Numerical Algorithms **25:** 213–222, 2000.
© 2000 *Kluwer Academic Publishers. Printed in the Netherlands.*

High-order Gauss–Lobatto formulae

Walter Gautschi

Purdue University

Received 10 April 1999; accepted in revised form 3 June 2000

Dedicated, in friendship and with high regard, to Richard S. Varga on the occasion of his 70th birthday

Currently, the method of choice for computing the $(n+2)$-point Gauss–Lobatto quadrature rule for any measure of integration is to first generate the Jacobi matrix of order $n+2$ for the measure at hand, then modify the three elements at the right lower corner of the matrix in a manner proposed in 1973 by Golub, and finally compute the eigenvalues and first components of the respective eigenvectors to produce the nodes and weights of the quadrature rule. In general, this works quite well, but when n becomes large, underflow problems cause the method to fail, at least in the software implementation provided by us in 1994. The reason is the singularity (caused by underflow) of the 2×2 system of linear equations that is used to compute the modified matrix elements. It is shown here that in the case of arbitrary Jacobi measures, these elements can be computed directly, without solving a linear system, thus allowing the method to function for essentially unrestricted values of n. In addition, it is shown that all weights of the quadrature rule can also be computed explicitly, which not only obviates the need to compute eigenvectors, but also provides better accuracy. Numerical comparisons are made to illustrate the effectiveness of this new implementation of the method.

1. Introduction

Given a positive measure $\mathrm{d}\lambda$ supported on the interval $[-1, 1]$ of the real line $\mathbb{R}$, and assuming all its moments to exist, one can consider a quadrature rule of the form

$$\int_{-1}^{1} f(t)\,\mathrm{d}\lambda(t) = \lambda_0 f(-1) + \sum_{k=1}^{n} \lambda_k f(t_k) + \lambda_{n+1} f(1) + R_n(f) \tag{1.1}$$

that is exact whenever f is a polynomial of degree $\leqslant 2n+1$,

$$R_n(f) = 0 \quad \text{if } f \in \mathbb{P}_{2n+1}. \tag{1.2}$$

It is called the ($(n+2)$-point) *Gauss–Lobatto rule* relative to the measure $\mathrm{d}\lambda$. As is well known, the interior nodes t_k are the zeros of $\pi_n(\,\cdot\,; \mathrm{d}\lambda_{\pm 1})$, the polynomial of degree n orthogonal with respect to the modified measure $\mathrm{d}\lambda_{\pm 1}(t) = (1-t^2)\,\mathrm{d}\lambda(t)$, and the weights λ_j can be obtained by interpolation at the nodes $-1, t_1, \dots, t_n, 1$.

A more elegant constructive procedure has been proposed by Golub [4]. It is based on a modified Jacobi matrix, the tridiagonal matrix of order $n + 2$

$$J^*_{n+2}(\mathrm{d}\lambda) = \begin{bmatrix} \alpha_0 & \sqrt{\beta_1} & & & & 0 \\ \sqrt{\beta_1} & \alpha_1 & \sqrt{\beta_2} & & & \\ & \sqrt{\beta_2} & \ddots & \ddots & & \\ & & \ddots & \alpha_{n-1} & \sqrt{\beta_n} & \\ & & & \sqrt{\beta_n} & \alpha_n & \sqrt{\beta^*_{n+1}} \\ 0 & & & & \sqrt{\beta^*_{n+1}} & \alpha^*_{n+1} \end{bmatrix}, \tag{1.3}$$

where α_k, β_k are the coefficients in the recurrence relation

$$\begin{aligned} &\pi_{k+1}(t) = (t - \alpha_k)\pi_k(t) - \beta_k\pi_{k-1}(t), \quad k = 0, 1, 2, \dots, \\ &\pi_{-1}(t) = 0, \qquad \pi_0(t) = 1 \end{aligned} \tag{1.4}$$

for the (monic) orthogonal polynomials $\pi_k(\,\cdot\,; \mathrm{d}\lambda)$, and α^*_{n+1}, β^*_{n+1} the solution of the 2×2 system of linear equations

$$\begin{bmatrix} \pi_{n+1}(-1) & \pi_n(-1) \\ \pi_{n+1}(1) & \pi_n(1) \end{bmatrix} \begin{bmatrix} \alpha^*_{n+1} \\ \beta^*_{n+1} \end{bmatrix} = \begin{bmatrix} -\pi_{n+1}(-1) \\ \pi_{n+1}(1) \end{bmatrix}. \tag{1.5}$$

The nodes of (1.1) (including ± 1) are then the eigenvalues of $J^*_{n+2}(\mathrm{d}\lambda)$, and the weights λ_j are obtainable in terms of the first components $v_{j,1}$ of the associated normalized eigenvectors v_j by means of

$$\lambda_j = \beta_0 v^2_{j,1}, \quad j = 0, 1, 2, \dots, n, n + 1, \tag{1.6}$$

where $\beta_0 = \int_{-1}^{1} \mathrm{d}\lambda(t)$ (cf. also [3]).

This works quite well as long as n is not too large. When n becomes large, the elements of the matrix in (1.5) become very small, and their products even smaller, so much so that when computing the determinant,

$$\Delta_n = \pi_{n+1}(-1)\pi_n(1) - \pi_{n+1}(1)\pi_n(-1), \tag{1.7}$$

both its terms underflow, creating a singular system. For the Legendre measure $\mathrm{d}\lambda(t) = \mathrm{d}t$, and single-precision IEEE arithmetic, this happens beginning with $n = 79$, and in double precision, beginning with $n = 543$.

One way of correcting the problem is to rescale the orthogonal polynomials. A better way is to compute the α^*_{n+1}, β^*_{n+1}, whenever possible, directly without solving a linear system. We show that this is indeed possible if $\mathrm{d}\lambda$ is the Jacobi measure

$$\mathrm{d}\lambda^{(\alpha,\beta)}(t) = (1 - t)^\alpha(1 + t)^\beta\,\mathrm{d}t, \quad \alpha > -1,\ \beta > -1. \tag{1.8}$$

In the case of the Legendre measure $\mathrm{d}\lambda(t) = \mathrm{d}\lambda^{(0,0)}$, it is known, furthermore, that the weights λ_j in (1.1) are expressible explicitly in terms of the Legendre polynomial P_{n+1} as follows (cf., e.g., [1, p. 104]):

$$\begin{aligned}\lambda_0 = \lambda_{n+1} &= \frac{2}{(n+1)(n+2)},\\ \lambda_k &= \frac{2}{(n+1)(n+2)[P_{n+1}(t_k)]^2}, \quad k = 1, 2, \dots, n \quad (\mathrm{d}\lambda(t) = \mathrm{d}t).\end{aligned} \tag{1.9}$$

We observed with our software [2] that the weights computed directly by (1.9) are, in general, rather more accurate than those produced via (1.6). We will show that analogous expressions for the weights, with similar advantages in accuracy, can be had for the Jacobi measure $\mathrm{d}\lambda^{(\alpha,\beta)}$.

2. The modified Jacobi matrix

We denote the monic polynomials orthogonal with respect to the Jacobi measure (1.8) by $\pi_k = \pi_k^{(\alpha,\beta)}$, and the Jacobi polynomials, as conventionally defined (cf., e.g., [5]) by $P_k = P_k^{(\alpha,\beta)}$. It is well known that

$$P_n(t) = k_n \pi_n(t), \quad k_n = \frac{1}{2^n}\binom{2n+\alpha+\beta}{n}, \tag{2.1}$$

and that

$$P_n(1) = \binom{n+\alpha}{n}, \qquad P_n(-1) = (-1)^n \binom{n+\beta}{n}. \tag{2.2}$$

The determinant in (1.7) then becomes

$$\Delta_n = \frac{1}{k_n k_{n+1}} D_n, \tag{2.3}$$

where

$$D_n = \begin{vmatrix} P_{n+1}(-1) & P_n(-1) \\ P_{n+1}(1) & P_n(1) \end{vmatrix}. \tag{2.4}$$

Using (2.2), one finds

$$D_n = \frac{(-1)^{n+1}}{n+1}(2n+\alpha+\beta+2)\binom{n+\alpha}{n}\binom{n+\beta}{n}, \tag{2.5}$$

and from (1.5) and (2.1) one gets

$$\begin{aligned}\alpha^*_{n+1} &= \frac{-P_{n+1}(-1)P_n(1) - P_{n+1}(1)P_n(-1)}{D_n},\\ \beta^*_{n+1} &= 2\frac{k_n}{k_{n+1}}\frac{P_{n+1}(-1)P_{n+1}(1)}{D_n}.\end{aligned}$$

Substituting (2.2) and (2.5) in these formulae yields

$$\begin{aligned} \alpha^*_{n+1} &= \frac{\alpha-\beta}{2n+\alpha+\beta+2}, \\ \beta^*_{n+1} &= 4\,\frac{(n+\alpha+1)(n+\beta+1)(n+\alpha+\beta+1)}{(2n+\alpha+\beta+1)(2n+\alpha+\beta+2)^2}. \end{aligned} \tag{2.6}$$

Since the recursion coefficients α_k, β_k for the Jacobi measure $\mathrm{d}\lambda^{(\alpha,\beta)}$ are explicitly known, the Gauss–Lobatto formula for $\mathrm{d}\lambda^{(\alpha,\beta)}$ is now readily computable via eigenvalues and (first components of) eigenvectors of the matrix (1.3). Accuracy, however, is improved if the weights are computed separately, bypassing (1.6). The next two sections develop the necessary formulae.

3. The boundary weights

Our concern from now on is with the Gauss–Jacobi–Lobatto formula

$$\int_{-1}^{1} f(t)\,\mathrm{d}\lambda^{(\alpha,\beta)}(t) = \lambda_0 f(-1) + \sum_{k=1}^{n} \lambda_k f(t_k) + \lambda_{n+1} f(1) + R_n(f). \tag{3.1}$$

We first deal with the boundary weights $\lambda_0 = \lambda_0^{(\alpha,\beta)}$ and $\lambda_{n+1} = \lambda_{n+1}^{(\alpha,\beta)}$. Since the change of variable $t \mapsto -t$ converts $\mathrm{d}\lambda^{(\alpha,\beta)}$ into $\mathrm{d}\lambda^{(\beta,\alpha)}$, one easily sees that $\lambda_{n+1}^{(\alpha,\beta)} = \lambda_0^{(\beta,\alpha)}$. It suffices therefore to compute $\lambda_0^{(\alpha,\beta)}$.

We recall that the interior nodes $t_k = t_k^{(\alpha,\beta)}$ in (3.1) are the zeros of $\pi_n(\,\cdot\,;\mathrm{d}\lambda_{\pm1}^{(\alpha,\beta)})$, where $\mathrm{d}\lambda_{\pm1}^{(\alpha,\beta)}(t) = (1-t^2)\,\mathrm{d}\lambda^{(\alpha,\beta)}(t) = \mathrm{d}\lambda^{(\alpha+1,\beta+1)}(t)$, thus the zeros of $P_n^{(\alpha+1,\beta+1)}$,

$$P_n^{(\alpha+1,\beta+1)}(t_k) = 0, \quad k = 1, 2, \ldots, n. \tag{3.2}$$

Since [5, equation (4.21.7)]

$$\frac{\mathrm{d}}{\mathrm{d}t}\bigl\{P_{n+1}^{(\alpha,\beta)}(t)\bigr\} = \frac{1}{2}(n+\alpha+\beta+2)P_n^{(\alpha+1,\beta+1)}(t), \tag{3.3}$$

they are also the zeros of $P_{n+1}^{(\alpha,\beta)\prime}$,

$$P_{n+1}^{(\alpha,\beta)\prime}(t_k) = 0, \quad k = 1, 2, \ldots, n. \tag{3.4}$$

Now putting $f(t) = (1-t)P_n^{(\alpha+1,\beta+1)}(t)$ in (3.1), and observing (3.2), yields

$$2\lambda_0 P_n^{(\alpha+1,\beta+1)}(-1) = \int_{-1}^{1} P_n^{(\alpha+1,\beta+1)}(t)\,\mathrm{d}\lambda^{(\alpha+1,\beta)}(t). \tag{3.5}$$

The integrand on the right, by the second relation in [5, equation (4.5.4)] (with α replaced by $\alpha+1$), can be written as

$$P_n^{(\alpha+1,\beta+1)}(t) = \frac{2}{2n+\alpha+\beta+3}\,\frac{(n+1)P_{n+1}^{(\alpha+1,\beta)}(t) + (n+\beta+1)P_n^{(\alpha+1,\beta)}(t)}{t+1}. \tag{3.6}$$

Noting from the second relation in (2.2) that

$$\begin{aligned} n+1 &= (-1)^n \frac{n+1}{\binom{n+\beta}{n}} P_n^{(\alpha+1,\beta)}(-1), \\ n+\beta+1 &= -(-1)^n \frac{n+1}{\binom{n+\beta}{n}} P_{n+1}^{(\alpha+1,\beta)}(-1), \end{aligned}$$

we can rewrite (3.6) as

$$\begin{aligned} P_n^{(\alpha+1,\beta+1)}(t) = {} & \frac{2}{2n+\alpha+\beta+3}\,\frac{(-1)^n(n+1)}{\binom{n+\beta}{n}} \\ & \times \frac{P_{n+1}^{(\alpha+1,\beta)}(t)P_n^{(\alpha+1,\beta)}(-1) - P_n^{(\alpha+1,\beta)}(t)P_{n+1}^{(\alpha+1,\beta)}(-1)}{t+1}. \end{aligned}$$

This calls for the application of the Christoffel–Darboux formula ([5, equation (4.5.2)] with α replaced by $\alpha+1$), which gives

$$\begin{aligned} P_n^{(\alpha+1,\beta+1)}(t) = {} & \frac{2}{2n+\alpha+\beta+3}\,\frac{(-1)^n(n+1)}{\binom{n+\beta}{n}}\, c_n(\alpha,\beta) \\ & \times \sum_{\nu=0}^{n} \frac{1}{h_\nu^{(\alpha+1,\beta)}}\, P_\nu^{(\alpha+1,\beta)}(t)P_\nu^{(\alpha+1,\beta)}(-1), \end{aligned} \tag{3.7}$$

where $h_\nu^{(\alpha+1,\beta)} = \int_{-1}^{1}[P_\nu^{(\alpha+1,\beta)}(t)]^2\,\mathrm{d}\lambda^{(\alpha+1,\beta)}(t)$ and

$$c_n(\alpha,\beta) = 2^{\alpha+\beta+1}(2n+\alpha+\beta+3)\,\frac{\Gamma(n+\alpha+2)\Gamma(n+\beta+1)}{\Gamma(n+2)\Gamma(n+\alpha+\beta+3)}. \tag{3.8}$$

Integration of (3.7) with the measure $\mathrm{d}\lambda^{(\alpha+1,\beta)}$ produces, by the orthogonality of the Jacobi polynomials,

$$\int_{-1}^{1} P_n^{(\alpha+1,\beta+1)}(t)\,\mathrm{d}\lambda^{(\alpha+1,\beta)}(t) = \frac{2}{2n+\alpha+\beta+3}\,\frac{(-1)^n(n+1)}{\binom{n+\beta}{n}}\, c_n(\alpha,\beta). \tag{3.9}$$

Therefore, the desired result now follows from (3.5) together with the second relation in (2.2) (with β replaced by $\beta+1$) and (3.9), (3.8), to read

$$\lambda_0^{(\alpha,\beta)} = 2^{\alpha+\beta+1}\,\frac{\Gamma(\alpha+2)\Gamma(\beta+1)}{\Gamma(\alpha+\beta+3)}\,\frac{\binom{n+\alpha+1}{n}}{\binom{n+\beta+1}{n}\binom{n+\alpha+\beta+2}{n}}. \tag{3.10}$$

As mentioned at the beginning of this section,

$$\lambda_{n+1}^{(\alpha,\beta)} = \lambda_0^{(\beta,\alpha)}. \tag{3.11}$$

In the ultraspherical case, (3.10) and (3.11) simplify to

$$\lambda_0^{(\alpha,\alpha)} = \lambda_{n+1}^{(\alpha,\alpha)} = 2^{2\alpha+1}\,\frac{\Gamma^2(\alpha+2)}{(\alpha+1)\Gamma(2\alpha+3)}\,\frac{1}{\binom{n+2\alpha+2}{n}}, \tag{3.12}$$

recovering, for $\alpha = 0$, the first of (1.9).

4. The interior weights

We now put $f(t) = (1-t^2)P_n^{(\alpha+1,\beta+1)}(t)/(t-t_k)$ in (3.1) and take advantage of (3.2) to obtain

$$\bigl(1-t_k^2\bigr)\lambda_k P_n^{(\alpha+1,\beta+1)\prime}(t_k) = \int_{-1}^{1} \frac{P_n^{(\alpha+1,\beta+1)}(t)}{t-t_k}\,\mathrm{d}\lambda^{(\alpha+1,\beta+1)}(t). \tag{4.1}$$

We first compute the coefficient $(1-t_k^2)P_n^{(\alpha+1,\beta+1)\prime}(t_k)$ on the left. By (3.3) we have

$$P_n^{(\alpha+1,\beta+1)\prime}(t) = \frac{2}{n+\alpha+\beta+2}P_{n+1}^{(\alpha,\beta)\prime\prime}(t),$$

and using the differential equation satisfied by the Jacobi polynomial $P_{n+1}^{(\alpha,\beta)}$ [5, equation (4.2.1)], we get

$$\begin{aligned}\bigl(1-t^2\bigr)P_n^{(\alpha+1,\beta+1)\prime}(t) = \frac{2}{n+\alpha+\beta+2}\bigl\{&\bigl[\alpha-\beta+(\alpha+\beta+2)t\bigr]P_{n+1}^{(\alpha,\beta)\prime}(t)\\ &-(n+1)(n+\alpha+\beta+2)P_{n+1}^{(\alpha,\beta)}(t)\bigr\}.\end{aligned}$$

Therefore, by (3.4),

$$\bigl(1-t_k^2\bigr)P_n^{(\alpha+1,\beta+1)\prime}(t_k) = -2(n+1)P_{n+1}^{(\alpha,\beta)}(t_k). \tag{4.2}$$

To compute the integral on the right of (4.1), we proceed similarly as in section 3, using the Christoffel–Darboux formula ([5, equation (4.5.2)] with n replaced by $n-1$, and α, β replaced by $\alpha+1$, $\beta+1$), but this time combining it with (3.2), to obtain

$$\int_{-1}^{1}\frac{P_n^{(\alpha+1,\beta+1)}(t)}{t-t_k}\,\mathrm{d}\lambda^{(\alpha+1,\beta+1)}(t) = \frac{d_n(\alpha,\beta)}{P_{n-1}^{(\alpha+1,\beta+1)}(t_k)}, \tag{4.3}$$

where

$$d_n(\alpha,\beta) = 2^{\alpha+\beta+2}(2n+\alpha+\beta+2)\,\frac{\Gamma(n+\alpha+1)\Gamma(n+\beta+1)}{\Gamma(n+1)\Gamma(n+\alpha+\beta+3)}. \tag{4.4}$$

On the other hand, by (3.3) with n replaced by $n-1$, we have

$$P_{n-1}^{(\alpha+1,\beta+1)}(t_k) = \frac{2}{n+\alpha+\beta+1} P_n^{(\alpha,\beta)\prime}(t_k), \tag{4.5}$$

and by the second relation in [5, equation (4.5.7)],

$$\begin{aligned}\left(1-t_k^2\right)P_n^{(\alpha,\beta)\prime}(t_k) = \frac{n+\alpha+\beta+1}{2n+\alpha+\beta+2}\Big\{&\big[(2n+\alpha+\beta+2)t_k+\alpha-\beta\big]P_n^{(\alpha,\beta)}(t_k)\\ &-2(n+1)(n+\alpha+\beta+1)P_{n+1}^{(\alpha,\beta)}(t_k)\Big\}.\end{aligned} \tag{4.6}$$

Recalling (3.4), we have from the first relation in [5, equation (4.5.7) with n replaced by $n+1$],

$$\begin{aligned}&(2n+\alpha+\beta+2)\left(1-t_k^2\right)P_{n+1}^{(\alpha,\beta)\prime}(t_k)\\ &\quad= -(n+1)\big[(2n+\alpha+\beta+2)t_k+\beta-\alpha\big]P_{n+1}^{(\alpha,\beta)}(t_k)\\ &\qquad+2(n+\alpha+1)(n+\beta+1)P_n^{(\alpha,\beta)}(t_k) = 0,\end{aligned}$$

that is,

$$P_n^{(\alpha,\beta)}(t_k) = \frac{(n+1)[(2n+\alpha+\beta+2)t_k+\beta-\alpha]}{2(n+\alpha+1)(n+\beta+1)}P_{n+1}^{(\alpha,\beta)}(t_k).$$

Sustituting this into (4.6), and the result into (4.5), yields

$$\begin{aligned}&P_{n-1}^{(\alpha+1,\beta+1)}(t_k)\\ &\quad= \frac{2(n+1)}{2n+\alpha+\beta+2}\left[\frac{(2n+\alpha+\beta+2)^2t_k^2-(\alpha-\beta)^2}{2(n+\alpha+1)(n+\beta+1)}-2\right]\frac{P_{n+1}^{(\alpha,\beta)}(t_k)}{1-t_k^2}.\end{aligned}$$

Now (4.1) in combination with (4.2), (4.3), and (4.4) yields the desired formula,

$$\begin{aligned}\lambda_k^{(\alpha,\beta)} = {}&\frac{2^{\alpha+\beta+1}\Gamma(\alpha+2)\Gamma(\beta+2)}{\Gamma(\alpha+\beta+3)(n+1)^2}\frac{\binom{n+\alpha+1}{n}\binom{n+\beta+1}{n}}{\binom{n+\alpha+\beta+2}{n}}\\ &\times\frac{1}{\frac{4(n+\alpha+1)(n+\beta+1)+(\alpha-\beta)^2}{(2n+\alpha+\beta+2)^2}-t_k^2}\frac{1-t_k^2}{[P_{n+1}^{(\alpha,\beta)}(t_k)]^2}.\end{aligned} \tag{4.7}$$

In the ultraspherical case, this becomes

$$\lambda_k^{(\alpha,\alpha)} = \frac{2^{2\alpha+1}}{(n+1)^2}\frac{\Gamma^2(n+\alpha+2)}{\Gamma(n+1)\Gamma(n+2\alpha+3)}\frac{1}{[P_{n+1}^{(\alpha,\alpha)}(t_k)]^2}, \tag{4.8}$$

which, for $\alpha=0$, recovers

$$\lambda_k^{(0,0)} = \frac{2}{(n+1)(n+2)}\frac{1}{[P_{k+1}(t_k)]^2}, \tag{4.9}$$

the second relation in (1.9).

5. Numerical comparisons

In the numerical experiments described in this section, we used our software [2], in particular, the routines `recur` and `gauss` and their higher-precision companions. We are comparing the results of generating the Gauss–Lobatto formula by

(1) computing the modified Jacobi matrix (1.3) directly, using (2.6), and then computing the nodes as eigenvalues of (1.3) and the weights by means of (1.6);

(2) computing (1.3) and its eigenvalues as in (1), but computing the weights directly, using (3.10), (3.11), and (4.7).

We begin with the classical Gauss–Lobatto formula, i.e., with the case $\alpha = \beta = 0$. For each n being run, we determine the accuracy of the single-precision weights obtained by either (1) or (2) by comparing them with the double-precision results, and we record the respective relative errors $err1$, $err2$. We also record the number of times $err2$ is smaller, respectively larger, than $err1$, and by what amounts (i.e., calculating $r = err2/err1$). We denote by $err1b$, $err2b$ the larger of the relative errors of the two boundary weights obtained via (1) and (2), respectively. Similarly, $err1i$, $err2i$ will denote the maximum relative errors for the interior weights. The quantities $rmin$ and $rmax$ are the minimum respectively maximum ratio r taken over all interior weights.

In our first experiment, we take $n = 1(1)20$. An excerpt of the results is shown in table 1. (Integers in parentheses denote decimal exponents.) It is seen that the boundary weights obtained by (2) are considerably more accurate than those obtained by (1). The interior weights from (2) are also consistently more accurate than those from (1) in terms of their maximum errors. The same is true for all other values of n. Overall, the interior weights computed by (2) are more accurate (less accurate) than those computed by (1) in 165 [45] of the 210 cases, i.e., in 78.6% (21.4%) of all cases.

Our next experiment takes $n = 20(20)500$. Selected, but representative, results are shown in table 2. If anything, they reinforce the superiority of method (2), especially (though not surprisingly) for the boundary weights, but also, to a lesser degree, for the interior weights. As indicated by the column headed $rmin$, the gain in accuracy, even for the interior weights, is potentially more pronounced. It is now in 95.6% of all 6,500 cases, that the weights produced by (2) are more accurate than those produced by (1). Evidently, method (1) is pushing the limits of single-precision accuracy, and method (2), except for the boundary weights, is not far behind.

Table 1
Comparison of methods (1) and (2) for $n = 5(5)20$ in the case $\alpha = \beta = 0$.

n	$err1b$	$err2b$	$err1i$	$err2i$	$rmin$	$rmax$
5	0.26(–5)	0.19(–7)	0.76(–6)	0.36(–6)	0.18(0)	0.15(2)
10	0.26(–5)	0.32(–7)	0.30(–5)	0.68(–6)	0.20(0)	0.23(1)
15	0.24(–4)	0.60(–7)	0.99(–5)	0.98(–6)	0.34(–2)	0.43(1)
20	0.39(–4)	0.22(–6)	0.10(–4)	0.22(–5)	0.13(–1)	0.33(1)

Table 2
Comparison of methods (1) and (2) for $n = 100(100)500$ in the case $\alpha = \beta = 0$.

n	*err1b*	*err2b*	*err1i*	*err2i*	*rmin*	*rmax*
100	0.17(–2)	0.17(–6)	0.22(–3)	0.20(–4)	0.41(–2)	0.15(1)
200	0.26(–2)	0.86(–6)	0.95(–3)	0.78(–4)	0.75(–4)	0.25(2)
300	0.11(–1)	0.53(–6)	0.36(–2)	0.17(–3)	0.13(–2)	0.66(2)
400	0.23(–1)	0.57(–7)	0.34(–2)	0.29(–3)	0.12(–3)	0.17(2)
500	0.45(–1)	0.95(–6)	0.56(–2)	0.17(–2)	0.13(–3)	0.57(2)

Table 3
Comparison of methods (1) and (2) for $n = 1000(1000)5000$ in the case $\alpha = \beta = 0$.

n	*err1b*	*err2b*	*err1i*	*err2i*	*rmin*	*rmax*
1000	0.18(–9)	0.39(–14)	0.29(–10)	0.15(–11)	0.43(–4)	0.60(2)
2000	0.28(–8)	0.21(–14)	0.23(–9)	0.29(–11)	0.34(–4)	0.71(1)
3000	0.77(–8)	0.73(–14)	0.39(–9)	0.12(–10)	0.17(–5)	0.12(2)
4000	0.11(–9)	0.48(–14)	0.53(–9)	0.23(–10)	0.86(–5)	0.37(2)
5000	0.53(–8)	0.42(–14)	0.86(–9)	0.18(–10)	0.13(–5)	0.10(3)

To proceed to higher values of n, we adopt double and quadruple precision. Letting $n = 1000(1000)5000$ we find (at a considerable expense in computer time) the results in table 3. These, on the whole, are analogous to those in table 2, if not still more in favor of method (2); the latter indeed is more accurate than method (1) in 99% of all cases.

Other values of α and β were also tried. Specifically, we compared the two methods for α and β going through the values -0.9, -0.5, 0.5, 1.0, 2.0, 5.0, 10.0, independently from each other but with $\beta \leqslant \alpha$, and $n = 20(20)100$ for each α, β. The results are similar to those displayed for $\alpha = \beta = 0$ except for extremely poor accuracy in some of the interior weights computed by method (1) when α and β are both $\geqslant 5$ and $n \geqslant 80$. On the whole ensemble of 8,400 interior weights, those produced by method (2) are more accurate in 90% of all cases, by as much as a factor of 3.4×10^7, and never less accurate by more than a factor of 7.1×10^2.

The superiority of method (2) for generating the quadrature rule may or may not lead to more accurate quadrature results, the matter depending very much on the specific functions f involved.

Acknowledgement

This work was motivated by an e-mail message of December 17, 1998, from Sanjay Velamparambil inquring of ways to fix the underflow problem mentioned in section 1 in the case of the Legendre measure.

References

[1] P.J. Davis and P. Rabinowitz, *Methods of Numerical Integration*, 2nd ed. (Academic Press, Orlando, 1984).

[2] W. Gautschi, Algorithm 726: ORTHPOL – a package of routines for generating orthogonal polynomials and Gauss-type quadrature rules, ACM Trans. Math. Software 20 (1994) 21–62.

[3] W. Gautschi, Orthogonal polynomials: applications and computation, in: *Acta Numerica 1996*, ed. A. Iserles (Cambridge Univ. Press, Cambridge, 1996) pp. 45–119.

[4] G.H. Golub, Some modified matrix eigenvalue problems, SIAM Rev. 15 (1973) 318–334.

[5] G. Szegő, *Orthogonal Polynomials*, 4th ed., American Mathematical Society Colloquium Publications, Vol. 23 (Amer. Math. Soc., Providence, RI, 1978).