NUMERICAL ANALYSIS PROJECT
MANUSCRIPT NA-91-06

NOVEMBER 1991

How to generate unknown orthogonal polynomials out of known orthogonal polynomials

by

Gene H. Golub
Bernd Fischer

NUMERICAL ANALYSIS PROJECT
COMPUTER SCIENCE DEPARTMENT
STANFORD UNIVERSITY
STANFORD, CALIFORNIA 94305

How to generate unknown orthogonal polynomials out of known orthogonal polynomials

BERND FISCHER

Institute of Applied Mathematics
University of Hamburg
2000 Hamburg 13, F.R.G.

and

GENE H. GOLUB

Department of Computer Science
Stanford University
Stanford, CA *94305,* U.S.A.

Abstract

We consider the problem of generating the three-term recursion coefficients of orthogonal polynomials for a weight function $v(t) = r(t)w(t)$, obtained by modifying a given weight function w by a rational function r. Algorithms for the construction of the orthogonal polynomials for the new weight v in terms of those for the old weight w axe presented. All the methods are based on modified moments. As applications we present Gaussian quadrature rules for integrals in which the integrand has singularities close to the interval of integration, and the generation of orthogonal polynomials for the (finite) Hermite weight e^{-t^2}, supported on a finite interval $[-b, b]$.

1980 Mathematics Subject Classification. Primary 65F25; Secondary 33A65.
The second author was in part supported by the US Army under grant DAAL 03-90-G-0105.

1. Introduction

Let w be a nonnegative weight function on [a, b], $a < b$. With w there is associated a system of orthogonal polynomials $\{p_j\}$, where p_j has exact degree j and

$$(1.1) \qquad < p_j, p_i >_w = \int_a^b p_j(t)p_i(t)w(t)dt = \begin{cases} > 0 \text{ if } i = j \\ = 0 \text{ if } i \neq j \end{cases}$$

They satisfy the three-term recurrence relation

$$(1.2) \qquad \begin{aligned} c_j p_j(t) &= (t - a_j)p_{j-1}(t) - b_j p_{j-2}(t), \quad j = 1, 2, \ldots \\ p_{-1}(t) &\equiv 0, \quad p_0(t) \equiv p_0, \end{aligned}$$

where b_j , $c_j > 0$ for $j \geq 1$ (b_1 is arbitrary).

Let π_l and $\hat{\pi}_m$ be given real polynomials of degree l and m, respectively, having no root in common. Assume furthermore that the resulting rational function

$$(1.3) \qquad r(t) = \frac{\pi_l(t)}{\hat{\pi}_m(t)}$$

is nonnegative on $[a, b]$. *Now* consider the new weight function

$$(1.4) \qquad v(t) = r(t)w(t).$$

Clearly, there exists a set of polynomials $\{\psi_j\}$ that are orthogonal with respect to $< \cdot, \cdot >_v$ (cf. (1.1)). In this paper we investigate the problem of numerically generating the recurrence coefficients α_j and β_j in the relation

$$(1.5) \qquad \begin{aligned} \gamma_j \psi_j(t) &= (x - \alpha_j)\psi_{j-1} - \beta_j \psi_{j-2}, \quad j = 1, 2, \ldots, n \\ \psi_{-1}(t) &\equiv 0, \quad \psi_0(t) \equiv \psi_0 \end{aligned}$$

under the assumption that the coefficients c_j, a_j, and b_j for whatever value of j is required, and the zero-order moment

$$\nu_0(w) = \int_a^b w(t)dt$$

are given. Note that the γ_j's > 0 are free parameters.

In order to compute the desired quantities we "break up" the given problem into small pieces. To this end let t_i, $i = 1, 2, \ldots$, and $z_j = x_j \pm iy_j$, $j = 1, 2, \ldots$ denote the real and complex poles of r, then

$$\begin{aligned} \int_a^b v(t)dt &= \int_a^b r(t)w(t)dt \\ &= \int_a^b \left(q(t) + \sum_i \frac{A_i}{t - t_i} + \sum_j \frac{B_j t + C_j}{(t - x_j)^2 + y_j^2} \right) w(t)dt, \end{aligned}$$

where q is a real polynomial. Hence, our procedure consists of repeated application of the following two elementary steps (I) and (II): compute the three-term recurrence coefficients of the orthogonal polynomials relative to

$$\text{(I)} \qquad v(t) = q(t)w(t),$$

$$\text{(II)} \qquad v(t) = \frac{w(t)}{t - x}, \text{ or } v(t) = \frac{w(t)}{(t-x)^2 + y^2}$$

in terms of those for the old weight function w. Of course, one could further break up the problem by writing q as a product of linear and quadratic factors (which requires the knowledge of the zeros of q). This is not necessary for the implementation of our algorithm which "directly" deals with $v(t) = q(t)w(t)$.

The final step, namely the problem of computing orthogonal polynomials associated with

$$v(t) = \sum_i v_i(t),$$

assuming we know the orthogonal polynomials relative to the given weight functions v_i, has been solved (even for the case, that the v_i's have different support) in a recent paper by Fischer and Golub [5].

The problem (I) is solved by a classical formula due to Christoffel (cf. Szegö [15, Thm. 2.5]), whereas an explicit solution of (II), for the most general case of weight functions of the form (1.4), is due to Uvarov [17], [18]. However, in both cases the resulting polynomials are expressed in determinantal form. Thus they are hardly useful for computational purposes.

Based on a work of Galant [6] and by making use of the fact that here the orthogonal polynomials are explicitly given in terms of the so-called "kernel polynomials" (cf. Chihara [3, Ch I, §7]), Gautschi [9] came up with a scheme for (I) for the special case of multiplying w by a linear factor or by a quadratic factor. In the same paper he derived algorithms for (II) by "inverting" those obtained for (I).

Kautsky and Golub [12] devised algorithms for (I) via a suitable modification of the Jacobi matrix relative to w by either a Lanczos type method or one step of an implicit QR method.

In this note we follow a different (classical) approach. All our algorithms are based on the so-called modified moments

$$\text{(1.6)} \qquad \nu_l = \nu_l(p_l, v) = < p_l, 1 >_v = \int_a^b p_l(t)v(t)dt.$$

It is well-known (see e.g. Wheeler[19]), that the quantities ν_l and the three term recurrence coefficients of p_l determine the desired orthogonal polynomials relative to v. The underlying procedure is known as the modified Chebyshev algorithm.

Hence, from this point of view, the solution of our problems basically comes down to the evaluation of suitable modified moments.

The paper is organized as follows. In Section 2 we briefly describe a (basic) modified Chebyshev algorithm. The solution of (I) is carried out in Section 3. In Section 4 we discuss two different approaches for the solution of problem (II). The fist one directly implements the computation of the corresponding modified moments ν_l, whereas the other is based on a suitable inversion of the process developed in §3. Finally, a number of examples illustrating the numerical performance of the various methods are given in Section 5. The examples include an application to Gaussian quadrature rules for integrals in which the integrand has singularities close to the interval of integration, and the generation of orthogonal polynomials for the (finite) Hermite weight e^{-t^2}, supported on a finite interval *[-b, b]*.

2. The modified Chebyshev algorithm

In this section we present a basic description of the modified Chebyshev algorithm. The goal is to compute the three-term recurrence coefficients of a system of orthogonal polynomials $\{\psi_j\}$ (cf. (1.5))

$$\begin{aligned} &\gamma_j\psi_j(t) = (x-\alpha_j)\psi_{j-1} - \beta_j\psi_{j-2}, \quad j = 1,2,\ldots, \\ &\psi_{-1}(t) \equiv 0, \quad \psi_0(t) \equiv \psi_0, \end{aligned} \tag{2.1}$$

relative to a given inner product

$$< f, g >_v = \int_a^b f(t)g(t)v(t)dt$$

associated with the nonnegative weight function v. The algorithm involves the modified moments

$$\nu_l = \nu_l(p_l, v) = < p_l, 1 >_v = \int_a^b p_l(t)v(t)dt, \quad l = 0,1,\ldots, \tag{2.2}$$

corresponding to v and a system of polynomials $\{p_k\}$. Here we arrive at an efficient algorithm, if we assume that this system also satisfies a three-term recurrence relation (cf. (1.2))

$$\begin{aligned} &c_j p_j(t) = (t-a_j)p_{j-1}(t) - b_j p_{j-2}(t), \quad j = 1,2,\ldots, \\ &p_{-1}(t) \equiv 0, \ p_0(t) \equiv p_0. \end{aligned} \tag{2.3}$$

We note that the case of ordinary moments, i.e., $p_k(t) = t^k$, was first treated by Chebyshev [2].

The desired coefficients can be conveniently computed in terms of the "mixed moment" matrix $R = [\sigma_{k,l}]$ (see e.g. Wheeler [19]), where

$$\sigma_{k,l} = <\psi_k, p_l>_v = \int_a^b \psi_k(t)p_l(t)v(t)dt, \quad k,l = 0,1,\ldots. \tag{2.4}$$

The key equations for the computation of the α_j's and β_j's are readily obtained from the two recurrence relations (2.1) and (2.3), respectively,

$$\begin{aligned}
\beta_{k+1} &= c_k \frac{\sigma_{k,k}}{\sigma_{k-1,k-1}}, \\
\alpha_{k+1} &= a_{k+1} + c_{k+1}\frac{\sigma_{k,k+1}}{\sigma_{k,k}} - \beta_{k+1}\frac{\sigma_{k-1,k}}{\sigma_{k,k}} \\
&= a_{k+1} + c_{k+1}\frac{\sigma_{k,k+1}}{\sigma_{k,k}} - c_k\frac{\sigma_{k-1,k}}{\sigma_{k-1,k-1}}, \\
\sigma_{k,l} &= \frac{1}{\gamma_k}[-\beta_k\sigma_{k-2,l} + b_{l+1}\sigma_{k-1,l-1} + \\
&\qquad (a_{l+1} - \alpha_k)\sigma_{k-1,l} + c_{l+1}\sigma_{k-1,l+1}].
\end{aligned} \tag{2.5}$$

Thus, updating the element $\sigma_{k,l}$ involves (in general) four other entries in R

$$\begin{array}{ccccc}
 & & 0 & & \\
 & & \sigma_{k-2,l} & & \\
 & 0 & 0 & 0 & \\
 & \sigma_{k-1,l-1} & \sigma_{k-1,l} & \sigma_{k-1,l+1} & \\
 & & \bullet & & \\
 & & \sigma_{k,l} & &
\end{array} \tag{2.6}$$

In order to start the algorithm we have to compute the modified moments

$$\nu_l = \frac{\sigma_{0,l}}{\psi_0} = \int_a^b p_l(t)v(t)dt. \tag{2.7}$$

The next sections are devoted to the efficient computation of these quantities. Note that equation (2.7) is the only occasion where the weight function v enters into the picture. The rest of the scheme depends only on the polynomials (2.3) (compare also Algorithm 1).

3. Solution of problem (I)

In this section we apply the modified Chebyshev algorithm to the construction of the orthogonal polynomials $\{\psi_j\}_{j=0}^n$ relative to

$$v(t) = \pi_m(t)w(t), \tag{3.1}$$

where π_m is a real polynomial (positive on [a, b]) of degree m and w a given non-negative weight function. Again, let $\{p_j\}$ denote the orthogonal polynomial for w (cf. (1.2) and (2.3)). As already pointed out in the previous Section we first have to compute the modified moments $\nu_l(p_l, v)$ (cf. (2.7)). To this end express the given polynomial π_m in terms of the basis provided by the p_j

$$\pi_m(t) = \sum_{j=0}^{m} \tau_j p_j(t).$$

Consequently, we obtain by (3.1), (2.7) and (1.1)

$$\begin{aligned} \nu_l &= \int_a^b p_l(t)\pi_m(t)w(t)dt = \sum_{j=0}^{m} \tau_j \int_a^b p_l(t)p_j(t)w(t)dt \\ &= \begin{cases} \tau_l \int_a^b p_l^2(t)w(t)dt & \text{if } l \le m \\ 0 & \text{if } l > m. \end{cases} \end{aligned} \tag{3.2}$$

It is well-known (see, e.g., Chihara [3, p.22]) that the normalization constant $\|p_l\|_w^2 = \int_a^b p_l^2(t)w(t)dt$ is easily computed from

$$\|p_l\|_w^2 = p_0^2 \nu_0(w) \prod_{j=2}^{l+1} \frac{b_j}{c_{j-1}}, \tag{3.3}$$

where

$$\nu_0(w) = \int_a^b w(t)dt \tag{3.4}$$

denotes the zero-order moment.

Note that the upper triangular matrix R (cf. (2.4)) has bandwidth m+l. This can be exploited in the algorithm, e.g., for $n = 4$ and $m = 2$ it follows directly from (2.6) that only the entries marked by a '$\star$' have to be computed in terms of the other indicated entries:

$k\backslash l$	0	1	2	3	4	5
-1		0	0	0		
0	ν_0	ν_1	ν_2	0	0	
1		$\star$	$\star$	$\star$	0	0
2			$\star$	$\star$	$\star$	0
3				$\star$	$\star$	

Finally, we arrive at:

Algorithm 1. Given *a weight function* w *on* [a, b], *the* associated *orthogonal polynomials* p_i, *the zero-order moment* $\nu_0(w)$, *and* a *real* polynomial π_m *(positive*

in [a, *b*]), this *algorithm computes the orthogonal polynomials* ψ_k, k = 0, 1, . . . , n *relative to* $v(t) = \pi_m(t)w(t)$.

Initialize.

Set $\sigma_{-1,l} = 0,\ l = 1, 2 \ldots, \min\{2n-2, m+1\}$ and $\sigma_{k,m+k+j} = 0$ (if $m + k + j < 2n - k - 1$), $k = 1, 2, \ldots, n-1,\ j = 1, 2$

- **choose** $\psi_0 > 0$
- **compute** $\sigma_{0,l} = \psi_0 \nu_l, l = 0,1, \ldots, \min\{2n-1, m\}$ 식 (3.2), (3.3) and (3.4)
- **compute** α_1 **by (2.5)**

$$\alpha_1 = a_1 + c_1 \frac{\sigma_{0,1}}{\sigma_{0,0}}$$

Iterate.

For $k = 1, 2, \ldots, n-1$ **do**

- **choose** $\gamma_k > 0$
- compute $\sigma_{k,l}$, β_{k+1}, and α_{k+1} by (2.5)

  for $l = k,\ k+1, \ldots, \min\{2n-1-k,\ m+k\}$ **do**

$$\sigma_{k,l} = \frac{1}{\gamma_k}\left[-\beta_k \sigma_{k-2,l} + b_{l+1}\sigma_{k-1,l-1} + (a_{l+1} - \alpha_k)\sigma_{k-1,l} + c_{l+1}\sigma_{k-1,l+1}\right]$$

  end

$$\beta_{k+1} = c_k \frac{\sigma_{k,k}}{\sigma_{k-1,k-1}},$$

$$\alpha_{k+1} = a_{k+1} + c_{k+1}\frac{\sigma_{k,k+1}}{\sigma_{k,k}} - c_k \frac{\sigma_{k-1,k}}{\sigma_{k-1,k-1}}$$

end

End.

Remarks. 1. The algorithm requires as input j_{max} **recursion coefficients** a_j, b_j, **and** c_j, j = 1, 2, . . . , j_{max}, where

$$j_{max} = \max_{k=1,2,\ldots,n-1} \min\{2n-1-k, m+k\}$$

2. It is straightforward to generalize the scheme for polynomials π_m **with complex coefficients.**

3. Gautschi [9] devised an algorithm for the special case $m = 1$, **i.e.,** $v(t) = (t - x)w(t)$. **His approach is based on the observation that here the desired polynomials, the so-called kernel polynomials, are explicitly given in terms of the "old"** polynomials p_k

$$\psi_k(t) = \frac{1}{t-x}\left[p_{k+1}(t) - \frac{p_{k+1}(x)}{p_k(x)}p_k(t)\right].$$

And by a repeated application of the complex version of his procedure, first with the linear factor $t - z$ and second with $t - \bar{z}$, he solved (in real arithmetic) the case of quadratic factors $(t - x)^2 + y^2$, where $z = x + iy$.

4. Kautsky and Golub [12] obtained different algorithms by directly attacking the Jacobi matrix associated with the p_j's.

4. Solution of problem (II)

In this chapter we present two algorithms for the computation of orthogonal polynomials $\{\psi_j\}_{j=0}^n$ relative to weight functions v obtained by modifying w by a linear divisor

$$v(t) = \frac{w(t)}{t - x} \tag{4.1}$$

or by a quadratic divisor

$$v(t) = \frac{w(t)}{(t - x)^2 + y^2} = \frac{w(t)}{(t - z)(t - \bar{z})} = -\frac{1}{y} \operatorname{Im}\left(\frac{w(t)}{t - z}\right), \tag{4.2}$$

where $z = x + iy$, $y > 0$.

The first procedure follows the lines of Section 3. That is, we are tempted to compute the modified moments $\nu_l(p_l, v)$ (cf. (2.2)) associated with the weights (4.1) and (4.2), respectively. One approach for accomplishing these tasks similarly is, in view of the last equation of (4.2), the evaluation of the complex integral

$$I_l(z) = \int_a^b \frac{p_l(t)}{t - z} w(t)\,dt, \quad z \in \mathbb{C} \setminus [a, b], \quad l = 0, 1, \ldots, \tag{4.3}$$

which seems to be at first glance a quite tricky problem. However, quite an efficient algorithm (cf. Gautschi [7]) is based on the observation that $I_0(z)$ has a (convergent) continued fraction expansion in terms of the three-term recurrence coefficients of the orthogonal polynomial p_l associated with w. By exploiting this property one arrives at the following (backward recurrence) algorithm.

Algorithm 2. *Given* **a weight** *function w on [a, b], the associated orthogonal polynomials p_j, the zero-order moment $\nu_0(w)$, a point $z \in C \setminus [a, b]$, and an error tolerance ε. Then this algorithm computes the quantities $I_l(z)$, $l = 0, 1, \ldots, n$, within a relative error of ε.*

Initialize.

set $c_0 = 1$, $b_1 = \nu_0(w)$, **and** $I_{-1}(z) = -1$

- **choose** N > n

Iterate.

until convergence do

- set $\rho_N = 0$
- **for** $j =$ N, N $- 1, \ldots, 0$ **do** $\rho_{j-1} = c_j b_{j+1}/(z - a_{j+1} - \rho_j)$
- **for** j = O,l,. . . , n **do** $I_j(z) = \rho_{j-1} I_{j-1}(z)$
- **test convergence**

$$\textbf{if} \quad \max_{j=0,1,\ldots,n} \frac{|I_j(z) - I_j^{old}(z)|}{|I_j(z)|} < \varepsilon \quad \textbf{stop.}$$

- **increase** N

end

End.

Remarks. 1. The algorithm is known to converge (cf. Gautschi [7, Thm. 3.1]). 2. For the scheme to be effective it is critical to have good estimates for the "starting index" N. Such estimates are known for some common weight functions, e.g., the Jacobi weights (cf. Gautschi [8, §5]). It is worth noticing that, in general, N decreases as z moves away from [a, *b]* (compare §5.2).

The second scheme implements a suitable "inversion" of Algorithm 1. A similar technique was used by Gautschi [9]. 0 ur derivation, however, appears to us more transparent than the one given in [9].

The trick is to assume that we already know the desired polynomials ψ_k and then proceed as for the solution of problem (I) by "artifically generating" the orthogonal polynomials p_k associated with (compare (4.1) and (4.2))

$$w(t) = (t - x)v(t) \quad \text{or} \quad w(t) = ((t - x)^2 + y^2)v(t)$$

via the mixed moment matrix $\hat{R} = [\hat{\sigma}_{k,l}]$

$$\hat{\sigma}_{k,l} = < p_k, \psi_l >_w = \int_a^b p_k(t)\psi_l(t)w(t)dt. \tag{4.4}$$

First we consider the case of a linear divisor, i.e., w(t) = $(t - x)v(t)$. Here $\hat{R}$ (cf. (4.4)) is an upper triangular matrix which has zero entries apart from the diagonal and super-diagonal. Therefore, we define for convenience

$$\hat{d}_k = \hat{\sigma}_{k,k} \quad \text{and} \quad \hat{e}_k = \hat{\sigma}_{k,k+1}.$$

Now the essential parts of Algorithm 1 (compare also (2.5)) read (after interchanging the roles of the "Greek and Latin coefficients")

$$\hat{d}_k = \frac{1}{c_k}\left[\beta_{k+1}\hat{d}_{k-1} + (\alpha_{k+1} - a_k)\hat{e}_{k-1}\right] \tag{1}$$

$$\hat{e}_k = \frac{1}{c_k}\beta_{k+2}\hat{e}_{k-1} \tag{2}$$

$$b_{k+1} = \gamma_k \frac{\hat{d}_k}{\hat{d}_{k-1}} \tag{3}$$

$$a_{k+1} = \alpha_{k+1} + \gamma_{k+1}\frac{\hat{e}_k}{\hat{d}_k} - b_{k+1}\frac{\hat{e}_{k-1}}{\hat{d}_k}. \tag{4}$$

Now let β_{k+1} and α_{k+1} become subject of these equations

$$\hat{d}_k = \frac{1}{\gamma_k} b_{k+1}\hat{d}_{k-1} \tag{3}$$

$$\beta_{k+1} = c_{k-1}\frac{\hat{e}_{k-1}}{\hat{e}_{k-2}} \tag{2}$$

$$\alpha_{k+1} = a_k + c_k\frac{\hat{d}_k}{\hat{e}_{k-1}} - \beta_{k+1}\frac{\hat{d}_{k-1}}{\hat{e}_{k-1}} \tag{1}$$

$$\hat{e}_k = \frac{1}{\gamma_{k+1}}\left[b_{k+1}\,\hat{e}_{k-1} + (a_{k+1} - \alpha_{k+1})\hat{d}_k\right]. \tag{4}$$

In order to start the algorithm we need to compute the quantities $\hat{e}_0$ and β_2 or, equivalently, α_1 and β_2. It turns out to be advantagegous to first compute α_1 "by hand". This can be done by observing that the Fourier expansion of $t\psi_n(t)$ in terms of ψ_j compared with the identities (2.1) immediately yields explicit expressions for the recursion coefficients

$$\alpha_j = \frac{< t\psi_{j-1}, \psi_{j-1} >_v}{< \psi_{j-1}, \psi_{j-1} >_v} \quad \text{and} \quad \beta_j = \gamma_{j-1}\frac{< \psi_{j-1}, \psi_{j-1} >_v}{< \psi_{j-2}, {}^{*j-2} >_v}. \tag{4.5}$$

In particular we have (recall $v(t) = w(t)/(t - x)$)

$$\begin{aligned} \alpha_1 &= \frac{< t\psi_0, \psi_0 >_v}{< \psi_0, \psi_0 >_v} = \frac{1}{\nu_0(v)}\int_a^b \left(\frac{x}{t-x} + 1\right) w(t)dt \\ &= x + \frac{\nu_0(w)}{\nu_0(v)} = x + K. \end{aligned} \tag{4.6}$$

Likewise, we obtain for β_2 in view of (2.1)

$$\begin{aligned} \beta_2 &= \gamma_1\frac{< \psi_1, \psi_1 >_v}{< \psi_0, \psi_0 >_v} = \frac{1}{\gamma_1\nu_0(v)}\int_a^b (t - \alpha_1)v(t)dt \\ &= \frac{1}{\gamma_1\nu_0(v)}\int_a^b \left[((t-x) - 2K)w(t) + K^2 v(t)\right] = \frac{K}{\gamma_1}(a_1 - \alpha_1), \end{aligned} \tag{4.7}$$

where we have used that (cf. (4.5) for w instead of v)

$$\int_a^b tw(t)dt = a_1\nu_0(w) \tag{4.8}$$

holds.

Finally, we arrive at the following algorithm:

Algorithm 3. *Given a weight function w on [a, b], the associated orthogonal polynomials p_j, $j = 0,1,\ldots n$, a number $x \notin (a, b)$, and the zero order moments $\nu_0(w)$ and $\nu_0(v)$, respectively, where $v(t) = w(t)/(t - x)$. Then this algorithm computes the orthogonal polynomials ψ_k, $k = 0, 1, \ldots, n$, relative to v.*

Initialize.

- choose $\psi_0 > 0$ and $\gamma_1 > 0$
- compute by (4.4), (4.6), and (4.7)

$$\hat{d}_0 = \psi_0 p_0 \nu_0(w), \quad \alpha_1 = x + \frac{\nu_0(w)}{\nu_0(v)} = x + K$$

$$\hat{e}_0 = \frac{\hat{d}_0}{\gamma_1}(a_1 - \alpha_1), \quad \beta_2 = \frac{K}{\gamma_1}(a_1 - \alpha_1)$$

Iterate.

For $k = 1, 2, \ldots, n-1$ do

- choose $\gamma_k > 0$
- compute

$$\hat{d}_k = \frac{1}{\gamma_k} b_{k+1}\hat{d}_{k-1}$$

$$\text{if } k > 1 \quad \beta_{k+1} = c_{k-1}\frac{\hat{e}_{k-1}}{\hat{e}_{k-2}}$$

$$\alpha_{k+1} = a_k + c_k\frac{\hat{d}_k}{\hat{e}_{k-1}} - \beta_{k+1}\frac{\hat{d}_{k-1}}{\hat{e}_{k-1}}$$

$$\text{if } k < n-1 \quad \hat{e}_k = \frac{1}{\gamma_{k+1}}\left[b_{k+1}\hat{e}_{k-1} + (a_{k+1} - \alpha_{k+1})\hat{d}_k\right]$$

end

End.

Remarks. 1. The zero-order moment $\nu_0(v)$ can be computed by **Algorithm** 2 (with $n = 0$) for $x \notin [a, b]$.

2. The algorithm generalizes in the obvious way to complex x.

3. The algorithm becomes unstable as x moves away from the support [a, *b]* (compare §5.2).

Now we at tack the problem of generating the orthogonal polynomials ψ_k relative to

$$v(t) = \frac{w(t)}{(t-x)^2+y^2} = \frac{w(t)}{(t-z)(t-\bar{z})}, \quad z = x + \mathrm{i}y, \quad y > 0.$$

One obvious strategy would be the repeated application of Algorithm 3 onto $v_1(t) = w(t)/(t-z)$ and $v_2(t) = w(t)/(t-\bar{z})$, respectively. Gautschi [9, §5.2] designed an algorithm along this lines. We do not follow this idea. We proceed analogously to the case of a linear divisor by "inverting" the Algorithm 1 with respect to $w(t) = [(t-x)^2 + y^2]v(t)$.

Here the corresponding x-nixed moment matrix $\hat{R}$ (cf. (4.4)) has bandwidth three and hence we define

$$\hat{d}_k = \hat{\sigma}_{k,k}, \quad \hat{e}_k = \hat{\sigma}_{k,k+1}, \quad \text{and} \quad \hat{f}_k = \hat{\sigma}_{k,k+2}.$$

This time the key equations of Algorithm 1 (compare also (2.5)) are

$$\hat{d}_k = \frac{1}{c_k}(-b_k f_{k-2} + \beta_{k+1}\hat{d}_{k-1} + (\alpha_{k+1} - a_k)\hat{e}_{k-1} + \gamma_{k+1}\hat{f}_{k-1})$$

$$\hat{e}_k = \frac{1}{c_k}(\beta_{k+2}\hat{e}_{k-1} + (\alpha_{k+2} - a_k)\hat{f}_{k-1})$$

$$\hat{f}_k = \frac{1}{c_k}\beta_{k+3}\hat{f}_{k-1}$$

$$b_{k+1} = \gamma_k \frac{d_k}{\hat{d}_{k-1}}$$

$$a_{k+1} = \alpha_{k+1} + \gamma_{k+1}\frac{\hat{e}_k}{\hat{d}_k} - b_{k+1}\frac{\hat{e}_{k-1}}{\hat{d}_k}$$

Again, we have to interchange the role of the Greek and Latin coefficients and to compute some "starting values". Here we need to know the quantities $\alpha_1, \alpha_2, \beta_2$, and β_3 beforehand. They can be computed by some (lengthy) routine computations based on the explicit representations (4.5). We omit the details. Note, that in view of (4.2) and (4.3), the zero-order moment

$$\nu_0(v) = \int_a^b v(t)dt = \frac{1}{y}\mathrm{Im}\left(\int_a^b \frac{w(t)}{t-z}dt\right) = \frac{1}{y}\mathrm{Im}(I_0(z))$$

can, again, be evaluated by means of Algorithm 2 (with *n = 0).*

Algorithm 4. ***Given a weight function w on [a, b], the associated orthogonal polynomials p_j, $j = 0, 1, \ldots n$, a complex number $z = x$ + iy, y > 0, the zero-order moment $\nu_0(w)$, and the integral $I_0(z)$. Then this algorithm computes the orthogonal polynomials ψ_k, $k = 0, 1, \ldots, n$ relative to v(t) $= w(t)/((t-x)^2 + y^2)$.***

Initialize.

- choose $\psi_0 > 0$ and $\gamma_1, \gamma_2 > 0$
- **compute**

$$\hat{d}_0 = \psi_0 p_0 \nu_0(w), \quad \alpha_1 = x + y\frac{\mathrm{Re}I_0(z)}{\mathrm{Im}I_0(z)}$$

$$\hat{e}_0 = \frac{\hat{d}_0}{\gamma_1}(a_1 - \alpha_1), \quad \beta_2 = \frac{1}{\gamma_1}\left[\frac{\nu_0(w)}{\mathrm{Im}I_0(z)} - (x - \alpha_1)^2 - y^2)\right]$$

$$\alpha_2 = \frac{a_1 - \alpha_1}{\beta_2\gamma_1} y\frac{\nu_0(w)}{\mathrm{Im}I_0(z)} + 2x - \alpha_1, \hat{f}_0 = \frac{c_1\hat{e}_1 - \beta_2\hat{e}_0}{\alpha_2 - a \quad l}$$

$$\beta_3 = \frac{b_2c_1 - (a_1 - \alpha_1)^2 - \beta_2\gamma_1}{\beta_2\gamma_1\gamma_2} y\frac{\nu_0(w)}{\mathrm{Im}I_0(z)} + \frac{1}{2}\left[(x - \alpha_1)^2 - (x - \alpha_2)^2\right]$$

Iterate.

For $k = 1, 2, \ldots, n-1$ **do**

- **if** $k > 2$ **choose** $\gamma_k > 0$
- **compute**

$$\hat{d}_k = \frac{1}{\gamma_k}\hat{d}_{k-1}b_{k+1}$$

$$\text{if} \quad k > 2 \quad \beta_{k+1} = c_{k-2}\frac{\hat{f}_{k-2}}{\hat{f}_{k-3}}$$

$$\text{if} \quad k > 1 \quad \alpha_{k+1} = a_{k-1} + c_{k-1}\frac{\hat{e}_{k-1}}{\hat{f}_{k-2}} - \beta_{k+1}\frac{\hat{e}_{k-2}}{\hat{f}_{k-2}}$$

$$\text{if} \quad k > 2 \quad \hat{f}_{k-1} = \frac{1}{\gamma_{k+1}}\left[c_k\hat{d}_k + b_k\hat{f}_{k-2} - \beta_{k+1}\hat{d}_{k-1} + (a_k - \alpha_{k+1})\hat{e}_{k-1}\right]$$

$$\hat{e}_k = \frac{1}{\gamma_{k+1}}\left[b_{k+1}\hat{e}_{k-1} + (a_{k+1} - \alpha_{k+1})\hat{d}_k\right].$$

end

End.

Remarks. 1. The algorithm computes the three (non-zero) diagonals of $\hat{R}$ columnwise according to the following star (compare (2.6))

$$\begin{array}{ccc} & 0 & \\ & \hat{\sigma}_{k-2,k} & \\ 0 & 0 & \bullet \\ \hat{\sigma}_{k-1,k-1} & \hat{\sigma}_{k-1,k} & \hat{\sigma}_{k-1,k+1} \\ & 0 & \\ & \hat{\sigma}_{k,k} & \end{array}$$

2. The algorithm becomes unstable as z moves away from the support [a, *b*].

5. Numerical Examples

In this Section we report on some experiments with the derived methods. All computations were carried out in MATLAB (approx. 16 significant decimal places).

1. ***Jacobi weights.***

The Jacobi weights

$$w^{(\alpha,\beta)}(t) = (1-t)^{\alpha}(1+t)^{\beta} \text{ on } [-1, 1], \text{ where } \alpha, \beta > -1, \tag{5.1}$$

are perfectly suited for testing purposes in the present context. The recursion coefficients of the associated Jacobi polynomials are explicitly known (see, e.g., Chihara [3, p.220]).

We checked the reliability of the algorithms by recovering the Jacobi polynomials relative to $w^{(\alpha+1,\beta-1)}$ via suitable modifications of $w^{(\alpha,\beta)}$

$$w^{(\alpha+1,\beta-1)} = \frac{1-t}{1+t}\, w^{(\alpha,\beta)}.$$

More precisely, we first computed the orthogonal polynomial relative to $v(t) = (1-t)w^{(\alpha,\beta)}$ by Algorithm 1 and then used the obtained recursion coefficients for the calculation of the orthogonal polynomials relative to $v(t)/(1+t)$ via Algorithm 3. We did these computations for orthogonal polynomials up to degree $n = 100$ and for various values of α and β. The observed absolute errors of the computed three-term recursion coefficients (compared with known coefficients of the Jacobi polynomials) were always below $6 * 10^{-16}$, i.e., the algorithms appear (in this context) to be quite numerically stable.

2. ***Integration in the presence of nearby singularities, a Schwarz-Christoffel problem.***

In this Section we demonstrate how to numerically integrate a function which has a singularity very close to the interval of integration. The idea is to absorb (at least) part of the singularity into the weight function and then to apply a suitable Gaussian quadrature rule.

Problems of this type typically arise in the context of Schwarz-Christoffel mappings, i.e., any conformal map of the unit disk or the upper half-plane onto any simply-connected polygonal region can be represented as

$$f(z) = C_1 \int^{z} \prod_{j=1}^{m} (t - z_j)^{\sigma_j}\, dt + C_2$$

where the numbers $\sigma_j \in (-1, 1]$ correspond to the angles at the vertices of the given polygonal region and the z_j's are the "prevertices" on the unit circle or the real axis (see, e.g., Trefethen [16]).

Here, we consider the conformal map of the upper half plane onto a rectangle

$$f(z) = C_1 \int_0^z \frac{dt}{\sqrt{(1-t^2)(1-k^2t^2)}} + C_2.$$

As a subproblem one has to evaluate

$$K(k) = \int_0^1 \frac{dt}{\sqrt{(1-t^2)(1-k^2t^2)}}, \tag{5.2}$$

the complete elliptic integral of the first kind with modulus $k \in (0,1)$. This integral can be readily evaluated (up to machine precision) by means of the so-called arithmetic-geometric mean (AGM) method (cf. Abramowitz and Stegun [1, §17.6]). The values obtained by the AGM iteration will later on serve as reference values.

Now let us demonstrate how to use to advantage Gaussian quadrature for the computation of the integral $K(k)$. To this end we rewrite $K(k)$ as

$$K(k) = \frac{2}{k} \int_{-1}^1 \left[(1-t)(3+t)\left(\frac{2-k}{k} - t\right)\left(\frac{2+k}{k} + t\right)\right]^{-1/2} dt. \tag{5.3}$$

Note, that the singularity at $t = (2-k)/k$, moves towards 1 as k tends to 1.

In order to take care of the singularity at the endpoint of the interval of integration $(1-t)^{-1/2}$ we apply a Gaussian quadrature rule based on the orthogonal polynomials relative to the Jacobi weight $w^{(-1/2,0)}$, i.e.,

$$K(k) \sim \frac{2}{k} \nu_0(w) \sum_{j=1}^n \tau_j \left[(3+\lambda_j)\left(\frac{2-k}{k} - \lambda_j\right)\left(\frac{2+k}{k} + \lambda_j\right)\right]^{-1/2}. \tag{5.4}$$

The nodes λ_j and weights τ_j can be efficiently and accurately computed by a method due to Golub and Welsch [10].

However, if the singularity $t = (2-k)/k$ is close to the endpoint 1 the Gauß-Jacobi rule *(5.4)* is very inaccurate (compare Table 5.1). In order to overcome this difficulty we incorporated the troublesome singularity into the weight function, i.e., we computed the Gaussian quadrature relative to the weight function

$$v(t) = \frac{w^{(-1/2,0)}}{\frac{2-k}{k} - t}. \tag{5.5}$$

The Gauß rule now reads

$$K(k) \sim \frac{2}{k} \nu_0(w) \sum_{j=1}^n \hat{\tau}_j \frac{\sqrt{\frac{2-k}{k} - \hat{\lambda}_j}}{\sqrt{(3+\hat{\lambda}_j)\left(\frac{2+k}{k} + \hat{\lambda}_j\right)}}. \tag{5.6}$$

We generated the orthogonal polynomials with respect to v both by **Algorithm 3** and by **Algorithm 2** (in conjunction with the modified Chebyshev algorithm). As long as the singularity $t = (2 - k)/k$ is moderately close to 1 (roughly for $k > .7$) both methods yield the same results. For small k **Algorithm 3** becomes unstable (in contrast to **Algorithm 2**). It is worth noticing that, however, the Gaussian quadrature rule (based on **Algorithm 3**) still produces remarkable good approximations to K(k), even in cases where some of the computed nodes were located outside the interval of integration.

The starting index N for **Algorithm 2** was estimated according to a formula due Gautschi [8], that is, we take the smallest integer N satisfying

$$N = n + \frac{\log(1/eps)}{2\log(r + \sqrt{r^2 - 1})}, \quad r = (2 - k)/k,$$

where *eps* = 2.22.. . $* 10^{-16}$ is the machine precision (in MATLAB).

In the following table we present the absolute errors (compared with the results of the AGM method) for the two rules (5.4) and (5.6) and various values of n and k.

	n=5		n=10		n=15	
k	(5.4)	(5.6)	(5.4)	(5.6)	(5.4)	(5.6)
0.70	-13.2	-16.5	-25.8	-29.4	-36.0	-34.9
0.75	-11.8	-14.8	-23.1	-26.6	-34.3	-34.7
0.80	-10.3	-13.2	-20.2	-23.8	-30.0	-34.7
0.85	-8.70	-11.5	-17.2	-20.7	-25.6	-29.4
0.90	-6.91	-9.65	-13.8	-17.2	-20.5	-24.3
0.95	-4.66	-7.24	-9.52	-12.8	-14.3	-17.9
0.99	-1.69	-3.77	-3.94	-6.67	-6.12	-9.23

Table 5.1. $\log_{10}$(errors) **in n-point Gaussian quadrature applied to the elliptic integral** ***K(k)***

We observe that rule (5.6) outperforms rule (5.4) by (at least) 3 decimal places of accuracy.

3. Generation of orthogonal polynomials for "complicated" weight functions

In this Section we show how to apply the derived methods for the computation of the orthogonal polynomials for the (finite) Hermite weight

$$w(t) = e^{-t^2}, \quad t \in [-b, b], \quad \text{where } b > 0. \tag{5.7}$$

Integrals involving this weight

$$\int_{-b}^{b} f(t)w(t)dt, \tag{5.8}$$

arise, for example, in quantum chemistry calculations (see, e.g., Chin [4] and references therein).

The generation of the Gaussian quadrature rule for (5.8) was treated by several authors, see, e.g., Piessens and Branders [14], King and Dubois [13], and Chin [4]. Their algonthms directly deal with the weight function (5.7). Here we follow a different approach. The idea is to approximate the given weight function (5.7) by polynomials and/or rational functions and then to compute the orthogonal polynomials relative to the (much easier to handle) best approxiamtions. Of course, the success of this approach partly depends on the ability to approximate (5.7) as close as possible. In order to achieve this goal we subdivided the given interval

$$[-b, b] = [-b_1, -a_1] \cup [-b_2, -a_2] \cup [a_1, b_1] \cup [a_2, b_2]$$

into smaller intervals and then computed a compound Chebyshev approximation. More precisely, for the case $b = 1$ we computed the best Chebyshev approximation $P_{2m}^{(i)}$

$$\max_{t\in[a_i,b_i]} |e^{-t^2} - P_{2m}^{(i)}(t)| = \min_{\tau_j \in \mathbb{R}} \max_{t\in[a_i,b_i]} |e^{-t^2} - \sum_{j=0}^{2m} \tau_j t^j|, \quad i = 1, 2, \tag{5.9}$$

where

$$[a_1, b_1] = [0, \sqrt{2}/2] \quad \text{and} \quad [a_2, b_2] = [\sqrt{2}/2, 1].$$

Note that $\sqrt{2}/2$ is the point of inflection of e^{-t^2}. Furthermore, observe that for the symmetric weight function $w(t) = w(-t)$ on the symmetric region $[-b_i, -a_i] \cup [a_i, b_i]$ the best Chebyshev approximation simplifies to a polynomial in t^2, i.e.,

$$P_{2m}^{(i)}(t) = P_m^{(i)}(t^2) = \sum_{j=0}^{m} \tau_j^{(i)} L_{2j}(t), \quad i = 1, 2, \tag{5.10}$$

where L_j is the jth Legendre polynomial on [-1, 1]. The best approximations were computed by using the Remez - algorithm (see, e.g., Golub and Smith [11] for a robust implementation) in terms of the basis provided by the Legendre polynomials (cf. (5.10)). For $m = 8$ we obtained for the minimal deviation

$$\max_{t\in[a_i,b_i]} |e^{-t^2} - P_{2m}^{(i)}(t)| < 10^{-14}, \quad i = 1, 2.$$

In the next step we computed the orthogonal polynomials relative to the "new" weight functions $w^{(i)}(t) = \sum_{j=0}^{m} \tau_j^{(i)} L_{2j}(t)$ on $[-b_i, -a_i] \cup [a_i, b_i]$, $i = 1, 2$, via Algorithm 1. Note that the corresponding modified moments (cf. (3.2) and (3.3)) are given by

$$\nu_j^{(i)}(L_j, w^{(i)}) = \begin{cases} 2\tau_j^{(i)}/(2j+1) & \text{for } j = 2k \text{ and } k \leq m \\ 0 & \text{otherwise .} \end{cases}$$

Finally we applied the methods developed in [5] for the computation of the orthonormal polynomials ψ_n relative to the weight function $w^{(12)} = w^{(1)} + w^{(2)}$, more precisely

$$w^{(12)}(t) = \begin{cases} w^{(1)}(t) & \text{for } a_1 \leq |t| \leq b_1 \\ w^{(2)}(t) & \text{for } a_2 \leq |t| \leq b_2. \end{cases}$$

In order to illustrate the effectiveness of the method, we computed the orthonormal polynomials up to degree $n = 100$. Some of the computed recurrence coefficients are shown in Table 5.2. Notice, that the symmetry of the problem implies

$$\beta_j \psi_j(t) = x\psi_{j-1}(t) - \beta_{j-1}\psi_{j-2}(t), j = 1, 2, \ldots$$

We compared the computed coefficients with the one listed in Chin [4]. The coinciding digits are printed in boldface.

k	β_k
1	**0.50369048 688442**
10	**0.500695856** 88288
20	**0.50016449692** 124
30	**0.50007184023** 437
40	**0.50004006396** 199
50	**0.50002551002** 673
60	**0.500017655244** 05
70	**0.500012939877** 80
80	**0.50000988918** 692
90	**0.50000780271012**
100	**0.5000063131058 3**

Table 5.2. Coefficient of the kth orthogonal polynomial relative to $w^{(12)}$

Table 5.2 clearly shows that the described algorithm is perfectly stable for the case $b = 1$. As it is not surprising, the approximation of the weight function (5.7) is more delicate for $b > 1$ than for $b = 1$. We will report on numerical experiments for the case $b > 1$ elsewhere.

Acknowledgement

Part of this work was done while the first author was visiting the Department of Computer Science at the Stanford University. He would like to thank Gene Golub for his hospitality.

References

1. **M. Abramowitz and I. Stegun, Eds., Handbook of mathematical functions, Dover Publications, 1965.**
2. **P. L. Chebyshev, Sur l'interpolation par la méthode des moindres carrés, Mém. Acad. Impér. Sci. St. Pétersbourg (7), no. 15 (1859), 1-24. [Oeuvres I, 473-498].**
3. **T. S. Chihara, An introduction to orthogonal polynomials, Gordon and Breach, New York, 1978.**
4. **R. C. Y. Chin, A domain decomposition method for generating orthogonal polynomials for a Gaussian weight on a finite interval, J. Comp. Phys. , to appear.**
5. **B. Fischer and G. H. Golub, On generating polynomials which are orthogonal over several intervals, Math. Comp. , 56 (1991), 711-730.**
6. **D. Galant, An implementation of Christoffel's theorem in the theory of orthogonal polynomials, Math. Comp. , 25 (1971), 111-113.**
7. **W. Gautschi, Computational aspects of three-term recurrence relations, SIAM Rev. , 9 (1967), 24-82.**
8. **W. Gautschi, Minimal solutions of three-term recurrence relations and orthogonal polynomials, Math. Comp. , 36 (1981), 547-554.**
9. **W. Gautschi, An algorithmic implementation of the generalized Christoffel theorem, in Numerical Integration (G. Hämmerlin, ed.), Internat. Ser. Numer. Math., 57 (1982), 89-106, Birkhäuser, Basel.**
10. **G. H. Golub and J.H. Welsch, Calculation of Gauss quadrature rules, Math. Comp. , 23 (1969), 221-230.**
11. **G. H. Golub and L. B. Smith,.Chebyshev approximation of continuous functions by a Chebyshev system of functions, Algorithm 414, CACM, 14 (1971), 737-746.**
12. **J. Kautsky and G. H. Golub, On the calculation of Jacobi matrices, Linear Alg. Appl. , 52/53 (1983), 439-455.**
13. **H. F. King and M. Dupuis, Numerical integration using Rys polynomials, J. Comp. Phys. , 21 (1976), 144-165.**
14. **R. Piessens and M. Branders, Table of Gaussian Quadrature formulas, Appl. Math. Prog. Div., University of Leuven, Leuven, 1975.**
15. **G. Szegö, Orthogonal polynomials, AMS Colloquium Publications 23, Revised ed., American Mathematical Society, New York, *1959.***

16. **L.N. Trefethen, Numerical computation of the Schwarz-Christoffel transformation, SIAM J. Sci. Stat. Comput. , 1 (1980), 82-102.**
17. **V.B. Uvarov, Relation between polynomials orthogonal with different weights (Russian), Dokl. Akad. Nauk SSSR, 126 (1959), 33-36.**
18. **V.B. Uvarov, The connection between systems of polynomials that are orthogonal with respect to different distribution functions, U.S.S.R. Computational Math. and Phys. , 9 (1969), pp. 25-36.**
19. **J.C. Wheeler, Modified moments and Gaussian quadrature, Rocky Mt. J. Math., 4 (1974), 287 - 296.**