{(1)}|$ then $h := 2^{1/2}h$ endif;
        $\lambda_j(t_{k+1}) := \lambda_j^{(\ell+1)}$;   $k := k + 1$;
        if $|t_k - 1| > \epsilon$ then goto $(*)$ endif;
        $\lambda_j := \lambda_j^{(\ell+1)}$
(***)   continue;
end $j$;
```

□

The variables ν_j indicate the type of path being followed. If $\nu_j > 0$, then $w(t)$ is varied radially, as in Algorithm 2. If $\nu_j = 2$, then $\nu_{j+1} = 0$, and we take $\lambda_{j+1}(t) = \bar{\lambda}_j(t)$. When a possible bifurcation is detected, then ν_j is set to -1, and the last Schur parameter $w(t)$ is varied along a parabolic arc.

We remark that the computations inside the j loops of Algorithms 2 and 3 can be carried out in parallel for different values of j. Since we are not interested in the eigenvalue paths themselves, but only in their endpoints, it is natural to choose the initial step size $h_0 = 1$ for both algorithms.

Actions taken when exit no_convergence is used in Algorithms 2 and 3 have not been specified in order to keep the description of the algorithms simple. We remark that in the computed examples of Section 5 when Algorithm 2 was applied to complex matrices H_n, the exit no_convergence was never used.

After each path has been followed, we check for common terminal points. We then perform a retry, with decreased initial step size and increased step size control parameter, for all paths that have common termination points, as well as for the conjugates of these paths, and also for any paths that did not successfully terminate in the first try. Complex perturbations are used in all retries for Algorithm 3. In the experiments reported in Section 5, we perform a maximum of four retries. Let q_{found} denote the number of eigenvalues determined by Algorithm 3 together with the retry mechanism. If $q_{\text{found}} < n$, then we employ a deflation procedure, described in Section 4, to H_n in order to obtain a (small) upper Hessenberg matrix H_k of order $k = n - q_{\text{found}}$. The eigenvalues of H_k are the k eigenvalues of H that have not been determined sufficiently accurately.

If Algorithm 2 or Algorithm 3 yields a numerically multiple eigenvalue of H_n, say λ_j, of multiplicity m_j, then either H_n has m_j eigenvalues very close to λ_j, or the algorithm has failed to track the n eigenvalue paths. These cases can be distinguished by evaluating

$$\frac{d^k}{dz^k}\phi_n(\lambda_j), \qquad k = 0, 1, \ldots, m_j - 1. \tag{3.5}$$

If the polynomial and its derivatives at λ_j in (3.5) vanish, then ϕ_n has a zero at λ_j of multiplicity m_j.

Both Algorithm 2 and Algorithm 3 require the evaluation of ϕ_n, ϕ_n' and $\tilde{\phi}_{n-1}$ at various points in the closed unit disk. The following scheme for this purpose can be obtained by differentiating the recursion formulas (1.3).

Algorithm 4. Evaluation of polynomials

```
**Input**: $\{\gamma_j\}_{j=1}^{n-1}$, $z$, $w(t)$; **Output**: $f(z,t)$, $f_z(z,t)$, $\tilde{\phi}_{n-1}(z)$;
    $\phi_0 := \tilde{\phi}_0 := 1$; $\phi_0' := \tilde{\phi}_0' := 0$;
    **for** $j = 1, 2, \ldots, n-1$ **do**
        $\tau := \phi_{j-1} + z\phi_{j-1}'$; $\phi_j' := \tau + \gamma_j\tilde{\phi}_{j-1}'$; $\tilde{\phi}_j' := \bar{\gamma}_j\tau + \tilde{\phi}_{j-1}'$;
        $\phi_j := z\phi_{j-1} + \gamma_j\tilde{\phi}_{j-1}$; $\tilde{\phi}_j := \bar{\gamma}_j z\phi_{j-1} + \tilde{\phi}_{j-1}$;
    **end** $j$;
    $\tilde{\phi}_{n-1}(z) := \tilde{\phi}_{n-1}$; $f_z(z,t) := \phi_{n-1} + z\phi_{n-1}' + w(t)\tilde{\phi}_{n-1}'$;
    $f(z,t) := z\phi_{n-1} + w(t)\tilde{\phi}_{n-1}$;
```
□

Algorithm 4 can easily be extended to the evaluation of higher order derivatives which may be required in (3.5). We conclude this section with a simple proof of the representation (1.6) of the matrix H_n defining the Szegő polynomial ϕ_n.

THEOREM 3.2. *Let $\{\gamma_j\}_{j=1}^n$ be a sequence of complex numbers of modulus less than unity, and let ϕ_n be the monic Szegő polynomial of degree n defined by the recursion formula (1.3). Then ϕ_n is the characteristic polynomial of the matrix H_n defined by (1.6).*

Proof. This result has been shown by Gragg [19]. We present a simple induction proof. Observe that the matrix H_n defined by (1.6) has the form

$$H_n = \begin{bmatrix} -\gamma_1 & -\sigma_1\gamma_2 & -\sigma_1\sigma_2\gamma_3 & \cdots & -\sigma_1\sigma_2\cdots\sigma_{n-1}\gamma_n \\ \sigma_1 & -\bar{\gamma}_1\gamma_2 & -\bar{\gamma}_1\sigma_2\gamma_3 & \cdots & -\bar{\gamma}_1\sigma_2\cdots\sigma_{n-1}\gamma_n \\ 0 & \sigma_2 & -\bar{\gamma}_2\gamma_3 & \cdots & -\bar{\gamma}_2\sigma_3\cdots\sigma_{n-1}\gamma_n \\ \vdots & & \ddots & \ddots & \vdots \\ 0 & \cdots & 0 & \sigma_{n-1} & -\bar{\gamma}_{n-1}\gamma_n \end{bmatrix}.$$

Consequently, the characteristic polynomial $\phi_n(z)$ of H_n satisfies

$$\phi_n(z) = \det(zI - H_n) = \begin{vmatrix} & & & & \sigma_1\sigma_2\cdots\sigma_{n-1}\gamma_n \\ & & & & \bar{\gamma}_1\sigma_2\cdots\sigma_{n-1}\gamma_n \\ & & zI - H_{n-1} & & \bar{\gamma}_2\sigma_3\cdots\sigma_{n-1}\gamma_n \\ & & & & \vdots \\ & & & & \bar{\gamma}_{n-2}\sigma_{n-1}\gamma_n \\ 0 \;\; 0 & \cdots & 0 & -\sigma_{n-1} & z + \bar{\gamma}_{n-1}\gamma_n \end{vmatrix}$$

$$= z\phi_{n-1}(z) + \gamma_n\chi_{n-1}(z), \tag{3.6}$$

where

$$\chi_{n-1}(z) = \begin{vmatrix} & & & & \sigma_1\sigma_2\cdots\sigma_{n-1} \\ & & & & \bar{\gamma}_1\sigma_2\cdots\sigma_{n-1} \\ & & zI - H_{n-1} & & \bar{\gamma}_2\sigma_3\cdots\sigma_{n-1} \\ & & & & \vdots \\ & & & & \bar{\gamma}_{n-2}\sigma_{n-1} \\ 0 \;\; 0 & \cdots & 0 & -\sigma_{n-1} & \bar{\gamma}_{n-1} \end{vmatrix}. \tag{3.7}$$

By expanding the determinant (3.7) along the bottom row, and using the fact that $|\gamma_j|^2 + \sigma_j^2 = 1$ for $1 \le j < n$, we obtain

$$\begin{aligned} \chi_{n-1} &= \bar{\gamma}_{n-1}\phi_{n-1} + \sigma_{n-1}^2\chi_{n-2} \\ &= \chi_{n-2} + \bar{\gamma}_{n-1}(\phi_{n-1} - \gamma_{n-1}\chi_{n-2}), \end{aligned}$$

which, together with (3.6), with n replaced by $n-1$, yields

$$\chi_{n-1}(z) = z\bar{\gamma}_{n-1}\phi_{n-2}(z) + \chi_{n-2}(z). \tag{3.8}$$

Define $\phi_0(z) = \chi_0(z) = 1$. Then (3.6) holds for $n = 1$, and therefore for all $n \ge 1$, with χ_{n-1} given recursively by (3.8) for $n > 1$.

Let $\tilde{\phi}_n(z)$ denote the auxiliary polynomials defined by (1.5). Then (3.8) shows that the $\tilde{\phi}_n$ satisfy the same recurrence relation as the χ_n. In view of that $\tilde{\phi}_0(z) = \chi_0(z) = 1$, we obtain that $\chi_n = \tilde{\phi}_n$ for $n \ge 0$. Thus, (3.6) shows that the characteristic polynomial of H_n is the Szegő polynomial ϕ_n given by (1.3). □

4. Deflation.

The simple procedure for handling bifurcations for real problems implemented by Algorithm 3 works well on a large majority of the examples we considered. However, for some matrices H_n, the algorithm fails to find one or two eigenvalues. It may be possible to determine all eigenvalues by using more sophisticated, and more complicated, strategies for handling bifurcations on the real axis; see, e.g., Li and Zeng [31] for a discussion of such methods. Another possibility for finding any remaining eigenvalues is to develop an efficient deflation procedure. In this section we show that the unitary Hessenberg QR algorithm of Gragg [20] can be used to efficiently perform deflation on H_n.

Assume the path-following procedure outlined in Algorithm 3 produces only $n-k$ accurate eigenvalues of H_n, where $k \ll n$. In the computed examples of Section 5, where the order n ranges from 4 to 18, k is never larger than 2. We can then perform *deflation* to remove the $n-k$ accurate eigenvalues of the matrix H_n. This yields a Hessenberg matrix H_k of order k, whose spectrum consists of the eigenvalues of H_n that have not been determined sufficiently accurately yet.

Deflation can be performed by applying the QR algorithm to H_n using the accurately computed eigenvalues as shifts. These will be referred to as *ultimate shifts*. In exact arithmetic, one step of the QR algorithm with an ultimate shift produces a new Hessenberg matrix whose $(n, n-1)$ entry is zero. However, in finite precision arithmetic, the $(n, n-1)$ entry might be of magnitude considerably larger than zero. One can repeat deflation, i.e., application of the ultimate shift by the QR algorithm, until the $(n, n-1)$ element of H_n is of "tiny" magnitude. In this manner we obtain a matrix H_{n-1} of order $n-1$, whose spectrum does not contain the deflated eigenvalue. This process is repeated for all accurately computed eigenvalues of H_n to yield a $k \times k$ Hessenberg matrix H_k. If $k \leq 2$, then the spectrum of H_k can be computed analytically. When $k > 2$, the eigenvalues of H_k can be determined by any standard algorithm for the computation of eigenvalues of a nonsymmetric matrix, such as the the QR algorithm.

One step of the QR algorithm with ultimate shift requires $O(n^2)$ arithmetic operations when implemented in terms of the matrix elements, so the direct application would be too slow for our purposes: its use would defeat the efficiency of the continuation method. However, one can attempt to use the representation of H_n in product form (1.6) to reduce the number of arithmetic operations necessary to $O(n)$. Such a procedure is known for the case that H_n is a unitary Hessenberg matrix. One QR step on the unitary matrix H_n produces another unitary Hessenberg matrix $\hat{H}_n$, and the transformation can be made in such a way that the intermediate Hessenberg matrices generated are also unitary. The transformation then can be implemented in terms of Schur parameters using only $O(n)$ arithmetic operations per QR step. This idea is fundamental to Gragg's unitary Hessenberg QR (UHQR) algorithm [20], which is applied with ultimate shifts in [6] to deflate unitary Hessenberg matrices, and to downdate discrete least squares trigonometric approximants and discrete Fourier transforms.

Our matrix H_n, in product form (1.6), is not unitary. Since $|\gamma_n| < 1$, the matrix obtained by applying one step of the QR algorithm to H_n is an upper Hessenberg matrix $\hat{H}_n$ that is not representable in product form (1.6). Consequently, the UHQR algorithm cannot be applied to our matrix H_n. We elaborate on this below, and show that in the special case that the $(n-1, n)$ entry of $\hat{H}_n$ vanishes, the leading principal submatrix of order $n-1$ of $\hat{H}_n$ is representable in product form.

PROPOSITION 4.1. *Let H_n be an upper Hessenberg matrix with nonnegative sub-*

diagonal elements. Then H_n is representable in the form (1.6) with each $|\gamma_j| \le 1$ if and only if

$$H_n^* H_n = \begin{bmatrix} I_{n-1} & 0 \\ 0^* & \kappa^2 \end{bmatrix}, \tag{4.1}$$

where $0 \le \kappa \le 1$. If H_n satisfies (4.1), then the representation (1.6) is unique, and $\kappa = |\gamma_n|$.

Proof. Clearly, any matrix of the form (1.6) satisfies (4.1). Conversely, assume (4.1) is satisfied, and let $H_n = QR$ be the unique QR factorization of H_n, where the diagonal elements of R are nonnegative. Then $R = \mathrm{diag}[1, 1, \ldots, 1, \kappa]$. Since Q is a unitary upper Hessenberg matrix with nonnegative subdiagonal elements, it has the unique representation $Q = G_1(\gamma_1) \cdots G_{n-1}(\gamma_{n-1})\hat{G}_n(\tilde{\gamma}_n)$, where $|\tilde{\gamma}_n| = 1$. Thus, H_n is in the form (1.6) with $\gamma_n = \kappa\tilde{\gamma}_n$. □

From this characterization it is easy to see why the QR algorithm does not preserve the form (1.6) when $|\gamma_n| < 1$. One QR step applied to H_n produces a matrix

$$\hat{H}_n = Q^* H_n Q = \begin{bmatrix} \hat{H}_{n-1} & y_{n-1} \\ \hat{\sigma}_{n-1} e_{n-1}^T & \eta_{nn} \end{bmatrix}, \tag{4.2}$$

where $Q = G_1(\alpha_1) \cdots G_{n-1}(\alpha_{n-1})\hat{G}_n(\alpha_n)$ is unitary. From this representation of Q, we have

$$\begin{aligned} \hat{H}_n^* \hat{H}_n &= Q^* \begin{bmatrix} I_{n-1} & 0 \\ 0^* & |\gamma_n|^2 \end{bmatrix} Q \\ &= \hat{G}_n^*(\alpha_n) G_{n-1}^*(\alpha_{n-1}) \begin{bmatrix} I_{n-1} & 0 \\ 0^* & |\gamma_n|^2 \end{bmatrix} G_{n-1}(\alpha_{n-1}) \hat{G}_n(\alpha_n) \\ &= \begin{bmatrix} I_{n-2} & 0 \\ 0^* & X \end{bmatrix}, \end{aligned}$$

where

$$X = \begin{bmatrix} |\alpha_{n-1}|^2 + \beta_{n-1}^2 |\gamma_n|^2 & \bar{\alpha}_{n-1}\beta_{n-1}\alpha_n(1 - |\gamma_n|^2) \\ \alpha_{n-1}\beta_{n-1}\bar{\alpha}_n(1 - |\gamma_n|^2) & \beta_{n-1}^2 + |\alpha_{n-1}|^2 |\gamma_n|^2 \end{bmatrix},$$

with $\beta_{n-1} = (1 - |\alpha_{n-1}|^2)^{1/2}$ and $|\alpha_n| = 1$. Clearly, if $\kappa = |\gamma_n| < 1$, then $\hat{H}_n$ will not be representable in the form (1.6) unless $|\alpha_{n-1}| = 1$ (and $\beta_{n-1} = 0$). But this cannot happen in exact arithmetic when the original matrix H_n is an irreducible Hessenberg matrix, since in this case Q is also irreducible, and hence $\beta_{n-1} \ne 0$.

On the other hand, let us assume that, after applying one QR step to H_n, we obtain the matrix $\hat{H}_n$ with $\hat{\sigma}_{n-1} = 0$ (i.e., an exact deflation). Then

$$\hat{H}_n^* \hat{H}_n = \begin{bmatrix} \hat{H}_{n-1}^* \hat{H}_{n-1} & \hat{H}_{n-1}^* y_{n-1} \\ y_{n-1}^* \hat{H}_{n-1} & y_{n-1}^* y_{n-1} + \eta_{nn}^2 \end{bmatrix},$$

and moreover,

$$\hat{H}_{n-1}^* \hat{H}_{n-1} = \begin{bmatrix} I_{n-2} & 0 \\ 0^* & \hat{\kappa}^2 \end{bmatrix},$$

where $\hat{\kappa}^2 = |\alpha_{n-1}|^2 + \beta_{n-1}^2 \kappa^2$. We therefore have the following result.

THEOREM 4.2. *Let H_n be given in product form (1.6), where $|\gamma_j| < 1$ for $1 \leq j \leq n$, and assume that one step of the QR algorithm with shift applied to H_n results in the upper Hessenberg matrix $\hat{H}_n$ given in (4.2). If the $(n, n-1)$ entry of $\hat{H}_n$ is zero, then the deflated submatrix $\hat{H}_{n-1}$ can be represented in product form. Moreover, this representation can be obtained by applying the UHQR algorithm to H_n.*

Thus, if a QR step with ultimate shift provides deflation to desired accuracy, we can apply the recursions among the reflection coefficients from the UHQR algorithm to remove a known eigenvalue from H_n using $O(n)$ operations. If we successfully deflate all $n-k$ known eigenvalues of H_n in this manner, we obtain the desired matrix H_k in the form the form (1.6) in $O(n^2)$ arithmetic operations. Our experiments indicate that the application of ultimate shifts works well for small to moderately sized problems. For large problems, such as finding the zeros of a Szegő polynomial with real reflection coefficients of degree 100, deflation can result in loss of accuracy. Further investigation of the deflation process is necessary before it can safely be applied to large problems. The related problem of using ultimate shifts on symmetric tridiagonal matrices is discussed by Parlett and Le [35].

5. Computed examples.

In this section we present the results of several computed examples which illustrate the performance of our continuation methods. The computer programs used were all written in FORTRAN 77, and the numerical experiments were carried out on an IBM RISC 6000/550 workstation and a SUN SparcStation 5 in single precision arithmetic, i.e., with approximately 7 significant decimal digits.

We obtain our test problems by randomly generating the reflection coefficients that define the Hessenberg matrix H_n. We compute the eigenvalues of the closest unitary Hessenberg matrix $H_n(0)$ with the unitary divide-and-conquer code described in [8, 9]. The eigenvalues of the unitary Hessenberg matrix $H_n(0)$ are the starting points of the eigenvalue paths. As the last reflection coefficient is moved away from the unit circle, i.e., as the parameter t in (1.9) is increased from 0 to 1, the eigenvalues of the intermediate matrices $H_n(t)$ move from the unit circle to the eigenvalues of the Hessenberg matrix $H_n = H_n(1)$. We use the continuation methods implemented by Algorithms 2 and 3 to track the eigenvalue paths as the last reflection coefficient moves from $\gamma_n/|\gamma_n|$ to γ_n. To check that the continuation methods find the correct eigenvalues, we compare the endpoints of the eigenvalue paths with the eigenvalues of the original Hessenberg matrix obtained from the EISPACK subroutines COMQR or HQR.

In general, when the reflection coefficients are complex numbers, the eigenvalues of the associated matrices or, equivalently, the zeros of the corresponding Szegő polynomials, are complex numbers inside the unit circle in generic positions. In this case we expect the simple continuation scheme based on a radial perturbation of the last reflection coefficient to be adequate for determining the eigenvalues of the Hessenberg matrix H_n. Indeed, Algorithm 2 never failed to yield the correct eigenvalues in our experiments when the reflection coefficients were generic complex numbers. Table 2 displays the results of 1000 runs on problems of order $n = 10, 20, \ldots, 100$, with initial step size $h_0 = 1$ and step size control parameter $\kappa = 1$. The table displays the total number of retries performed during the 1000 runs, the average number of iterations, and the average CPU time of Algorithm 2 and of the EISPACK subroutine COMQR on a SparcStation 5. Note that the number of retries increases linearly with n in these

TABLE 2
Results for Algorithm 2: $h_0 := 1$, $\kappa := 1$.

n	retries	average iterations per eigenvalue	cpu seconds	
			Algorithm 2	COMQR
10	11	3.67	0.25E–01	0.23E–01
20	27	2.91	0.83E–01	0.15E+00
30	23	2.62	0.17E+00	0.47E+00
40	30	2.49	0.26E+00	0.11E+01
50	33	2.40	0.37E+00	0.20E+01
60	42	2.34	0.49E+00	0.34E+01
70	43	2.29	0.63E+00	0.54E+01
80	59	2.29	0.75E+00	0.79E+01
90	81	2.25	0.91E+00	0.11E+02
100	95	2.24	0.11E+01	0.15E+02

experiments.

Figures 1 and 2 show the paths followed by the eigenvalues of two families of matrices $H_8(t)$, defined by complex reflection coefficients, when increasing t from 0 to 1. It is interesting to see that, while in Figure 1 no eigenvalue moves very far from the unit circle, this is no longer the case for the eigenvalues in Figure 2. This is because the modulus of the last reflection coefficient used to generate the matrix H_8 for Figure 2 is very small.

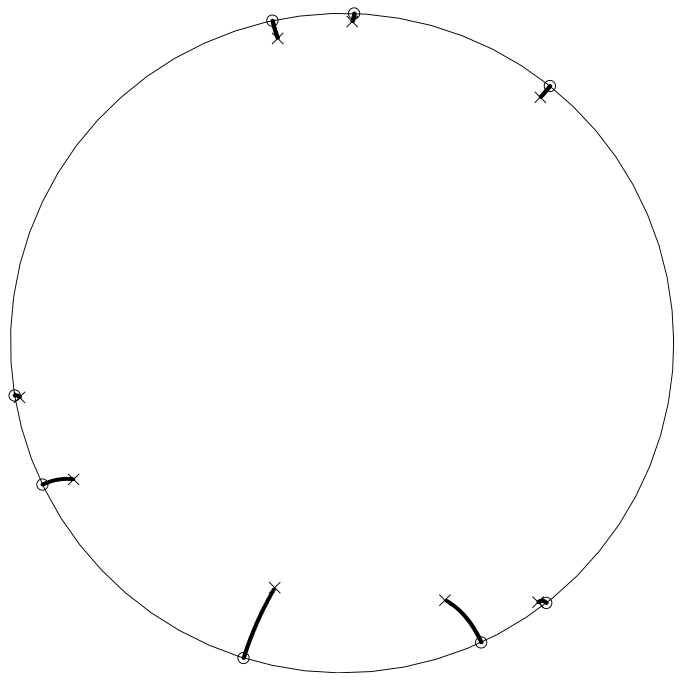

FIG. 1. *Complex reflection coefficients,* $n = 8$, $|\gamma_n| = 0.50$.

When the reflection coefficients are all real, the zeros of the corresponding Szegő polynomials are either real or occur in complex conjugate pairs and bifurcations occur when complex eigenvalue paths touch the real axis, or when two real eigenvalue paths

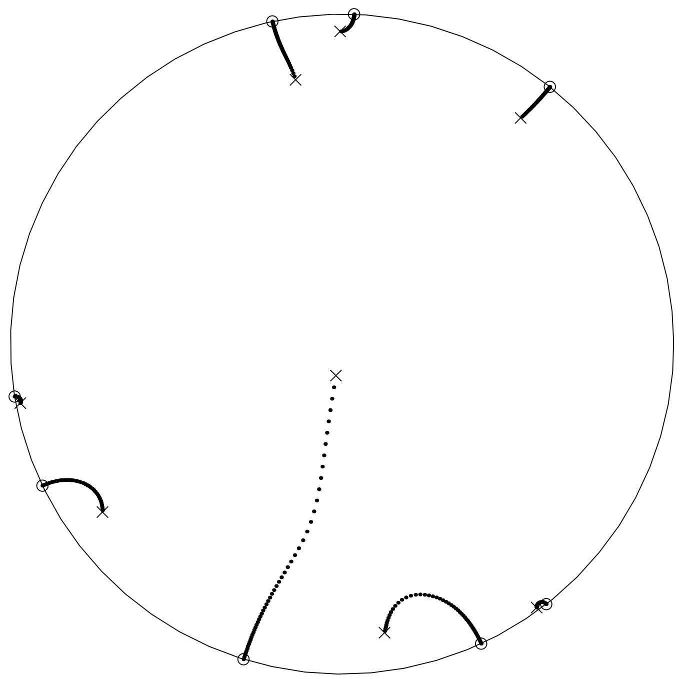

FIG. 2. *Complex reflection coefficients,* $n = 8$, $|\gamma_n| = 0.050$.

meet and bifurcate into a complex conjugate pair. The second column of Table 3 indicates that the likelihood of having bifurcations increases with the size of the problem. However, even for problems of small size, e.g., 4 or 6, we observed that a bifurcation of a pair of eigenvalue paths occurs, on the average, in approximately 20 percent of the problems. When a bifurcation on the real axis occurs, we track again the paths of the eigenvalues which bifurcated from their initial points on the unit circle, and we make the last reflection coefficient move along a parabolic arc in the complex plane. Figure 3 shows the paths traveled by the eigenvalues in the case of real reflection coefficients when the last reflection coefficient is moved along a real line segment. Notice that two of the complex conjugate eigenvalue paths meet on the real line and bifurcate. Figure 4 shows how bifurcation of the eigenvalue paths can be avoided by moving the last reflection coefficient along a parabolic arc in the complex plane.

When, in spite of the complex perturbation of the last reflection coefficient, we were unable to compute all eigenvalues accurately by the continuation method, the deflation scheme of Section 4 delivered the remaining eigenvalues.

Table 3 summarizes our experience using the continuation scheme for real reflection coefficients for computing the zeros of 1000 Szegő polynomials of even degree between 4 and 18. The degree of the Szegő polynomials, n, is listed in the first column. For each n we consider the upper Hessenberg matrix H_n defined by n reflection coefficients randomly generated from the uniform $[-1, 1]$ distribution. We compute the eigenvalues of the unitary matrix $H_n(0)$, obtained by changing the last reflection coefficients to -1 or 1, by the unitary divide-and-conquer code described in [8, 9]. Starting from each eigenvalue of $H_n(0)$, we track the eigenvalue paths using Algorithm 3 until either the endpoints of the paths are reached, or the algorithm stops after reaching the maximum number of iterations. Column 2 of Table 3 shows how many times out of 1000 runs the algorithm reached the endpoint of all continuation paths and determined

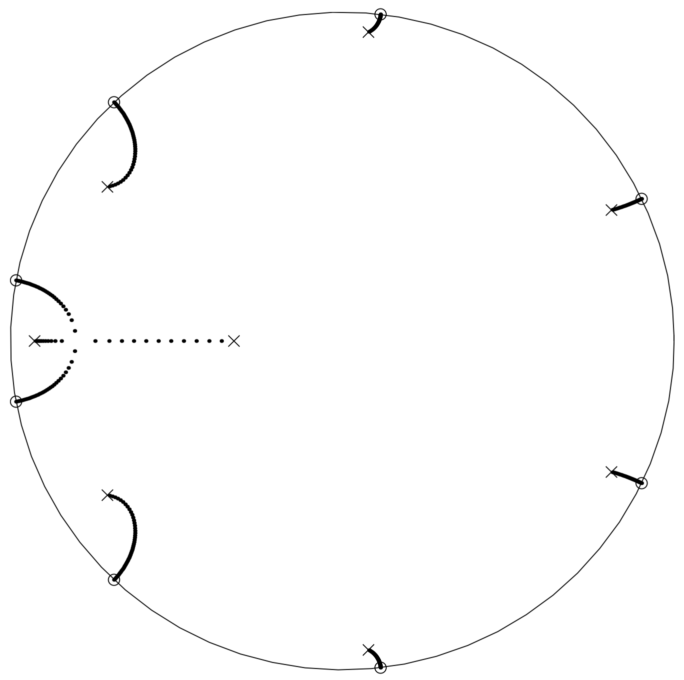

FIG. 3. *Real reflection coefficients, $n = 8$, γ_n associated with H_n of magnitude* 0.16*, γ_n is moved along real line segment.*

n distinct eigenvalues for the matrix H_n. The numbers listed in Column 2 shows that the continuation algorithm for real reflection coefficients almost always computes all eigenvalues correctly. The third column of Table 3, labeled "bifurcations", counts the total number of bifurcations which occurred in the course of the 1000 experiments, i.e., in the computation of $1000n$ eigenvalues. We remark that, while often only one bifurcation occurred, several problems display two or even three bifurcations. As we would expect, the number of bifurcations increases with the degree of the Szegő polynomial. The last column of Table 3, labeled "iterations/eigenvalue" reports the average number of Newton steps needed for the computation of each eigenvalue. We remark that the value reported is the average over the computation of $1000n$ eigenvalues. We observed that for some problems the number of iterations per eigenvalue can be as low as 2 or higher than 20. The table suggests that the average number of iterations per eigenvalue does not increase with the size of the problem.

6. Conclusion.

Two algorithms are described for the computation of zeros of Szegő polynomials. They require generally only $O(n^2)$ arithmetic operations to determine the zeros of a Szegő polynomial of degree n. The structure of the algorithms makes them easy to implement on a parallel computer. This may be important for real-time signal processing application.

Acknowledgements. We would like to thank Carl Jagels for a careful reading of the paper. L.R. would like to thank Wolf Hofmann for reference [10]. G.A. would like to thank Layne Watson for helpful discussions.

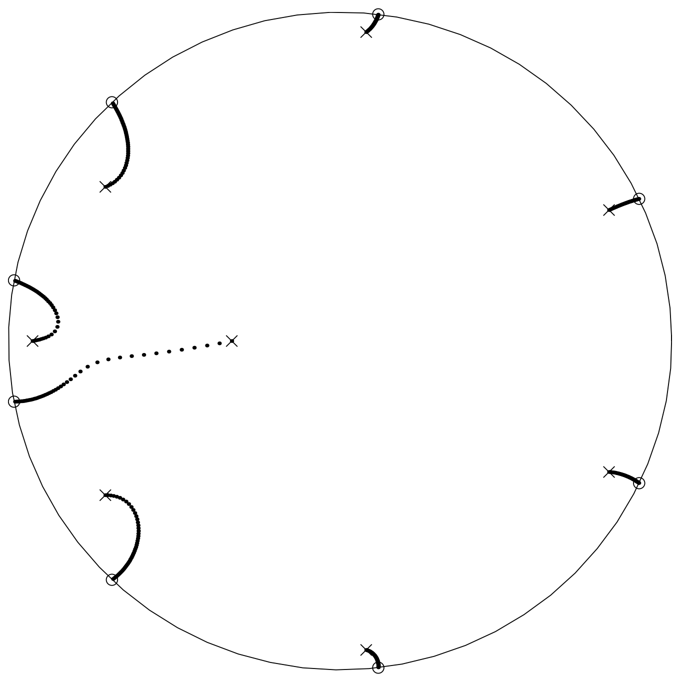

FIG. 4. *Real reflection coefficients, $n = 8$, γ_n associated with H_n of magnitude* 0.16*, γ_n is moved along parabolic arc in the complex plane.*

TABLE 3

n	distinct eigenvalues	bifurcations	iterations/eigenvalue
4	1000	191	6.69
6	1000	266	6.28
8	1000	322	6.07
10	998	398	6.53
12	1000	452	6.02
14	995	533	6.38
16	997	547	6.07
18	997	571	5.87

REFERENCES

[1] E. L. Allgower and K. Georg, Numerical Continuation Methods, Springer, Berlin, 1990.

[2] E. L. Allgower and K. Georg, Continuation and path following, Acta Numer. 1993, Cambridge U. P., Cambridge, 1993, pp. 1–64.

[3] G. S. Ammar and W. B. Gragg, The generalized Schur algorithm for the superfast solution of Toeplitz systems, in Rational Approximation and its Applications in Mathematics and Physics, eds. J. Gilewicz, M. Pindor and W. Siemaszko, Lecture Notes in Mathematics # 1237, Springer, Berlin, 1987, pp. 313–330.

[4] G. S. Ammar and W. B. Gragg, Numerical experience with a superfast real Toeplitz solver, Linear Algebra Appl., 121 (1989), pp. 185–206.

[5] G. S. Ammar. W. B. Gragg and L. Reichel, On the eigenproblem for orthogonal matrices, in Proceedings 25th IEEE Conference on Decision and Control, Athens, 1986, pp. 1963–1966.

[6] G. S. Ammar, W. B. Gragg, and L. Reichel, Downdating Szegő polynomials and data fitting

applications, Linear Algebra Appl., 172 (1992), pp. 315–336.

[7] G. S. Ammar, W. B. Gragg and L. Reichel, An analogue for Szegő polynomials of the Clenshaw algorithm, J. Comput. Appl. Math., 40 (1993), pp. 211–216.

[8] G. S. Ammar, L. Reichel and D. C. Sorensen, An implementation of a divide and conquer algorithm for the unitary eigenproblem, ACM Trans. Math. Software, 18 (1992), pp. 292–307.

[9] G. S. Ammar, L. Reichel and D. C. Sorensen, Algorithm 730: An implementation of a divide and conquer algorithm for the unitary eigenproblem, ACM Trans. Math. Software, 20 (1994), p. 161.

[10] K. Brumme, Formantenbestimmung in der digitalen Sprachverarbeitung durch LPC - und verwandte Verfahren, Diplomarbeit, Department of Applied Mathematics, University of Hamburg, Hamburg, Germany, 1987.

[11] A. Bunse-Gerstner and L. Elsner, Schur parameter pencils for the solution of the unitary eigenproblem, Linear Algebra Appl., 154-156 (1991), pp. 741–778.

[12] J. A. Cadzow, Foundations of Digital Signal Processing and Data Analysis, MacMillan, New York, 1987.

[13] J. W. Demmel, Trading off parallelism and numerical stability, in Linear Algebra for Large Scale and Real-Time Applications, eds. M. S. Moonen, G. H. Golub and B. L. R. De Moor, Kluwer, Dordrecht, 1993, pp. 49–68.

[14] P. Deuflhard and A. Hohmann, Numerische Mathematik, de Gruyter, Berlin, 1991.

[15] J. J. Dongarra and M. Sidani, A parallel algorithm for the non-symmetric eigenvalue problem, SIAM J. Sci. Comput., 14 (1993), pp. 542–569.

[16] K. Fan and A. J. Hoffman, Some metric inequalities in the space of matrices, Proc. Amer. Math. Soc., 6 (1955), pp. 111–116.

[17] S. Goedecker, Remark on algorithms to find roots of polynomials, SIAM J. Sci. Comput., 15 (1994), pp. 1059–1063.

[18] G. H. Golub and C. F. Van Loan, Matrix Computations, 2nd ed., Johns Hopkins U. P., Baltimore, 1989.

[19] 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, ed. E. S. Nikolaev, Moscow U. P., Moscow, 1982, pp. 16–32. Published in English in revised form in J. Comput. Appl. Math., 46 (1993), pp. 183–198.

[20] W. B. Gragg, The QR algorithm for unitary Hessenberg matrices, J. Comput. Appl. Math., 16 (1986), pp. 1–8.

[21] W. B. Gragg and L. Reichel, A divide and conquer method for the unitary eigenproblem, in Hypercube Multiprocessors 1987, ed. M. T. Heath, SIAM, Philadelphia, 1987, pp. 639–647.

[22] W. B. Gragg and L. Reichel, A divide and conquer method for unitary and orthogonal eigenproblems, Numer. Math., 57 (1990), pp. 695–718.

[23] U. Grenander and G. Szegő, Toeplitz Forms and Their Applications, 2nd ed., Chelsea, New York, 1984.

[24] C. Jagels and L. Reichel, On the construction of Szegő polynomials, J. Comput. Appl. Math., 46 (1993), pp. 241–254.

[25] W. B. Jones, O. Njåstad and E. B. Saff, Szegő polynomials associated with Wiener-Levinson filters, J. Comput. Appl. Math., 32 (1990), pp. 387–406.

[26] W. B. Jones, W. J. Thron, O. Njåstad and H. Waadeland, Szegő polynomials applied to frequency analysis, J. Comput. Appl. Math., 40 (1993), pp. 217–228.

[27] T. Kailath, Linear estimation for stationary and near-stationary processes, in Modern Signal Processing, ed. T. Kailath, Hemisphere, New York, 1985, pp. 59–128.

[28] T. Kailath, Signal processing applications of some moment problems, in Moments in Mathematics, ed. H. J. Landau, Amer. Math. Soc., Providence, 1987, pp. 71–109.

[29] T. Kato, Perturbation Theory for Linear Operators, 2nd ed., Springer, Berlin, 1976.

[30] T.-Y. Li and N. H. Rhee, Homotopy algorithm for symmetric eigenvalue problems, Numer. Math., 55 (1989), pp. 265–280.

[31] T.-Y. Li and Z. Zeng, Homotopy-determinant algorithm for solving nonsymmetric eigenvalue problems, Math. Comp., 59 (1992), pp. 483–502.

[32] T.-Y. Li, H. Zhang and X.-H. Sun, Parallel homotopy algorithm for the symmetric tridiagonal eigenvalue problem, SIAM J. Sci. Stat. Comput., 12 (1991), pp. 469–487.

[33] J. S. Lim, Fundamentals of digital signal processing, in Modern Signal Processing, ed. T. Kailath, Hemisphere, New York, 1985, pp. 1–57.

[34] K. Pan and E. B. Saff, Asymptotics for zeros of Szegő polynomials associated with trigonometric polynomial signals, J. Approx. Theory, 71 (1992), pp. 239–251.

[35] B. N. Parlett and J. Le, Forward instability of tridiagonal QR, SIAM J. Matrix Anal. Appl., 14 (1993), pp. 279–316.

[36] L. Reichel and G. S. Ammar, Fast approximation of dominant harmonics by solving an orthogonal eigenvalue problem, in Mathematics in Signal Processing II, ed. J. G. McWhirter, Clarendon, Oxford, 1990, pp. 575–591.

[37] B. Smith, J. Boyle, Y. Ikebe, V. Klema and C. Moler, Matrix Eigensystem Routines: EISPACK Guide, Springer, Berlin, 2nd ed., 1976.

[38] J. Stoer and R. Bulirsch, Introduction to Numerical Analysis, Springer, New York, 1980.

[39] G. Szegő, Orthogonal Polynomials, 4th ed., Amer. Math. Soc., Providence, 1975.

[40] K.-C. Toh and L. N. Trefethen, Pseudozeros of polynomials and pseudospectra of companion matrices, Numer. Math., 68 (1994), pp. 403–425.