An Analogue for Szegő Polynomials of the Clenshaw Algorithm

Gregory S. Ammar* William B. Gragg† Lothar Reichel‡§

November 21, 1991

Abstract

Linear combinations of polynomials that are orthogonal with respect to an inner product defined on (part of) the real axis are commonly evaluated by the Clenshaw algorithm. We present an analogous algorithm for the evaluation of a linear combination $\sum_{j=0}^{n} \alpha_j \phi_j$ of polynomials ϕ_j that are orthogonal with respect to an inner product defined on (part of) the unit circle. The ϕ_j are known as Szegő polynomials, and find applications, e.g., in signal processing. We also discuss how to express $\sum_{j=0}^{n} \alpha_j \phi_j$ as a linear combination of monomials.

Key Words: Clenshaw algorithm, Szegő polynomial

1 Introduction

Let $\mu(t)$ be a distribution function with infinitely many points of increase in the interval $[-\pi, \pi]$, and define for polynomials p and q the inner product on the unit circle

$$(p, q) := \frac{1}{2\pi} \int_{-\pi}^{\pi} \overline{p(z)} q(z) d\mu(t), \quad z = e^{it}, \tag{1.1}$$

where the bar denotes complex conjugation and $i := \sqrt{-1}$. Introduce the norm $\|p\| := (p, p)^{1/2}$. There is an infinite sequence of polynomials ϕ_j, $j = 0, 1, 2, \ldots$, that are *orthonormal* with respect to (1.1). The ϕ_j are uniquely determined by the requirements that ϕ_j have positive leading coefficient and be of degree j. The ϕ_j are known as *Szegő polynomials*, and their properties are discussed, e.g., in [3, 4, 6, 8].

The orthonormal Szegő polynomials satisfy a recursion relation of the form

$$\begin{aligned} \phi_0(z) &= \tilde{\phi}_0(z) = 1/\sigma_0, & (1.2)\\ \sigma_{j+1}\phi_{j+1}(z) &= z\phi_j(z) + \gamma_{j+1}\tilde{\phi}_j(z), \quad j = 0, 1, 2, \ldots, & (1.3)\\ \sigma_{j+1}\tilde{\phi}_{j+1}(z) &= \bar{\gamma}_{j+1} z\phi_j(z) + \tilde{\phi}_j(z), \quad j = 0, 1, 2, \ldots, & (1.4) \end{aligned}$$

*Northern Illinois University, Department of Mathematical Sciences, DeKalb, IL 60115.
†Naval Postgraduate School, Department of Mathematics, Monterey, CA 93943.
‡Kent State University, Department of Mathematics and Computer Science, Kent, OH 44242.
§Research supported by a National Research Council fellowship and by NSF grant DMS-9002884.

where the recursion coefficients $\gamma_{j+1} \in \mathbb{C}$ and $\sigma_{j+1} > 0$ are determined by

$$\sigma_0 = \delta_0 = (1, 1)^{1/2}, \tag{1.5}$$

$$\gamma_{j+1} = -(1, z\phi_j)/\delta_j, \tag{1.6}$$

$$\sigma_{j+1} = (1 - |\gamma_{j+1}|^2)^{1/2}, \tag{1.7}$$

$$\delta_{j+1} = \delta_j \sigma_{j+1}, \tag{1.8}$$

see, e.g., [3, 8]. It can be verified by induction that $\tilde{\phi}_j(z) = z^j \bar{\phi}_j(1/z)$ for all j, i.e., if $\phi_j(z) = \sum_{k=0}^{j} \beta_k z^k$ then $\tilde{\phi}_j(z) = \sum_{k=0}^{j} \bar{\beta}_{j-k} z^k$. We therefore refer to the $\tilde{\phi}_j$ as *reversed polynomials*. The γ_j are in the literature sometimes referred to as *Schur parameters* or *reflection coefficients*. Our assumption that $\mu(t)$ has infinitely many points of increase implies that $|\gamma_j| < 1$ for $j \geq 1$, and equations (1.2)–(1.8) hold for all $j \geq 0$. Szegő polynomials find applications in signal processing and in the approximation of functions by trigonometric polynomials [1, 3, 4, 7].

The Clenshaw algorithm is a popular method for evaluating finite linear combinations $\sum_{j=0}^{n} \alpha_j \psi_j$ of polynomials ψ_j that satisfy a 3-term recursion relation. The algorithm was introduced by Clenshaw [2] for the case when the ψ_j are Chebyshev polynomials, and was extended by Luke [5] to functions ψ_j that satisfy a higher order difference equation. A recent discussion of the algorithm is given by Wimp [9].

The present paper describes an algorithm for the evaluation of linear combinations of Szegő polynomials

$$s_n(z) := \sum_{j=0}^{n} \alpha_j \phi_j(z) \tag{1.9}$$

from the recursion coefficients $\{\gamma_j\}_{j=1}^{n}$ and $\{\sigma_j\}_{j=0}^{n}$. One method, which is presented in [7], is to use the Szegő recursions (1.3)–(1.4) to compute the values of the ϕ_j at z, and to use these values in the evaluation of (1.9). The algorithm presented here is analogous with the Clenshaw algorithm, and is obtained by repeatedly expressing the Szegő polynomial of highest degree in terms of lower degree polynomials using the recursion relation for the ϕ_j. Advantages of the new algorithm over the algorithm in [7] include (i) the polynomials ϕ_j do not have to be evaluated explicitly, and (ii) the coefficients α_j enter the computation in the order of decreasing j. If $|\alpha_j|$ decreases rapidly with j, then the algorithm may yield a very small propagated round-off error; see [5, 9] for an analysis.

Our algorithm for the evaluation of linear combinations of Szegő polynomials (1.9) is presented in Section 2. In Section 3 we use this scheme to derive an algorithm for expressing (1.9) as a linear combination of monomials. Such a change of polynomial basis may be of advantage if (1.9) is to be evaluated at many points allocated equidistantly on a circle. This evaluation can be arrived at rapidly with the fast Fourier transform (FFT) algorithm when (1.9) is expressed in terms of monomials. Section 4 contains computed examples.

2 The algorithm

Let $z \in \mathbb{C}$ be fixed and define the coefficients $\tau_k = \tau_k(z)$ and $\tilde{\tau}_k = \tilde{\tau}_k(z)$ by

$$s_n(z) = \sum_{j=0}^{k-1} \alpha_j \phi_j(z) + \tau_k \sigma_k \phi_k(z) + \tilde{\tau}_k \sigma_k \tilde{\phi}_k(z), \quad 0 \le k \le n\,.$$

Then $k = n$ yields

$$\alpha_n \phi_n(z) = \tau_n \sigma_n \phi_n(z) + \tilde{\tau}_n \sigma_n \tilde{\phi}_n(z),$$

i.e., $\tau_n = \alpha_n/\sigma_n$ and $\tilde{\tau}_n = 0$. Moreover, letting $k = 0$, we obtain, in view of (1.2), that

$$s_n(z) = \tau_0 \sigma_0 \phi_0(z) + \tilde{\tau}_0 \sigma_0 \tilde{\phi}_0(z) = \tau_0 + \tilde{\tau}_0\,.$$

Our analogue of the Clenshaw algorithm is then obtained from the recursion relations for τ_k and $\tilde{\tau}_k$, which are easily derived from recursion formulas (1.3)–(1.4) for the ϕ_j and $\tilde{\phi}_j$.

Algorithm 2.1 (Evaluation of (1.9) – an analogue of the Clenshaw algorithm)

Input: z, n, $\{\gamma_j\}_{j=1}^n$, $\{\sigma_j\}_{j=0}^n$, $\{\alpha_j\}_{j=0}^n$;

Output: $s_n(z)$;

$\tau_n := \alpha_n/\sigma_n$; $\tilde{\tau}_n := 0$;

for $k := n-1, n-2, \ldots, 0$ **do**

$\quad \tau_k := \sigma_k^{-1}(\alpha_k + z(\tau_{k+1} + \bar{\gamma}_{k+1}\tilde{\tau}_{k+1}))$;

$\quad \tilde{\tau}_k := \sigma_k^{-1}(\gamma_{k+1}\tau_{k+1} + \tilde{\tau}_{k+1})$;

$s_n(z) := \tau_0 + \tilde{\tau}_0$; □

3 Change of basis

It can be desirable to express $s_n(z)$ defined by (1.9) as a linear combination of monomials

$$s_n(z) = \sum_{j=0}^{n} \beta_j z^j, \tag{3.1}$$

because this may allow the evaluation of $s_n(z)$ by the FFT algorithm. We show how Algorithm 2.1 can be modified so as to allow the computation of the coefficients β_j from the α_j in representation (1.9).

Introduce the vectors $\boldsymbol{\tau}_k = [\tau_{k0}, \tau_{k1}, \ldots, \tau_{k,n-k}]^T$ and $\tilde{\boldsymbol{\tau}}_k = [\tilde{\tau}_{k0}, \tilde{\tau}_{k1}, \ldots, \tilde{\tau}_{k,n-k}]^T$, $0 \leq k \leq n$. We let τ_{kj} and $\tilde{\tau}_{kj}$ be coefficients of z^j, and obtain from the recursions of Algorithm 2.1 the following scheme for determining the expansion (3.1) from (1.9).

Algorithm 3.1 (Change of basis – Szegő polynomials to monomials)

Input: n, $\{\gamma_j\}_{j=1}^n$, $\{\sigma_j\}_{j=0}^n$, $\{\alpha_j\}_{j=0}^n$;

Output: $\{\beta_j\}_{j=0}^n$;

$\boldsymbol{\tau}_n := \alpha_n/\sigma_n$; $\tilde{\boldsymbol{\tau}}_n := 0$;

for $k := n-1, n-2, \ldots, 0$ **do**

$$\boldsymbol{\tau}_k := \sigma_k^{-1}\left(\begin{bmatrix} \alpha_k \\ \boldsymbol{\tau}_{k+1} \end{bmatrix} + \bar{\gamma}_{k+1}\begin{bmatrix} 0 \\ \tilde{\boldsymbol{\tau}}_{k+1} \end{bmatrix}\right);$$

$$\tilde{\boldsymbol{\tau}}_k := \sigma_k^{-1}\left(\gamma_{k+1}\begin{bmatrix} \tilde{\boldsymbol{\tau}}_{k+1} \\ 0 \end{bmatrix} + \begin{bmatrix} \tilde{\boldsymbol{\tau}}_{k+1} \\ 0 \end{bmatrix}\right);$$

for $k := 0, 1, \ldots, n$ **do**

$\beta_k := \tau_{0k} + \tilde{\tau}_{0k}$; □

4 Computed examples

We now present some examples to illustrate the numerical behavior of our analogue of the Clenshaw algorithm (Algorithm 2.1). We compare the accuracy of Algorithm 2.1 with that of Algorithm 4.2 in [7], which uses the Szegő recursions to explicitly compute the values of the ϕ_j in the evaluation of (1.9). Each experiment was carried out in FORTRAN 77 on a SparcStation SLC, on which there are approximately 7 and 16 significant decimal digits in single-precision and double-precision calculations, respectively.

For each experiment, we input $n := 100$ Schur parameters γ_j with $|\gamma_j| < 1$ and n coefficients α_j, $1 \leq j \leq n$, and evaluate (1.9) at the $5n$ roots of unity using a single-precision implementation of Algorithm 2.1. We also perform the $5n$ evaluations using a single-precision implementation of Algorithm 4.2 in [7]. Then, using the results of the latter algorithm in double-precision arithmetic as exact answers, we compute the maximum relative error among the $5n$ values computed by each single-precision algorithm.

Table 1 shows the results when the Schur parameters are constant, $\gamma_j \equiv \rho$, and the coefficients α_j are randomly generated uniformly distributed real numbers with $|\alpha_j| < j^{-\nu}$. For a particular choice of ρ and ν, we perform the experiment 20 times, and calculate the average of the maximum relative error obtained using Algorithm 2.1 (denoted by C, for Clenshaw, in the tables), and for Algorithm 4.2 of [7] (denoted by S, for Szegő, in the tables). Also displayed in the tables, under the heading "better," is the ratio of times that the maximum relative error for Algorithm C was smaller than that of Algorithm S. This experiment was performed for various values of ρ and ν. Observe that Algorithm C usually achieved a smaller maximum relative error than Algorithm S, and that the averages for Algorithm C were consistently smaller that those of Algorithm S.

Table 2 shows the results of the same set of experiments, except that the Schur parameters were chosen to be $\gamma_j = \rho^j$. The results here are similar to those in Table 1, except that the average maximum relative error for Algorithm C was larger that that of Algorithm S in one example ($\rho = 0.9$, $\nu = 2$).

Finally, Table 3 shows the results when the Schur parameters were randomly generated complex numbers with $|\gamma_j| < \rho$, i.e., their magnitudes are uniformly distributed in $[0, \rho)$ and their arguments are uniformly distributed in $[0, 2\pi)$. In these examples, the average maximum relative error for Algorithm C was smaller than that of Algorithm S in only six of the twelve runs.

These experimental results indicate that Algorithm 2.1, which utilizes the same idea as Clenshaw's algorithm, often provides a more accurate evaluation of (1.9) than Algorithm 4.2 of [7], which explicitly evaluates the Szegő polynomials, when the coefficients α_j tend to zero as j increases.

References

[1] G.S. Ammar, W.B. Gragg and L. Reichel, Downdating of Szegő polynomials and applications to data fitting, Report, Kent State University, Department of Mathematics and Computer Science, Kent, OH, 1991.

[2] C.W. Clenshaw, A note on the summation of Chebyshev series, *MTAC* **9** (1955) 118–120.

[3] U. Grenander and G. Szegő, *Toeplitz Forms and Their Applications* (Chelsea, New York, NY, 1984).

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

[5] Y.L. Luke, *Mathematical Functions and Their Approximation* (Academic Press, New York, NY, 1975).

[6] H.N. Mhaskar and E.B. Saff, On the distribution of zeros of polynomials orthogonal on the unit circle, *J. Approx. Theory* **63** (1990) 30–38.

[7] L. Reichel, G.S. Ammar and W.B. Gragg, Discrete least squares approximation by trigonometric polynomials, *Math. Comp.* **57** (1991) 273–289.

[8] G. Szegő, *Orthogonal Polynomials*, 4th ed. (Amer. Math. Soc., Providence, RI, 1975).

[9] J. Wimp, *Computation with Recurrence Relations* (Pitman, Boston, MA, 1984).

Table 1: Average maximum relative errors over 20 runs

$\gamma_j \equiv \rho$; randomly generated $\alpha_j \in \mathbb{R}$ with $|\alpha_j| < j^{-\nu}$.

	$\nu = 1$			$\nu = 2$			$\nu = 3$		
ρ	S	C	better	S	C	better	S	C	better
0.40	5.12e–06	3.72e–06	17/20	3.11e–06	2.31e–06	16/20	2.84e–06	1.85e–06	19/20
0.80	2.34e–05	9.83e–06	19/20	1.05e–05	8.38e–06	16/20	1.03e–05	5.25e–06	20/20
0.90	1.24e–04	1.43e–05	20/20	1.33e–04	1.47e–05	20/20	2.97e–05	5.99e–06	20/20
0.99	1.66e–04	4.04e–05	20/20	1.60e–04	4.84e–05	20/20	1.63e–04	4.29e–05	20/20

Table 2: Average maximum relative errors over 20 runs

$\gamma_j := \rho^j$; randomly generated $\alpha_j \in \mathbb{R}$ with $|\alpha_j| < j^{-\nu}$.

	$\nu = 1$			$\nu = 2$			$\nu = 3$		
ρ	S	C	better	S	C	better	S	C	better
0.40	2.44e–06	1.83e–06	14/20	9.44e–07	2.95e–07	20/20	7.89e–07	1.24e–07	20/20
0.80	2.68e–06	2.17e–06	15/20	9.68e–07	6.70e–07	18/20	9.61e–07	2.49e–07	20/20
0.90	3.41e–06	2.78e–06	10/20	2.50e–06	2.83e–06	11/20	1.13e–06	1.01e–06	16/20
0.99	1.05e–05	9.59e–06	13/20	7.11e–06	6.83e–06	14/20	6.56e–06	5.97e–06	11/20

Table 3: Average maximum relative errors over 20 runs

randomly generated $\gamma_j \in \mathbb{C}$ with $|\gamma_j| < \rho$; randomly generated $\alpha_j \in \mathbb{R}$ with $|\alpha_j| < j^{-\nu}$.

	$\nu = 1$			$\nu = 2$			$\nu = 3$		
ρ	S	C	better	S	C	better	S	C	better
0.40	7.77e–06	9.22e–06	7/20	1.25e–06	5.62e–07	19/20	9.02e–07	1.45e–07	20/20
0.80	8.53e–04	5.26e–04	12/20	1.59e–04	1.65e–04	10/20	5.43e–05	4.41e–05	15/20
0.90	4.71e–04	3.77e–04	12/20	4.10e–04	4.10e–04	7/20	2.72e–04	2.78e–04	7/20
0.99	8.97e–04	6.70e–04	16/20	9.64e–04	1.16e–03	12/20	7.16e–04	7.25e–04	7/20