SIAM J. SCI. COMPUT.
Vol. 35, No. 2, pp. A652–A674


# FAST AND ACCURATE COMPUTATION OF GAUSS–LEGENDRE AND GAUSS–JACOBI QUADRATURE NODES AND WEIGHTS*

NICHOLAS HALE† AND ALEX TOWNSEND†

**Abstract.** An efficient algorithm for the accurate computation of Gauss–Legendre and Gauss–Jacobi quadrature nodes and weights is presented. The algorithm is based on Newton's root-finding method with initial guesses and function evaluations computed via asymptotic formulae. The $n$-point quadrature rule is computed in $\mathcal{O}(n)$ operations to an accuracy of essentially double precision for any $n \geq 100$.

**Key words.** quadrature, Gauss–Legendre, Gauss–Jacobi, asymptotic expansion

**AMS subject classifications.** 33C45, 41A60, 65D30, 65D32

**DOI.** 10.1137/120889873

## 1. Introduction.

Quadrature, or numerical integration, is the process of approximating the definite integral of a given function. Typically this approximation takes the form of a weighted sum of function evaluations, so that an $n$-point quadrature rule is given by

$$\int_a^b f(x)dx \approx \sum_{k=1}^{n} w_k f(x_k) \tag{1.1}$$

for some set of *nodes* $\{x_k\}$ and *weights* $\{w_k\}$. There are many different choices for the nodes and weights, and the *Gauss–Legendre* rule is defined by the unique choice such that (1.1) is exact when $f$ is any polynomial of degree $2n-1$ or less. More generally, a quadrature rule is referred to as "Gaussian" if for some given positive, integrable weight function $w(x)$, the approximation

$$\int_a^b w(x)f(x)dx \approx \sum_{k=1}^{n} w_k f(x_k)$$

is exact for all polynomials of degree $2n-1$ or less.

Gauss, in 1814, constructed the Gauss–Legendre quadrature rule using hypergeometric functions and continued fractions [17], and Jacobi, twelve years later, noted the quadrature nodes were precisely the roots of the Legendre polynomial of degree $n$ [31, 32].[1] Now almost all introductory numerical analysis texts show that the Gauss quadrature nodes are the simple roots of an associated orthogonal polynomial

In this paper we are concerned with *Gauss–Jacobi* quadrature, associated with the canonical interval $[-1, 1]$ and the Jacobi weight function

$$w(x) = (1+x)^\alpha (1-x)^\beta, \qquad \alpha, \beta > -1.$$

*Submitted to the journal's Methods and Algorithms for Scientific Computing section September 4, 2012; accepted for publication (in revised form) December 14, 2012; published electronically March 6, 2013. This work was supported by The MathWorks, Inc., and King Abdullah University of Science and Technology (KAUST), award KUK-C1-013-04.
http://www.siam.org/journals/sisc/35-2/88987.html
†Mathematical Institute, 24-29 St Giles', Oxford, OX1 3LB, England (hale@maths.ox.ac.uk, http://people.maths.ox.ac.uk/hale, townsend@maths.ox.ac.uk, http://people.maths.ox.ac.uk/townsend).
[1] An excellent account of the history of Gauss quadrature is given by Gautschi [19].

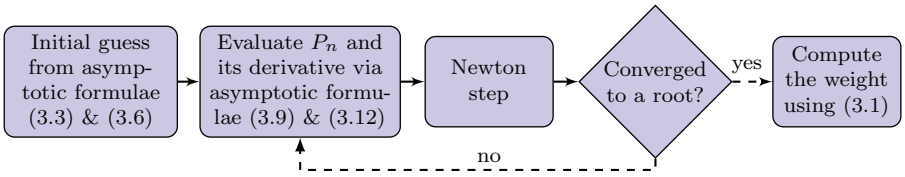


FIG. 1.1. *Algorithm for computing each Gauss–Legendre node and weight by a combination of Newton's method and asymptotic formulae. The formulae* (3.9) *and* (3.12) *refer to "interior" and "boundary" formulae, where the "boundary region" is simply the ten nodes nearest to* $\pm 1$. *An almost identical technique for more general Gauss–Jacobi quadratures is discussed in section* 3.3.

TABLE 1.1

*Errors (absolute, relative maximum, maximum relative, and quadrature—see section* 4 *for definitions) and computational time for computing* $n = 10^2, \ldots, 10^6$ *Gauss–Legendre nodes and weights using the algorithm described in this paper on a* 2011 1.8*-GHz Intel Core i*7 *MacBook Air with* MATLAB 2012*a*.

| $n$ | $\epsilon_{\text{abs}}\{x_k\}$ | $\epsilon_{\text{rm}}\{w_k\}$ | $\epsilon_{\text{mr}}\{w_k\}$ | $\epsilon_{\text{quad}}\{x_k, w_k\}$ | Time (secs) |
|---|---|---|---|---|---|
| 100 | 1.18e-16 | 1.15e-16 | 1.25e-15 | 1.71e-16 | 0.0085 |
| 1,000 | 1.63e-16 | 8.27e-16 | 1.92e-15 | 1.11e-16 | 0.0105 |
| 10,000 | 1.78e-16 | 1.14e-15 | 1.69e-15 | 1.11e-16 | 0.0261 |
| 100,000 | 2.22e-16 | 1.09e-15 | 1.48e-15 | 4.44e-16 | 0.2600 |
| 1,000,000 | 3.33e-16 | 2.70e-15 | 3.02e-15 | 6.66e-16 | 2.3198 |

In this case, the nodes $\{x_k\}$ are the roots of the Jacobi polynomial $P_n^{(\alpha,\beta)}$, and the weights $\{w_k\}$ are given by [45, p. 352]

$$w_k = \frac{\Gamma(n+\alpha+1)\Gamma(n+\beta+1)}{\Gamma(n+\alpha+\beta+1)n!} \frac{2^{\alpha+\beta+1}}{(1-x_k^2)\left[P_n^{(\alpha,\beta)'}(x_k)\right]^2}, \quad k = 1, \ldots, n. \tag{1.2}$$

Thus, computing Gauss–Jacobi nodes and weights reduces to finding the roots of a Jacobi polynomial and evaluating its derivative.

Due to the fast convergence of Gauss quadrature rules, particularly when $f$ is $C^\infty$ or analytic, most applications typically require only a relatively small number of nodes, say 5–200. Similarly, adaptive quadrature methods which perform recursive subdivision also do so with only a small number of points. However, there is some call for large global Legendre and Jacobi grids, for example, in spectral methods and high-degree polynomial integration [46, 53]. Furthermore, the relation between the quadrature and barycentric weights, as pointed out by Wang and Xiang [50, Theorem 3.1], allows the stable evaluation of Legendre and Jacobi interpolants.

Existing approaches for computing the nodes and weights, some of which have been widely used for many years, suffer from $\mathcal{O}(n^2)$ complexity or error which grows with $n$, which can be limiting when $n$ is large. In this paper we develop a new technique which utilizes asymptotic formulae for both accurate initial guesses of the roots and efficient evaluations of the degree $n$ Jacobi polynomial $P_n^{(\alpha,\beta)}$ inside Newton's method. With this new algorithm it is possible to compute the nodes and weights of Gauss–Jacobi quadrature rules in just $\mathcal{O}(n)$ operations to almost full double precision for any $n \geq 100$.

Furthermore, the algorithm can be easily parallelized or vectorized, making it very efficient in a variety of architectures and languages. A simple flowchart of the algorithm is shown in Figure 1.1 and a demonstration of the accuracy and computational time is given in Table 1.1.

This paper has the following structure: In section 2 we briefly review some existing methods and implementations for computing Gauss quadrature nodes and weights before introducing our new algorithm, first for the special case of Gauss–Legendre quadrature, and then generalized to Gauss–Jacobi, in section 3. Section 4 demonstrates the accuracy and efficiency of the computed quadrature rule as compared with a selection of the existing methods from section 2 and some extended precision computations. In section 5 we discuss further issues related to Gaussian quadrature and the new algorithm, such as extension to Radau, Lobatto, Hermite, and Laguerre quadratures, before concluding in section 6.

**2. Existing methods.** The traditional, and by far most widely used, method for computing Gauss quadrature nodes and weights is the Golub–Welsch (GW) algorithm [24], based upon the observation of Wilf [51, p. 55], which exploits the three-term recurrence relations satisfied by all real orthogonal polynomials. The relation gives rise to a symmetric tridiagonal matrix, the eigenvalues of which are the nodes of the quadrature rule, with the weights easily determined from the corresponding eigenvectors. The GW algorithm takes $\mathcal{O}(n^2)$ operations to solve this eigenvalue problem by taking advantage of the structure of the matrix and noting that only the first component of the normalized eigenvector need be computed. The complexity can be reduced to $\mathcal{O}(n \log n)$ if only the nodes are required [26]. However, MATLAB implementations of the GW/Wilf approach using `eig` are unable to take advantage of the special matrix structure, and the computational complexity is in fact $\mathcal{O}(n^3)$. Swarztrauber [44] has previously observed that the Golub–Welsch method leads to an $\mathcal{O}(n)$ error in the Gauss–Legendre nodes and an $\mathcal{O}(n^2)$ error in the weights, but our numerical experiments in Figure 2.1 suggest these may in fact be closer to $\mathcal{O}(\sqrt{n})$ for the nodes and $\mathcal{O}(n^{3/2})$ for the relative maximum error in the weights. One can further reduce error in the weights by using the recurrence to evaluate derivative values as a postprocessing step [34] or by modifying the weight formula [54]. Implementations can be found, for example, in [18, 25].

An alternative approach is to simply use the same three-term recurrence to compute Newton iterates which converge to the zeros of the orthogonal polynomial [39, 42]. Since the recurrence requires $\mathcal{O}(n)$ operations for each evaluation of the degree $n$ polynomial and its derivative, we again expect the total complexity for all the nodes and weights to be $\mathcal{O}(n^2)$. Here we observe a relative maximum error in the weights of $\mathcal{O}(n)$ and that the nodes can be computed to essentially machine precision independently of $n$. Initial guesses for the Newton iterations are discussed in section 3.1. Implementations can be found in [4, 14], and a further variant has been proposed [44]. For convenience, we refer to this as the REC algorithm. .

The current state of the art is the Glasier–Liu–Rohklin (GLR) algorithm [23], which computes all the nodes and weights of the $n$-point quadrature rule in a total of $\mathcal{O}(n)$ operations. This also employs Newton's method, but here the function and derivative evaluations are computed with $n$-independent complexity by sequentially hopping from one node to the next using a local 30-term Taylor series approximation generated from the second-order differential equation satisfied by the orthogonal polynomial. The GLR algorithm is very fast, but its sequential nature makes it difficult to parallelize or to vectorize for array-based languages like MATLAB. Here we again observe that the relative maximum error in the weights grows like $\mathcal{O}(n)$. Implementations can be found in [10, 46].

Figure 2.1 compares both the accuracy of each of these algorithms as $n$ is increased, and Table 2.1 summarizes these data. Note that in Figure 2.1 we show the

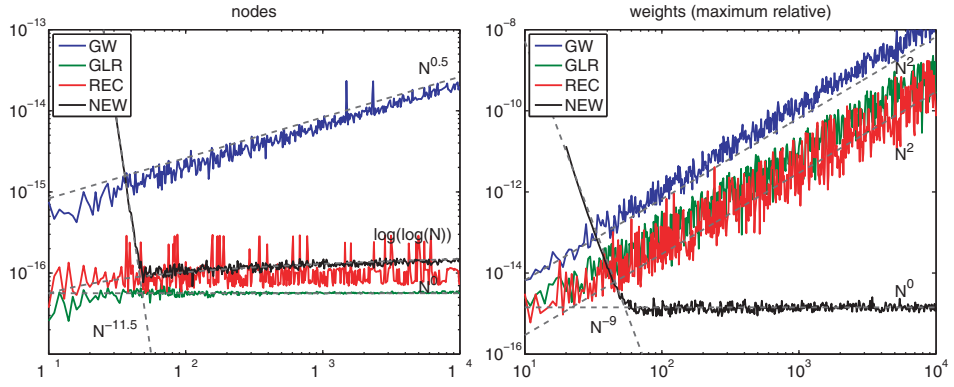


FIG. 2.1. *Observed error in computing Gauss–Legendre nodes and weights. GW uses the* `dgauss` *routine from the ORTHPOL library* [18], *REC is* `glfixed` *from the GSL library* [14], *GLR is a Fortran implementation supplied by the authors of* [23], *and NEW is a* MATLAB *implementation of the algorithm presented in this paper. The C and Fortran codes are interfaced with* MATLAB *via MEX files. See Figure* 4.3 *for analogous plots of quadrature error and computational time.*

TABLE 2.1

*Observed errors and computational complexity when computing Gauss–Legendre nodes and weights (see Figures* 2.1 *and* 4.3*) using the four algorithms described in this paper. Here, and throughout,* $\mathcal{O}(1)$ *denotes a number independent of* $n$. *See section* 4 *for definitions of the error measures.*

| Algorithm | $\epsilon_{\text{abs}}\{x_k\}$ | $\epsilon_{\text{rm}}\{w_k\}$ | $\epsilon_{\text{mr}}\{w_k\}$ | $\epsilon_{\text{quad}}\{x_k, w_k\}$ | Time |
|---|---|---|---|---|---|
| GW | $\mathcal{O}(\sqrt{n})$ | $\mathcal{O}(n^{3/2})$ | $\mathcal{O}(n^2)$ | $\mathcal{O}(n)$ | $\mathcal{O}(n^2)$ |
| REC | $\mathcal{O}(1)$ | $\mathcal{O}(n)$ | $\mathcal{O}(n^2)$ | $\mathcal{O}(\sqrt{n})$ | $\mathcal{O}(n^{1.7})$ |
| GLR | $\mathcal{O}(1)$ | $\mathcal{O}(n)$ | $\mathcal{O}(n^2)$ | $\mathcal{O}(n^{0.66})$ | $\mathcal{O}(n)$ |
| NEW | $\mathcal{O}(\log(\log(n)))$ | $\mathcal{O}(\log(\log(n)))$ | $\mathcal{O}(1)$ | $\mathcal{O}(1)$ | $\mathcal{O}(n)$ |

maximum relative error, rather than the relative maximum. Full definitions of the error measures and further discussion of these results can be found in section 4. We note that while the three algorithms mentioned above can be used in a more general setting, the algorithm presented in this paper requires only $\mathcal{O}(n)$ operations to compute an $n$-point Gauss–Jacobi quadrature rule with absolute and maximum relative errors in both the nodes *and* weights that are essentially independent of $n$.

**3. New method.** Similarly to both the REC and GLR approaches, our new method is based upon Newton's method for finding roots of a univariate function. However, the difference here is in how we evaluate the Jacobi polynomial and its derivative, where we take the premise that $n$ is large and employ asymptotic expansions. Bogaert, Michiels, and Fostier [7] have recently explored this idea in the context of fast evaluation of Legendre series (see the "note added in proof" following section 6 for further details).

We now briefly discuss some details of Newton's method relevant to this application before introducing the asymptotic expansions of the Legendre and Jacobi polynomials of large degree.

**3.1. Newton's method.** It is well known that the Legendre and Jacobi nodes cluster quadratically near $\pm 1$, and that this can have adverse affects on their numerical computation. For extremely large $n$ the clustering eventually leads to coalescence on the discrete floating point scale, so that if, for fun, the billionth Gauss–Legendre

quadrature rule is constructed, then many of the nodes near $\pm 1$ are indistinguishable. To be more precise, points begin to coalesce once $n \gtrsim \sqrt{1/\varepsilon_{mach}}$, where $\varepsilon_{mach}$ is the precision used to store the nodes, meaning one would rarely expect to use more than this many points in practice. Relatedly, cancellation in the $\left(1 - x_k^2\right)$ term in the denominator of (1.2) near $\pm 1$ can make accurate computation of the weights difficult.

Swarztrauber has advocated modifying the Golub–Welsch method to compute the approximately equally spaced points in the transplanted $\theta$-space, $\theta_k = \cos^{-1} x_k$, to avoid this clustering [44]. For this reason, and also because the asymptotic expansions of Legendre polynomials are most naturally defined in terms of $P_n(\cos\theta)$, we also choose to work in $\theta$-space. We immediately note that a possible downside of working in this $\theta$-space is that it is only possible to obtain an *absolute* accuracy of machine epsilon near $x = 0$, where $\theta \approx \pi/2$. In $x$-space one can obtain a better *relative* accuracy near the origin, but this is rarely required, and often comes at the expense of accuracy of the weights near $\pm 1$.

Thus, $\theta_k^{[0]}$, an initial guess to the $k$th root is chosen and successive iterates constructed via

$$\theta_k^{[j+1]} = \theta_k^{[j]} - P_n(\cos\theta_k^{[j]}) \left( -\sin\theta_k^{[j]} P_n'(\cos\theta_k^{[j]}) \right)^{-1}, \qquad j = 0, 1, 2, \ldots.$$

Once the iterates have converged, the nodes are given by $x_k = \cos\theta_k$ and the weights by

$$w_k = \frac{C_{n,\alpha,\beta}}{(1 - x_k^2)\left[P_n'(x_k)\right]^2} = \frac{C_{n,\alpha,\beta}}{\left[\frac{d}{d\theta} P_n(\cos\theta_k)\right]^2}, \tag{3.1}$$

where

$$C_{n,\alpha,\beta} = 2^{\alpha+\beta+1} \frac{\Gamma(n+\alpha+1)\Gamma(n+\beta+1)}{\Gamma(n+\alpha+\beta+1)n!}. \tag{3.2}$$

Note that the expression in $\theta$-space avoids the $(1 - x^2)$ term in the denominator that is susceptible to round-off error near $\pm 1$. While there are a number of mathematically equivalent formulae for the weights [44], we find (3.1) advantageous both because the required derivative must be computed to calculate a Newton update, and because it is less susceptible to errors in the node locations [54].

Since the zeros of the orthogonal Jacobi polynomial are simple, the Newton iterates converge quadratically when started at a sufficiently good initial guess. Petras [39] has shown that, for the Gauss–Legendre nodes, very simple initial guesses, such as the Chebyshev points, are good enough for convergence. However, better initial guesses will require fewer Newton iterations, and these can be obtained from asymptotic approximations for the roots. In the next section we shall see that there are two regimes for the asymptotic expansions: an interior expansion and a boundary expansion (see Figure 3.2), and the same is true, and indeed strongly related [16], for the initial guesses. Furthermore, since all Jacobi polynomials satisfy the reflection relation

$$P_n^{(\alpha,\beta)}(-x) = (-1)^n P_n^{(\beta,\alpha)}(x),$$

we need only consider $x \in [0, 1)$, i.e., $\theta \in (0, \pi/2]$. To this end we introduce the notation $\bar{k} = n - k + 1$, so that $x_{\bar{k}}$ is the $k$th closest node to $x = 1$. Then, away from $x = 1$, there are formulae for the Gauss–Legendre nodes given by Tricomi [49],

$$x_{\bar{k}} = \left\{ 1 - \frac{(n-1)}{8n^3} - \frac{1}{384n^4} \left( 39 - \frac{28}{\sin^2\phi_k} \right) \right\} \cos\phi_k + \mathcal{O}(n^{-5}), \tag{3.3}$$

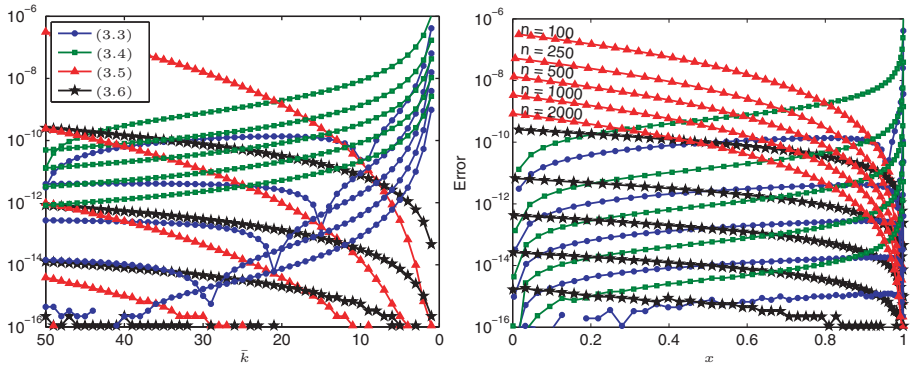


FIG. 3.1. *Accuracy of asymptotic Legendre root approximations* (3.3)–(3.6) *for the* 50 *roots nearest to* $x = 1$ *(left) and the interval* $[0, 1]$ *(right), with* $n = 100$, 250, 500, 1000, 2000 *(descending). An optimal choice would involve a selection of three of the formulae, but since an accuracy of below the root of machine precision is sufficient for our purposes we simply choose* (3.3) *when* $x \le 1/2$ *and* (3.6) *when* $x > 1/2$.

where $\phi_k = (k - 1/4)\,\pi/(n + 1/2)$, and by Gatteschi and Pittaluga [16],

$$x_{\bar{k}} = \cos\left\{\phi_k + \frac{1}{4(n+1/2)^2}\left(\frac{1}{4}\cot\frac{\phi_k}{2} - \frac{1}{4}\tan\frac{\phi_k}{2}\right)\right\} + \mathcal{O}(n^{-4}). \tag{3.4}$$

Although (3.3) is a higher-order approximation, (3.4) generalizes (with some restrictions on $\alpha$ and $\beta$) to the Gauss–Jacobi case (see (3.19)). For nodes near $x = 1$, there are formulae given by Gatteschi [15],

$$x_{\bar{k}} = \cos\left\{\frac{j_k}{\nu}\left(1 - \frac{4}{720\nu^4}\left(j_k^2/2 - 1\right)\right)\right\} + j_k^5\mathcal{O}(n^{-7}), \tag{3.5}$$

where $\nu = \sqrt{(n + 1/2)^2 + 1/12}$, and by Olver [38, Ex. 12.5],

$$x_{\bar{k}} = \cos\left\{\psi_k + \frac{\psi_k\cot(\psi_k) - 1}{8\psi_k(n+1/2)^2}\right\} + j_k^2\mathcal{O}(n^{-5}), \quad \psi_k = \frac{j_k}{n+1/2}, \tag{3.6}$$

where $j_k$ is the $k$th root of $J_0(x)$, the Bessel function of the first kind. The roots of $J_0(x)$ are independent of $n$ so they can be precomputed and tabulated [22] or computed on the fly [8]. Both (3.5) and (3.6) can be generalized to roots of more general Jacobi polynomials. These results, and many others, can be found in a survey of Gatteschi's work [21].

Lether [33] investigates which of the above approximations to use for each $k$, and an empirical rule suggested by Yakimiw [54] is to use (3.5) when $\bar{k} \le \lceil 0.063(n+33)(n-1.5)/n\rceil$ and (3.3) otherwise. However, Figure 3.1 (left) shows the accuracy of the 50 nodes nearest $x = 1$ for a range of $n$ much larger than that considered by Lether, and suggests that, except for a very small number of the points at the boundary, (3.6) gives better accuracy than (3.5). In particular, using the rule of thumb that $j_k \sim k\pi$, balancing the error terms in (3.5) and (3.6) confirms this crossover occurs when $\bar{k} \approx n^{2/3}/\pi$.

Rather than concoct a complicated optimal choice for a given $k$ and $n$, we simply observe that Figure 3.1 (right) suggests that (3.3) and (3.6) cross when $x \approx 1/2$, and

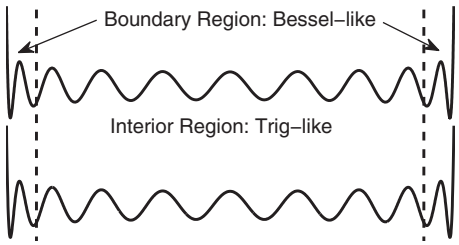


FIG. 3.2. *Comparing the degree-20 Legendre polynomial (upper) with the first term in the asymptotic formulae* (3.9) *and* (3.12) *(lower). In the interior of the domain, Legendre polynomials can be well approximated by trigonometric functions, whereas near the boundary region they more closely resemble Bessel functions. Even for* $n = 20$ *the asymptotic expansions are good approximations.*

choose $\theta_k^{[0]} = \arccos(x_k^{[0]})$, where

$$x_k^{[0]} = \begin{cases} (3.3) \text{ for } x \le 1/2 \quad (\text{i.e., } \theta \ge \pi/3)\,, \\ (3.6) \text{ for } x > 1/2 \quad (\text{i.e., } \theta < \pi/3)\,. \end{cases}$$

**3.2. Gauss–Legendre.** Gauss–Legendre is the most commonly used of the Gaussian quadrature rules. It corresponds to the constant weight function $w(x) = 1$, and hence gives rise to the approximation (1.1). The nodes are the zeros of the Legendre polynomial $P_n(x)$, and the formula (1.2) for the weights simplifies to

$$w_k = \frac{2}{(1-x_k^2)\left[P_n'(x_k)\right]^2} = \frac{2}{\left[\frac{d}{d\theta}P_n(\cos\theta_k)\right]^2}. \tag{3.7}$$

By symmetry the nodes and weights are reflected about $x = 0$ and one needs only compute those in $[0, 1)$ or, equivalently, $\theta \in (0, \pi/2]$. The derivative of the Legendre polynomial, $P_n'$, satisfies the recurrence relation

$$(1-x^2)P_n'(x) = -nxP_n(x) + nP_{n-1}(x)$$

or, equivalently in the $\theta$-variable,

$$-\sin\theta\, \frac{d}{d\theta}P_n(\cos\theta) = -n\cos\theta P_n(\cos\theta) + nP_{n-1}(\cos\theta). \tag{3.8}$$

To evaluate $P_n$ (and $P_{n-1}$ for $\frac{d}{d\theta}P_n$), we use two different asymptotic formulae, which we refer to as the *interior* and *boundary* expansions. The interior expansion (3.9) involves only elementary functions and has readily computable coefficients, but is not valid for $x$ near $\pm 1$. Conversely, the "boundary" expansion (3.12), which is in fact valid for all $\theta \in [0, \frac{\pi}{2}]$, involves Bessel functions and only the first few coefficients are known in closed form. Figure 3.2 shows the first term in each of these expansions.

**3.2.1. Interior asymptotic formula.** The interior asymptotic formula we use was derived by Stieltjes in 1890 from a contour integral representation for the Legendre polynomial [43]. It is given by

$$P_n(\cos\theta) = C_n \sum_{m=0}^{M-1} h_{n,m} \frac{\cos(\alpha_{n,m})}{(2\sin\theta)^{m+\frac{1}{2}}} + R_{n,M}(\theta)\,, \tag{3.9}$$

where $\alpha_{n,m} = \left(n + m + \frac{1}{2}\right)\theta - \left(m + \frac{1}{2}\right)\frac{\pi}{2}$,

$$C_n = \frac{4}{\pi}\prod_{j=1}^{n}\frac{j}{j+1/2} = \sqrt{\frac{4}{\pi}}\frac{\Gamma(n+1)}{\Gamma\left(n+3/2\right)}, \tag{3.10}$$

and

$$h_{n,m} = \prod_{j=1}^{m}\frac{(j-1/2)^2}{j(n+j+1/2)}, \quad m > 0,$$

with $h_{n,0} = 1$. Szegő [45] bounds the error term in (3.9) by

$$|R_{n,M}(\theta)| < C_n h_{n,M}\frac{\max\{|\cos\theta|^{-1}, 2\sin\theta\}}{(2\sin\theta)^{M+\frac{1}{2}}}. \tag{3.11}$$

Another asymptotic formula[2] was derived by Darboux in 1878 [11], but Olver [37] shows that although this expansion approximates $P_n(\cos\theta)$ well for $M \ll n$, it actually converges to $2P_n(\cos\theta)$ as $M \to \infty$.

Stieltjes' expansion (3.9) converges to $P_n(\cos\theta)$ in $\theta \in [\pi/6, 5\pi/6]$ and diverges otherwise [43]. In practice, a finite number of terms are taken and Szegő [45] argues that the interior expansion can be used for almost the whole interval $[0, \pi]$. We discuss this issue further in section 3.2.4.

The error bound (3.11) is so similar to the terms in the expansion that in loop-based languages it can be checked for each $\theta$ at minimal cost. In array-based languages we use a fixed $M$ for all $\theta$, even though the error bound reveals that fewer terms can be taken when $\theta \approx \pi/2$. Since the terms are computationally cheap, we suggest computing the first 20 and then truncating using (3.11) evaluated at the smallest $\theta$ in the interior region.

**3.2.2. Boundary asymptotic formula.** Unfortunately, there is no asymptotic expansion of the Legendre polynomials involving only elementary functions that is valid near $x = \pm 1$ [35]. The boundary asymptotic formula we use was obtained by Baratella and Gatteschi [5], based on the method described by Olver [36], and is an expansion in Bessel functions of the first kind. It is derived by considering the second-order differential equation satisfied by Legendre polynomials, and is given by

$$P_n(\cos\theta) = \sqrt{\frac{\theta}{\sin\theta}}\left(J_0(\rho\theta)\sum_{m=0}^{M}\frac{A_m(\theta)}{\rho^{2m}} + \theta J_1(\rho\theta)\sum_{m=0}^{M-1}\frac{B_m(\theta)}{\rho^{2m+1}}\right) + S_{n,M}(\theta), \tag{3.12}$$

where $\rho = n + \frac{1}{2}$ and

$$S_{n,M}(\theta) = \begin{cases} \theta^{\frac{5}{2}}\mathcal{O}\left(n^{-2M}\right), & 0 < \theta \le \frac{c}{n}, \\ \theta\mathcal{O}\left(n^{-2M-\frac{3}{2}}\right), & \frac{c}{n} \le \theta \le \frac{\pi}{2}. \end{cases}$$

Only the first few terms are known explicitly:

$$A_0 = 1, \quad B_0 = \frac{1}{4}g(\theta), \quad \text{and} \quad A_1(\theta) = \frac{1}{8}g'(\theta) - \frac{1}{8}\frac{g(\theta)}{\theta} - \frac{1}{32}g^2(\theta), \tag{3.13}$$

[2]Both Darboux's and Stieltjes' formulae can be more readily located in Theorems 8.21.4 and 8.21.5 of [45], respectively.

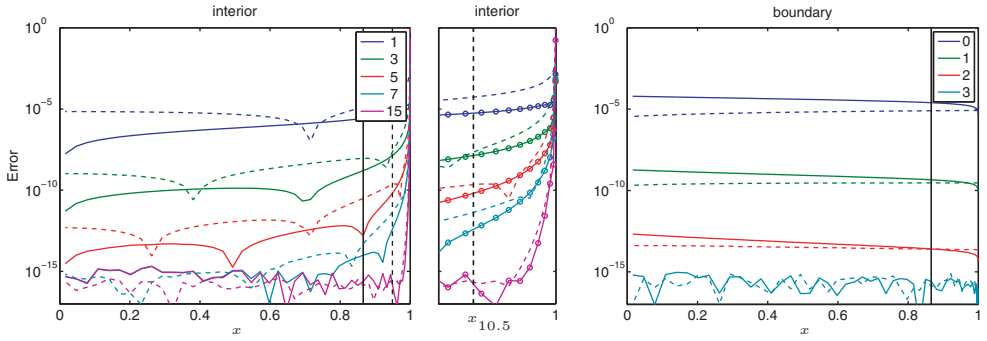


FIG. 3.3. *Error in evaluation of* $P_{100}(\cos\theta_k)$ *(solid line) and* $\frac{d}{d\theta}P_{100}(\cos\theta_k)$ *(dashed line: relative) using the interior asymptotic formula* (3.9) *with* $M = 1, 3, 5, 7, 15$ *(left and center) and the boundary asymptotic formula* (3.12) *with* $M = 0, 1, 2, 3$ *(right). Here the* $\theta_k$ *are the approximations to the Legendre roots from* (3.6)*, and the reported error is as compared to an extended precision computation. The vertical solid and dashed lines are at* $x = \cos(\pi/6)$ *and* $(x_{\overline{11}} + x_{\overline{10}})/2$*, respectively.*

where $g(\theta) = (\theta\cot\theta - 1)/2\theta$, but extra terms can be calculated numerically from the relations given in [5]. In practice we use Chebyshev interpolants to compute the indefinite integrals required for terms up to $M = 3$. This is enough to evaluate (3.12) to around machine precision near the boundaries for $n \geq 100$ and is far from the divergent regime of the asymptotic formula (see Figure 3.3). In fact, for $n \gtrsim 250$, only the first three terms are required ($M = 2$), and for $n \gtrsim 4000$ only the first two (i.e., those which are known explicitly).

In [7] Bogaert, Michiels, and Fostier derive an alternative asymptotic expansion that contains terms of the form $y^n J_n(x)$.

### 3.2.3. Computational issues.

**Computing derivatives.** To compute the Newton updates and quadrature weights we must evaluate $\frac{d}{d\theta}P_n(\cos\theta)$ which, by (3.8), can be computed from $P_n$ and $P_{n-1}$. In the interior asymptotic formula (3.9) one finds the constants are conveniently related by

$$C_{n-1} = \frac{n+1/2}{n}C_n \quad \text{and} \quad h_{n-1,m} = \frac{n+m-1/2}{n+1/2}h_{n,m},$$

so that the derivative can be cheaply evaluated at the same time as $P_n(\cos\theta)$ using

$$\frac{d}{d\theta}P_n(\cos\theta) \approx C_n \sum_{m=0}^{M-1} h_{n,m}\frac{(m-1/2)\cot\theta\cos\alpha_{n,m} + (n+m-1/2)\sin\alpha_{n,m}}{(2\sin\theta)^{m+\frac{1}{2}}}.$$

While Newton's method will not suffer from small inaccuracies in evaluating derivatives [47], it is clear from (3.7) that the relative error in evaluating the quadrature weights is proportional to the relative error in evaluating $\frac{d}{d\theta}P_n(\cos\theta_k)$. Thus to obtain an $\mathcal{O}(1)$ relative error in the weights, we require an $\mathcal{O}(1)$ relative error in the derivative evaluations. Since $h_{n,m} = \mathcal{O}(n^{-m})$ will hide the errors in further terms, we need only look at the $m = 0$ term, where we observe $\sin\alpha_{n,0} = \sin((n+\frac{1}{2})\theta_{\tilde{k}} + \pi/4) = \sin(k\pi + \mathcal{O}(n^{-1}))$, which can easily be evaluated with $\mathcal{O}(1)$ accuracy using Taylor series expansion about $k\pi$. Double-angle formulae can be used to avoid evaluating sines and cosines of large arguments in the other $\sin\alpha_{n,m}$ and $\cos\alpha_{n,m}$ terms.

The derivative of the boundary formula can also be computed using (3.8), requiring the additional evaluation of the Bessel functions $J_0$ and $J_1$ at $(n-\frac{1}{2})\theta$. The first term when evaluating the derivative at the nodes is then given by

$$\frac{d}{d\theta}P_n(\cos\theta_{\bar{k}}) = -n\sqrt{\frac{\theta_{\bar{k}}}{\sin\theta_{\bar{k}}}}\left[\frac{\cos\theta_{\bar{k}}J_0(\rho\theta_{\bar{k}}) - J_0((\rho-1)\theta_{\bar{k}})}{\sin\theta_{\bar{k}}}\right] + \cdots, \tag{3.14}$$

and by the same argument as above for the interior formula, we require an $\mathcal{O}(1)$ error in the term in the square parenthesis to obtain an $\mathcal{O}(1)$ relative error in the weights. Since, for a fixed $k$, $\theta_{\bar{k}}$ and $\sin\theta_{\bar{k}}$ are $\mathcal{O}(n^{-1})$, we must then demand an $\mathcal{O}(n^{-1})$ error in the numerator of this term. Standard methods of evaluating $J_0(\rho\theta_{\bar{k}})$ and $J_0((\rho-1)\theta_{\bar{k}})$ in double precision typically give only an absolute accuracy of 14–15 digits, and so are not sufficient.

Instead we observe that if $J_0((\rho-1)\theta_{\bar{k}})$ is computed by evaluating the Taylor series around $\rho\theta_{\bar{k}}$, (3.14) becomes

$$\frac{d}{d\theta}P_n(\cos\theta_{\bar{k}}) = -n\sqrt{\frac{\theta_{\bar{k}}}{\sin\theta_{\bar{k}}}}\left[\frac{\cos\theta_{\bar{k}}-1}{\sin\theta_{\bar{k}}}J_0(\rho\theta_k) + \frac{\theta_{\bar{k}}}{\sin\theta_{\bar{k}}}\sum_{l=1}^{\infty}\frac{J_0^{(l)}(\rho\theta_{\bar{k}})}{l!}(-\theta)^{l-1}\right] + \cdots,$$

where the required derivatives are given in closed form by the relation [35, eq. (18.6.7)]

$$J_\nu^{(l)}(z) = 2^{-l}\sum_{j=0}^{l}(-1)^j\binom{l}{j}J_{\nu-l+2j}(z).$$

Now, since $(\cos\theta_{\bar{k}}-1)/\sin\theta_{\bar{k}} \sim -\theta_{\bar{k}}/2$ and, by (3.6), $\theta_{\bar{k}} \sim j_k/\rho$, the $\mathcal{O}(1)$ errors in computing $J_0(\rho\theta_k)$ in $P_n$ and $P_{n-1}$ cancel (to $\mathcal{O}(n^{-1})$), and one obtains a relative error of $\mathcal{O}(1)$ for the derivative evaluation. This cancellation highlights the importance of choosing the right form of the weight formula (3.1).

**The ratio of gamma functions.** The constant $C_n$ in (3.10) for the interior asymptotic formula requires some careful computation. The first expression in (3.10) is a product of $n$ numbers which evaluates to $\mathcal{O}(\sqrt{n})$, and an error that grows like $\mathcal{O}(\sqrt{n})$ might be expected if this is computed directly. The second expression is the ratio of two gamma functions, $\Gamma(n+1)/\Gamma(n+3/2)$, which can easily overflow if $n$ is large.

To compute this constant accurately, and in particular with an error independent of $n$, we again take advantage of the fact that $n$ is large and use Stirling's series [12] for the approximation of gamma functions:

$$\Gamma(z+1) \sim z^{z+\frac{1}{2}}e^{-z}\sqrt{2\pi}S(z), \qquad |\arg(z)| < \pi,$$

where the first four terms of $S(z)$ are

$$S(z) = 1 + \frac{1}{12z} + \frac{1}{288z^2} - \frac{139}{51840z^3} - \frac{571}{2488320z^4} + \mathcal{O}(z^{-5}). \tag{3.15}$$

In slightly more generality, we substitute Stirling's series in the ratio of $\Gamma(n+1)$ and $\Gamma(n+1+\alpha)$ and simplify to obtain

$$\frac{\Gamma(n+1)}{\Gamma(n+1+\alpha)} \sim \sqrt{\frac{n}{n+\alpha}}\left(\frac{n}{n+\alpha}\right)^{n+\alpha}\left(\frac{e}{n}\right)^{\alpha}\frac{S(n)}{S(n+\alpha)}. \tag{3.16}$$

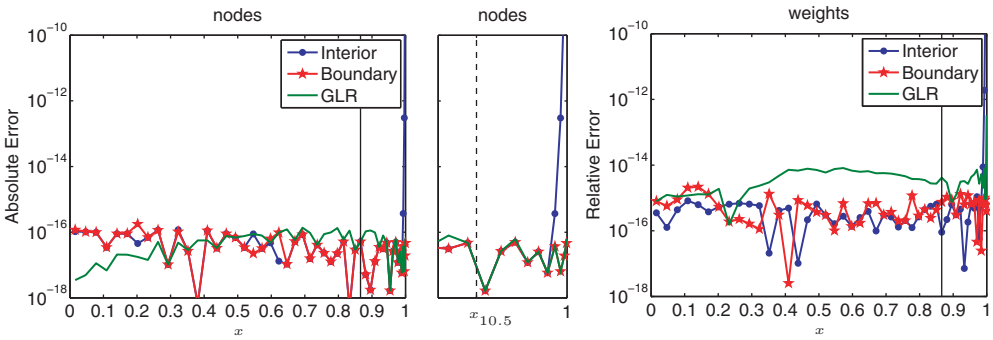


FIG. 3.4. *Error in Gauss–Legendre nodes (top) and weights (bottom) for* $n = 100$ *using interior and boundary asymptotic formulae. The GLR results are included for reference. As expected, the interior formula diverges near the boundary, but only in the designated boundary region (dashed vertical line). Although GLR achieves better relative accuracy in the nodes near* $x = 0$*, the asymptotic formulae give more accurate weights, particularly near* $x = 1$ *(see Figure* 3.5*).*

The term with the exponent $n + \alpha$ can then be evaluated using the series expansion

$$(n+\alpha)^{n+\alpha} = n^{n+\alpha} \exp\left( (n+\alpha) \sum_{j=1}^{\infty} \frac{(-1)^j}{j} \left(\frac{\alpha}{n}\right)^j \right), \tag{3.17}$$

which can be derived by considering the standard expansion of $\log(1 + \alpha/n)$. This approach turns out to alleviate overflow concerns and to be very accurate, and we note that although Stirling's series is not convergent, 10 terms in the function $S(z)$ are more than enough to achieve double precision in the evaluation of (3.10) for all $n \geq 100$. Many authors have proposed alternative ways of computing (3.10) [2, 7, 38], but (3.16) has the advantage of being readily applicable in the Gauss–Jacobi formulae that we will see shortly.

**3.2.4. Preliminary results.** Since the interior formula contains only elementary functions and has readily computable coefficients, we find it much easier to work with than the boundary formula, and aim to use it for as many of the nodes as possible. In particular, we find the computation cost per node in the boundary formula to be approximately 1000 times that of the interior formula. Unfortunately, the interior formula (3.9) diverges as $M \to \infty$ for $x > \cos(\pi/6)$ [43], and so formally this should define the interior region. However, in practice, and in agreement with Szegő [45], we find that for the finite number of terms required to achieve an accuracy of machine precision, (3.9) can be used to evaluate much more closely to the boundary than this suggests (see Figure 3.3). Therefore, based upon heuristic observations such as those in Figures 3.3–3.5, we define the regions as

$$\begin{aligned} &\text{interior region:} && \{x_{11}, \ldots, x_{n-10}\}, \\ &\text{boundary region:} && \{x_1, \ldots, x_{10}\} \cup \{x_{n-9}, \ldots, x_n\}. \end{aligned} \tag{3.18}$$

That is, the boundary region consists of the ten nodes nearest the boundary.

Figure 3.4 shows the error in computing the Gauss–Legendre nodes and weights for $n = 100$ using both in the interior and boundary formulae on the interval $x \in [0, 1]$, with the GLR method also included for reference. As expected, the interior formula diverges near the boundary, but importantly to the right of the vertical dashed line

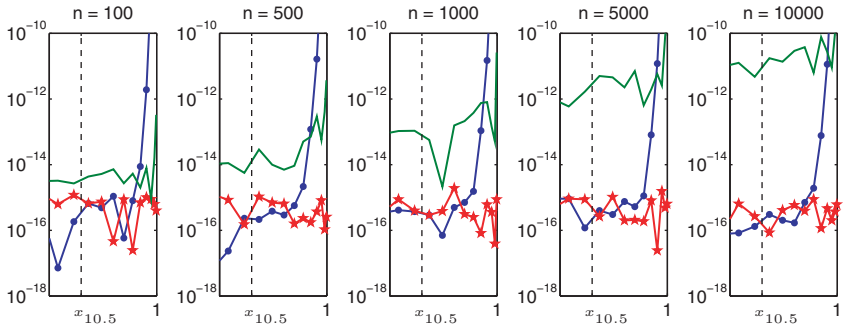


FIG. 3.5. *Error in the* 13 *rightmost Gauss–Legendre weights for* $n = 100, 500, 1000, 5000, 10000$. *We see that even as* $n$ *is increased, the interior formula gives good results for all but the final five or six nodes. In practice, we define the "boundary region" as the* 10 *nodes nearest the boundary.*

depicting the proposed border between the interior and boundary regions. Although GLR achieves better relative accuracy in the nodes near $x = 0$, the asymptotic formulae are within a factor of two or three of machine precision and give more accurate weights near the boundary. Figure 3.5 shows a close-up of the nodes near $x = 1$ for larger values of $n$ and demonstrates both that (3.18) is a reasonable choice for defining the boundary region, and that the relative error in the weights computed by the boundary formula does not increase with $n$. More detailed results can be found in section 4.

**3.3. Gauss–Jacobi.** The asymptotic expansions in the previous section extend readily to more general Jacobi polynomials. For $\alpha = \beta \neq 0$ the polynomials are known as *ultraspherical* or *Gegenbauer*, and simplifications similar to the Legendre case, such as symmetry of the nodes, can be used for improved efficiency. For $\alpha \neq \beta$ we exploit the reflection formulae

$$P_n^{(\alpha,\beta)}(-x) = (-1)^n P_n^{(\beta,\alpha)}(x), \qquad \tfrac{d}{dx} P_n^{(\alpha,\beta)}(-x) = (-1)^{n-1} \tfrac{d}{dx} P_n^{(\beta,\alpha)}(x),$$

so that we need only evaluate Jacobi polynomials on the right half of the interval, i.e., $x \in [0, 1)$, $\theta \in (0, \pi/2]$.

Most of the approximations for the roots of the Legendre polynomial can be generalized to the Jacobi case for use as initial guesses in Newton's method. Gatteschi and Pittaluga's approximation for the roots of $P_n^{(\alpha,\beta)}$ away from $\pm 1$ is given by

$$x_{\bar{k}} = \cos\left\{\phi_k + \frac{1}{4\rho^2}\left(\left(\frac{1}{4} - \alpha^2\right)\cot\frac{\phi_k}{2} - \left(\frac{1}{4} - \beta^2\right)\tan\frac{\phi_k}{2}\right)\right\} + \mathcal{O}(n^{-4}), \tag{3.19}$$

for $\alpha, \beta \in \left[-\frac{1}{2}, \frac{1}{2}\right]$, where $\rho = n + (\alpha + \beta + 1)/2$ and $\phi_k = (k + \alpha/2 - 1/4)\,\pi/\rho$. Similarly, with $j_{\alpha,k}$ denoting the $k$th root of $J_\alpha(z)$, the approximation given by Gatteschi for the nodes near $x = 1$ becomes

$$x_{\bar{k}} = \cos\left\{\frac{j_{\alpha,k}}{\nu}\left(1 - \frac{4 - \alpha^2 - 15\beta^2}{720\nu^4}\left(j_{\alpha,k}^2/2 + \alpha^2 - 1\right)\right)\right\} + j_{\alpha,k}^5\,\mathcal{O}(n^{-7}), \tag{3.20}$$

for $\alpha, \beta \in \left[-\frac{1}{2}, \frac{1}{2}\right]$, where $\nu = \sqrt{\rho^2 + (1 - \alpha^2 - 3\beta^2)/12}$, and Olver's approximation becomes

$$x_{\bar{k}} = \cos\left\{\psi_k + (\alpha^2 - 1/4)\frac{\psi_k \cot(\psi_k) - 1}{2\rho^2\psi_k} - (\alpha^2 - \beta^2)\frac{\tan(\phi_k/2)}{4\rho^2}\right\} + j_{\alpha,k}^2\,\mathcal{O}(n^{-5}) \tag{3.21}$$

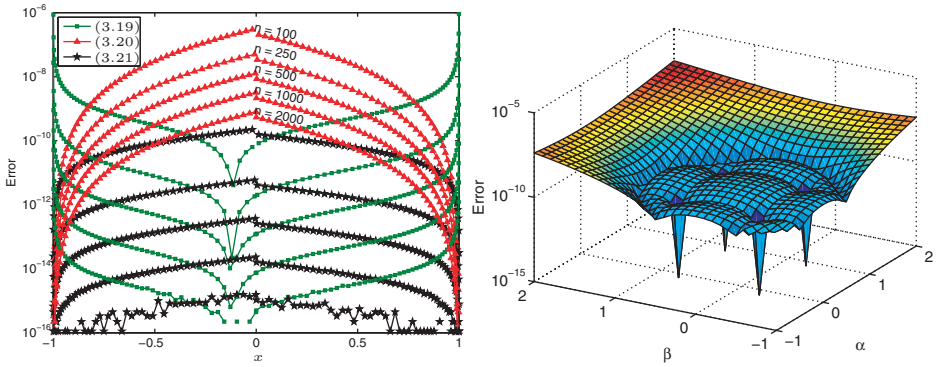


FIG. 3.6. *Left: Error in approximations* (3.19)–(3.21) *for the roots of* $P_n^{(0.1,-0.3)}(x)$*, with* $n = 100$*,* 250*,* 500*,* 1000*,* 2000 *(descending). Here the crossover between the interior approximation* (3.19) *and Olver's "boundary" approximation* (3.21) *doesn't occur until* $x \approx 0.1$*. Right: Error in approximation* (3.21) *for the roots of* $P_{100}^{(\alpha,\beta)}(x)$ *for* $\alpha, \beta \in [-.9999, 2]$*. Although the accuracy degrades outside of the interval* $[-1/2, 1/2]$*, it is still sufficient for convergence of the Newton iteration.*

for $\alpha > -\frac{1}{2}, \beta > -1 - \alpha$, where $\psi_k = j_{\alpha,k}/\rho$ [35, eq. (18.16.8)]. The Bessel roots $j_{\alpha,k}$ are not required to a high degree of accuracy and can be computed using, for example, McMahon's expansion [38, p. 247] or Chebyshev approximations [41]. Figure 3.6 (left) shows the accuracy of each of these approximations for the roots of $P_n^{(0.1,-0.3)}$. Although the error terms in (3.19)–(3.21) are only valid for limited ranges of $\alpha$ and $\beta$, Figure 3.6 (right) demonstrates that reasonable approximations are obtained more generally.

As before, we use an interior asymptotic expansion which involves only elementary functions, but is not valid near the endpoints, and a boundary asymptotic expansion involving Bessel functions. The interior asymptotic formula for the Jacobi polynomial is given by Hahn [27] and takes the form

$$\begin{aligned}(3.22)\qquad &\sin^{\alpha+\frac{1}{2}}\left(\tfrac{1}{2}\theta\right)\cos^{\beta+\frac{1}{2}}\left(\tfrac{1}{2}\theta\right)P_n^{(\alpha,\beta)}(\cos\theta)\\ &\quad=\frac{2^{2\rho}}{\pi}\mathrm{B}(n+\alpha+1,n+\beta+1)\sum_{m=0}^{M-1}\frac{f_m(\theta)}{2^m\left(2\rho+1\right)_m}+U_{M,n}^{\alpha,\beta}(\theta),\end{aligned}$$

where $\rho = n + (\alpha+\beta+1)/2$, $\mathrm{B}(\alpha,\beta)$ is the beta function,

$$\begin{aligned}(3.23)\qquad f_m(\theta)&=\sum_{l=0}^{m}\frac{C_{m,l}^{\alpha,\beta}}{l!(m-l)!}\frac{\cos\left(\theta_{n,m,l}\right)}{\sin^l\left(\tfrac{1}{2}\theta\right)\cos^{m-l}\left(\tfrac{1}{2}\theta\right)},\\ \theta_{n,m,l}&=\tfrac{1}{2}\left(2\rho+m\right)\theta-\tfrac{1}{2}\left(\alpha+l+\tfrac{1}{2}\right)\pi,\end{aligned}$$

and

$$C_{m,l}^{\alpha,\beta}=\left(\tfrac{1}{2}+\alpha\right)_l\left(\tfrac{1}{2}-\alpha\right)_l\left(\tfrac{1}{2}+\beta\right)_{m-l}\left(\tfrac{1}{2}-\beta\right)_{m-l},$$

where $(z)_l$ is Pochhammer's notation for the rising factorial. For $\alpha, \beta \in (-\frac{1}{2}, \frac{1}{2})$ the error term $U_{M,n}$ is less than twice the magnitude of the first neglected term, and for all $n \geq 2$, $\alpha, \beta \in (-\frac{1}{2}, \frac{1}{2})$, $\theta \in [\pi/3, 2\pi/3]$ the formula (3.22) converges as $M \to \infty$

[27]. Although $[\pi/3, 2\pi/3]$ is significantly smaller than the corresponding interval for the Legendre polynomial, numerical experiments suggest that in practice (3.22) gives good approximations in a much larger interval. We note also that the summation in (3.23), which is not present in the Legendre case, means (3.22) will have a complexity of $\mathcal{O}(M^2 n)$ compared to the $\mathcal{O}(Mn)$ of (3.9). Since, typically, $M$ is 10–20, and because we no longer have symmetry in the nodes, we should therefore expect the computation time of the Gauss–Jacobi points to be around a factor of 10–20 longer than the Gauss–Legendre.

Again, all known asymptotic formulae for Jacobi polynomials valid near the boundaries involve special functions, and we use the boundary asymptotic expansion involving Bessel functions given by Baratella and Gatteschi [5]:

(3.24)
$$\begin{aligned}&\sin^{\alpha+\frac{1}{2}}\left(\tfrac{1}{2}\theta\right)\cos^{\beta+\frac{1}{2}}\left(\tfrac{1}{2}\theta\right)P_n^{(\alpha,\beta)}(\cos\theta)\\&\quad=\frac{\Gamma\left(n+\alpha+1\right)}{\rho^\alpha n!}\sqrt{\frac{\theta}{2}}\left(J_\alpha(\rho\theta)\sum_{m=0}^{M}\frac{A_m(\theta)}{\rho^{2m}}+\theta J_{\alpha+1}(\rho\theta)\sum_{m=0}^{M-1}\frac{B_m(\theta)}{\rho^{2m+1}}\right)+\mathcal{O}\left(n^{-2M}\right)\end{aligned}$$

for $\alpha, \beta > -1$, where $\rho = n + (\alpha+\beta+1)/2$, $J_\alpha$ and $J_{\alpha+1}$ are Bessel functions of the first kind,

$$g(\theta) = \left(\tfrac{1}{4}-\alpha^2\right)\left(\cot\left(\tfrac{1}{2}\theta\right) - 2/\theta\right) - \left(\tfrac{1}{4}-\beta^2\right)\tan\left(\tfrac{1}{2}\theta\right),$$

$A_0(\theta)$ and $B_0(\theta)$ are as in (3.13), and

$$A_1(\theta) = \frac{1}{8}g'(\theta) - \frac{1+2\alpha}{8}\frac{g(\theta)}{\theta} - \frac{1}{32}g^2(\theta).$$

Similarly to the Legendre case, only these first three terms are known explicitly, but more can be computed numerically using the relations given in [5]. We note that there are other asymptotic formulae involving Bessel functions [3, 13, 52], but these typically require roughly twice as many terms as (3.24) to obtain a similar asymptotic error reduction.

The relation for the derivative of the Jacobi polynomials is a little more complicated than for the Legendre polynomials:

$$\begin{aligned}&(2n+\alpha+\beta)(1-x^2)\frac{d}{dx}P_n^{(\alpha,\beta)}(x)\\&\quad= n(\alpha-\beta-(2n+\alpha+\beta)x)P_n^{(\alpha,\beta)}(x) + 2(n+\alpha)(n+\beta)P_{n-1}^{(\alpha,\beta)}(x),\end{aligned}$$

but the ideas discussed in section 3.2.3 still apply.

### 3.3.1. Computational issues.

**The ratio of gamma functions.** This time it is the constant in front of the boundary asymptotic formula which contains the ratio of gamma functions $\Gamma(n+\alpha+1)/n!$, but this can be dealt with in the same way as discussed in section 3.2.3. Similar ideas can also be used to compute the beta function in (3.22) as

$$2^{2n}\mathrm{B}(n+\alpha+1, n+\beta+1) = \frac{2^{2n}\Gamma\left(n+\alpha+1\right)\Gamma\left(n+\beta+1\right)}{\Gamma\left(2n+\alpha+\beta+1\right)\left(2n+\alpha+\beta+1\right)}. \tag{3.25}$$

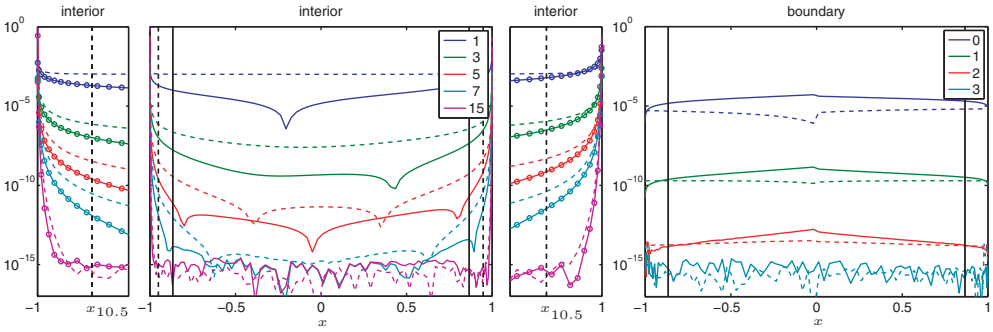


FIG. 3.7. *Error in evaluating the Jacobi polynomial* $P_{100}^{(0.1,-0.3)}$ *and its derivative using the asymptotic formulae* (3.22) *and* (3.25). *Clearly the interior formula* (3.22) *can in practice be used much closer to the boundary than the convergence region of* $\cos(5\pi/6) < x < \cos(\pi/6)$ *suggests (vertical solid line). The first and third panels show that, as with the Legendre polynomial, the boundary formula is only needed for the ten nodes nearest the boundary (vertical dashed lines).*

In particular, again using Stirling's series to approximate the gamma functions and then canceling common terms, we find this ratio may be expressed as

$$2^{(1-\alpha-\beta)/2}\sqrt{\pi}\exp\left(\sum_{j=1}^{\infty}\frac{q_j^{\alpha,\beta}}{n^j}\right)\sqrt{\frac{(n+\alpha)(n+\beta)}{2n+\alpha+\beta}}\frac{S(n+\alpha)S(n+\beta)}{S(2n+\alpha+\beta)(2n+\alpha+\beta+1)},$$

where $S(z)$ is as in (3.15) and

$$q_j^{\alpha,\beta} = \frac{(-1)^{j+1}}{j(j+1)}\left(\alpha^{j+1}+\beta^{j+1}+2\left(\frac{\alpha+\beta}{2}\right)^{j+1}\right),$$

which arrises from similar expansions to (3.17). Each of the terms, and hence (3.25), can be computed accurately and stably.

One final constant requiring careful computation in the Jacobi case is that appearing in (3.2) for the quadrature weights,

$$\frac{\Gamma(n+\alpha+1)\Gamma(n+\beta+1)}{\Gamma(n+\alpha+\beta+1)n!}.$$

Here we note that this ratio is precisely of the form described by Bühring [9, eq. (3)], with $c=0$ and $n \to n+1$, and again asymptotic approximations can be used.

We again stress that although these are asymptotic formulae and not necessarily convergent, they can readily achieve an accuracy of 16 digits for any $n \geq 100$, which is suitable for our purposes.

**3.3.2. Preliminary results.** In the Jacobi case the use of the interior formula is now seemingly hindered by the two constraints that convergence as $M \to \infty$ requires both $\alpha, \beta \in (-1/2, 1/2)$ and $\theta \in [\pi/3, 2\pi/3]$. However, similarly to the interior asymptotic formula for the Legendre polynomial, we find that in practice this second constraint can be ignored for the finite $M$ needed for machine precision, and that the boundary formula is only required for the ten nodes nearest the boundary. Figure 3.7 shows the accuracy of the interior and boundary asymptotic formulae for $P_{100}^{(0.1,-0.3)}(\cos\theta_k^{[0]})$ and $\frac{d}{d\theta}P_{100}^{(0.1,-0.3)}(\theta_k^{[0]})$, and justifies this choice of boundary region. In section 4 we provide evidence that the restriction $\alpha, \beta \in (-\frac{1}{2}, \frac{1}{2})$ can also be largely ignored.

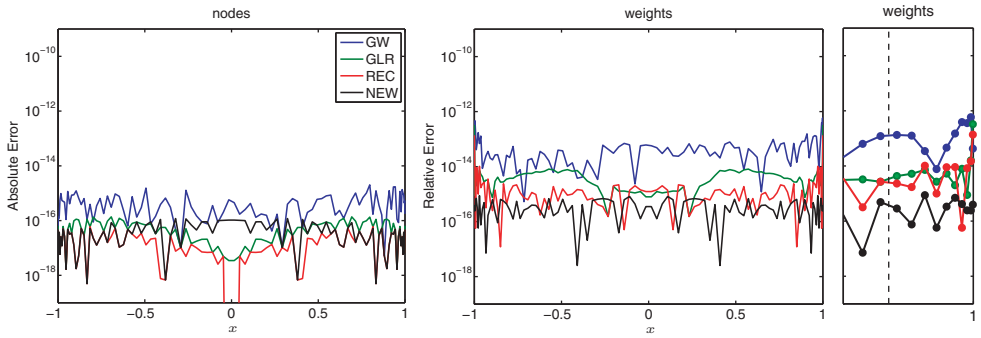


FIG. 4.1. *Error in the computed Gauss–Legendre nodes (left: absolute) and weights (center and right: relative maximum) for* $n = 100$. *The rightmost panel shows the error in the last ten weights.*

**4. Results.** We consider a number of different measurements of the error:

$$\text{absolute error}: \quad \varepsilon_{\text{abs}}\{x_k\} = \max_{k=1,\dots,n} |x_k - x_k^{\text{true}}|;$$

$$\text{relative maximum error}: \quad \varepsilon_{\text{rm}}\{w_k\} = \frac{\max_k |w_k - w_k^{\text{true}}|}{\max_k |w_k^{\text{true}}|};$$

$$\text{maximum relative error}: \quad \varepsilon_{\text{mr}}\{w_k\} = \max_{k=1,\dots,n} \left| \frac{w_k - w_k^{\text{true}}}{w_k^{\text{true}}} \right|.$$

For comparative purposes we also include a measure, similar to that in [23], which weighs the accuracy of the quadratures rather than the nodes and weights directly. Recall that an $n$-point Gauss rule is exact for polynomials of degree up to $2n-1$ and hence, for $0 \le i, j < n$,

$$\begin{aligned}\int_{-1}^{1} (1-x)^\alpha (1+x)^\beta P_s^{(\alpha,\beta)}(x) P_t^{(\alpha,\beta)}(x)\,dx &= \sum_{k=1}^{n} w_k P_s^{(\alpha,\beta)}(x_k) P_t^{(\alpha,\beta)}(x_k) \\ &= \begin{cases} \frac{C_{n,\alpha,\beta}}{2n+\alpha+\beta+1} & \text{if } s = t, \\ 0 & \text{otherwise,} \end{cases}\end{aligned}$$

where $C_{n,\alpha,\beta}$ is the constant in (3.2). This identity can then be used to measure the accuracy of the quadrature rule by selecting arbitrary indices $I = \{i_1, i_2, \dots\}$ and defining

$$\epsilon_{\text{quad}}\{x_k, w_k\} = \max_{s,t \in I} \left| \frac{\delta_{st} C_{n,\alpha,\beta}}{2n+\alpha+\beta+1} - \sum_{k=1}^{n} w_k P_s^{(\alpha,\beta)}(x_k) P_t^{(\alpha,\beta)}(x_k) \right|,$$

where $\delta_{st}$ is the Kronecker delta function. For convenience and reproducibility we choose the set $I$ to be the first 11 Fibonacci numbers (or fewer, if they are greater than $n$ in magnitude), and to avoid introducing additional error we evaluate the polynomials in extended precision. Recall that for $\alpha = \beta = 0$, $C_{n,\alpha,\beta} = 2$.

We now compare our method against the other algorithms described in section 2, using the ORTHPOL [18] implementation of the GW algorithm, the GSL implementation of the recurrence relation [14], and a Fortran implementation of the GLR algorithm supplied by the authors of [23]. Figure 4.1 shows the absolute error in the nodes and maximum relative error in the weights for $n = 100$, and Figure 4.2 shows

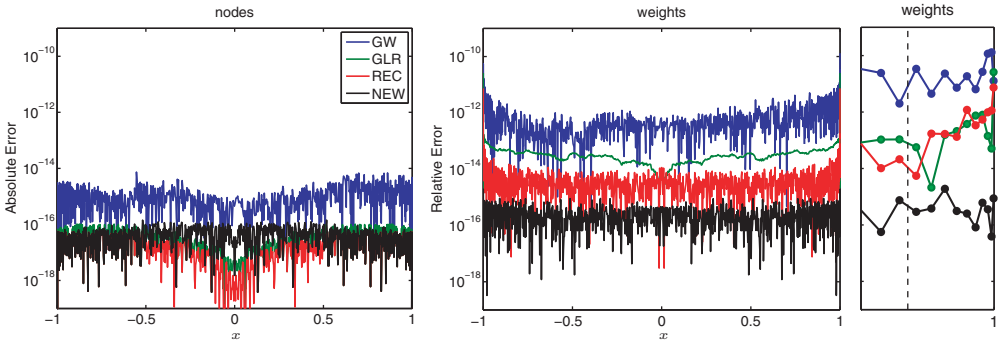


FIG. 4.2. *Error in the computed Gauss–Legendre nodes (left: absolute) and weights (center and right: relative maximum) for* $n = 1000$.

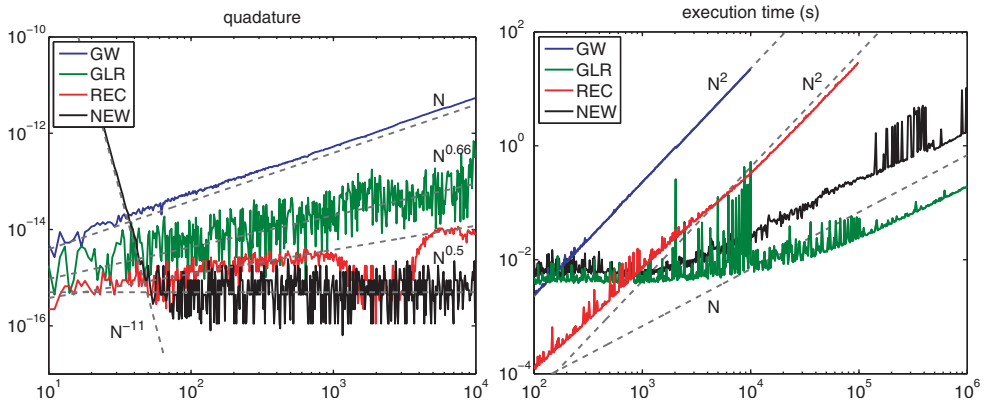


FIG. 4.3. *Maximum quadrature error (left) and computational time (right) in computing Gauss-Legendre nodes and weights. See Figure* 2.1 *for node and weight errors. Note that the timings here are for implementations in different programming languages, and should not be used for direct comparison.*

the same for $n = 1000$. Whereas GLR and REC obtain good relative precision in the nodes near $x = 0$, our Newton iteration operates in $\theta$-space and we only obtain good absolute accuracy. However, the error is below machine precision, and uniform throughout the interval $[-1, 1]$. The error in the weights is also uniform throughout the interval and, particularly near the boundaries, significantly lower than that of the other methods. Figures 2.1 and 4.3 investigate the performance of each of the methods as $n$ is increased, and Table 1.1 summarizes the results of our new method. An almost identical table for the GLR method can be found in [23, Table 3]. In particular, we note the method described in this paper has a complexity of only $\mathcal{O}(n)$ and that the errors in the nodes and weights are essentially[3] $n$-independent.

Figure 4.4 repeats Figure 4.1 for Gauss–Jacobi quadrature with $n = 1000$, $\alpha = 0.1$, and $\beta = -0.3$, where here we compare against Chebfun implementations of GW, REC, and GLR. Again we see that the new method produces a good absolute accuracy of

[3]We observe in Figure 2.1 an $\mathcal{O}(\log(\log(n)))$ growth in the error of the nodes. The error appears to arise in nodes near $x = 0$, and we are as yet unable to account for it. However, since $\log(\log(\texttt{realmax})) < 7$ in MATLAB (and furthermore $\log(\log(\sqrt{2^{-52}})) < 3$), it is unlikely that this will present any practical limitations.

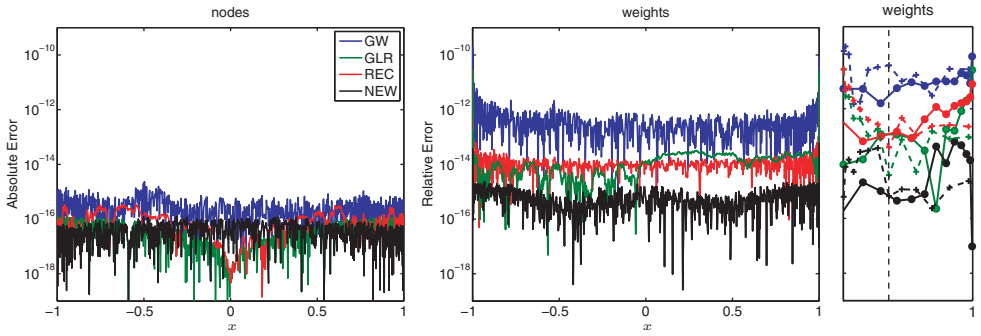


FIG. 4.4. *Error in the computed Gauss–Jacobi nodes (left) and weights (center and right) for* $n = 1000, \alpha = 0.1, \beta = -0.3$. *The rightmost panel shows the errors for the weights closest to* $x = -1$ *(dashed line) and* $x = 1$ *(solid line). The rise in error of the few weights near the boundary nodes can be attributed to error in evaluation of the Bessel function.*

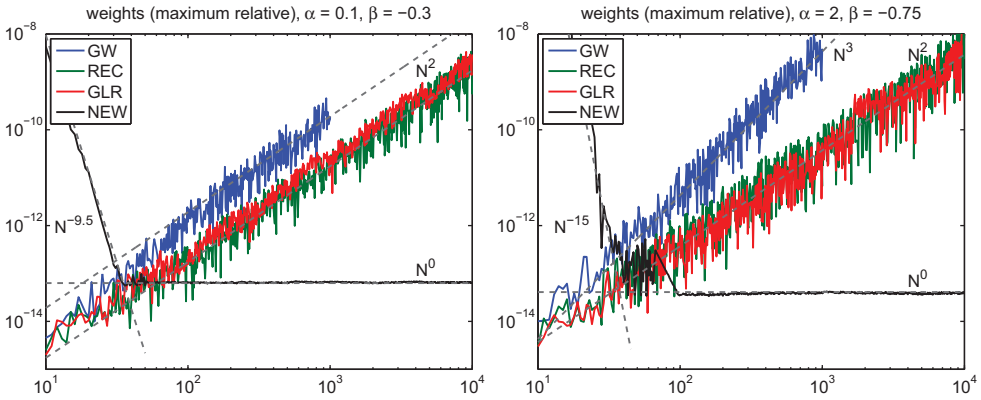


FIG. 4.5. *Observed maximum relative errors in the weights in Gauss–Jacobi for* $\alpha = 0.1, \beta = -0.3$ *(left) and* $\alpha = 2, \beta = -0.75$ *(right). As with the analogous plot for the Gauss–Legendre weights (Figure* 2.1*), we see the error in the new algorithm is independent of* $n$.

TABLE 4.1

*Accuracy and computation time for Gauss–Jacobi nodes and weights computed by the algorithm in this paper for* $n = 10^2, \ldots, 10^6$ *and* $\alpha = 0.1, \beta = -0.3$. *As expected, the computational times are approximately ten times greater than those in Table* 1.1.

| $n$ | $\epsilon_{\text{abs}}\{x_k\}$ | $\epsilon_{\text{rm}}\{w_k\}$ | $\epsilon_{\text{mr}}\{w_k\}$ | $\epsilon_{\text{quad}}\{x_k, w_k\}$ | Time (secs) |
|---|---|---|---|---|---|
| 100 | 1.42e-16 | 3.64e-15 | 4.52e-14 | 9.30e-16 | 0.0418 |
| 1,000 | 2.06e-16 | 8.83e-15 | 6.66e-14 | 7.32e-16 | 0.0532 |
| 10,000 | 1.11e-16 | 3.91e-15 | 6.38e-14 | 7.46e-16 | 0.2503 |
| 100,000 | 4.44e-16 | 3.40e-16 | 1.16e-14 | 7.42e-16 | 1.5685 |
| 1,000,000 | 4.44e-16 | 6.53e-16 | 3.50e-14 | 1.11e-15 | 15.256 |

around machine precision for each of the nodes, and an error in the weights lower than that of the existing methods. Unfortunately, here we find an error of around $10^{-13}$ in the weights near the boundary, which appears directly related to the error in evaluating the Bessel functions $J_\alpha$ and $J_\beta$ at these points. However, we find that this error is fixed independently of $n$ (see Figure 4.5). Tables 4.1 and 4.2 repeat Table 1.1 for Gauss–Jacobi with $\alpha = 0.1, \beta = -0.3$ and $\alpha = 2, \beta = -0.75$, respectively, and show that high and essentially $n$-independent precision is maintained in both the nodes and

TABLE 4.2

*Accuracy and computation time for Gauss–Jacobi nodes and weights computed by the algorithm in this paper for $n = 10^2, \dots, 10^6$ and $\alpha = 2, \beta = -.75$. Even for $\alpha$ and $\beta$ outside $[-\frac{1}{2}, \frac{1}{2}]$, good accuracy is achieved.*

| $n$ | $\epsilon_{\text{abs}}\{x_k\}$ | $\epsilon_{\text{rm}}\{w_k\}$ | $\epsilon_{\text{mr}}\{w_k\}$ | $\epsilon_{\text{quad}}\{x_k, w_k\}$ | Time (secs) |
|---|---|---|---|---|---|
| 100 | 2.11e-16 | 6.77e-15 | 4.13e-14 | 4.45e-15 | 0.0412 |
| 1,000 | 1.46e-16 | 1.02e-14 | 4.42e-14 | 4.49e-15 | 0.0515 |
| 10,000 | 1.11e-16 | 8.28e-15 | 3.53e-14 | 4.56e-15 | 0.1858 |
| 100,000 | 1.11e-16 | 1.23e-15 | 5.46e-14 | 4.37e-15 | 2.0865 |
| 1,000,000 | 1.11e-16 | 7.01e-15 | 7.31e-14 | 4.49e-15 | 15.713 |

the weights for the Gauss–Jacobi quadrature rules. As expected, the Gauss–Jacobi rules take around a factor of 10 longer to compute than the Gauss–Legendre, due to the extra summation in (3.23).

**5. Future extensions.** We have concentrated on computing Gauss–Legendre and Gauss–Jacobi quadrature rules in double precision, but the methodology we have described can be extended in a number of directions. Here we discuss these briefly.

**Higher precision.** Throughout this paper we have focused on double precision accuracy, but quadratic (or even variable) precision could also be achieved using the same techniques. However, additional terms in the boundary asymptotic formulae are required as well as a better understanding of when/if the expansions diverge. Furthermore, since only the first few terms in the boundary expansions are readily available, the minimum value of $n$ for which full quadratic precision could be reached would be larger than the $n = 100$ for double precision.

**Gauss–Radau and Gauss–Lobatto quadrature.** Gauss–Radau and Gauss–Lobatto are variations on the Gauss–Legendre quadrature rule where one or more of the endpoints are preassigned, and the approach discussed in this paper is equally applicable here. For example, the Gauss–Radau nodes and weights satisfy [1, eq. (25.4.31)]

$$\frac{P_{n-1}(x_k) + P_n(x_k)}{1 + x_k} = 0, \quad k = 2, \dots n, \quad w_k = \begin{cases} \frac{1}{(1-x_k)[P'_{n-1}(x_k)]^2}, & k = 2, \dots, n, \\ \frac{2}{n^2}, & k = 1 \end{cases}$$

while for the Gauss–Lobatto rule [1, eq. (25.4.32)]

$$\frac{x_k P_{n-1}(x_k) - P_n(x_k)}{1 - x_k^2} = 0, \quad k = 2, \dots n-1, \quad w_i = \begin{cases} \frac{2}{n(n-1)[P_{n-1}(x_k)]^2}, & k = 2, \dots, n-1, \\ \frac{2}{n(n-1)}, & k = 1, n. \end{cases}$$

Initial guesses can be obtained by noting that the interior Radau nodes are also the zeros of $P_{n-1}^{(0,1)}$, and the interior Lobatto nodes are the zeros of $P_{n-2}^{(1,1)}$ [35, eq. (18.9.6)], for which the approximations in section 3.3 can be used. One could also consider the more general case of Gauss–Jacobi–Radau [20].

**Hermite and Laguerre quadrature.** The Hermite and Laguerre polynomials also have asymptotic expansions [40, 45] which could be used within Newton iterations to compute Gauss–Hermite and Gauss–Laguerre quadrature nodes and weights. In

particular, the nodes are roots of Hermite and Laguerre polynomials, $H_n(x)$ and $L_n(x)$, and the weights are given by

$$w_k = \frac{\sqrt{\pi}2^{n+1}n!}{[H_n'(x_k)]^2} \qquad \text{and} \qquad w_k = \frac{1}{x_k[L_n'(x_k)]^2},$$

respectively [1, eqs. (25.4.45) and (25.4.46)]. In practice, one usually works with *normalized* Hermite and Laguerre polynomials and *scaled* quadrature weights to avoid problems of overflow and underflow [23].

**Polynomial evaluation.** The asymptotic formulae used for evaluations within the Newton iterations could also be used to evaluate the $P_n^{(\alpha,\beta)}(x_k)$ at any point within $[-1, 1]$. However, a number of the techniques described in this paper rely on the fact that we are evaluating near a root of the Jacobi polynomial, and would not apply in general. A few brief experiments suggest that an accuracy of 14–15 digits can be achieved, again in $\mathcal{O}(n)$ operations, but we have not investigated this further. Bogaert, Michiels, and Fostier [7] apply a similar technique for evaluating Legendre polynomials.

**Barycentric weights.** Given $n$ function values $f_1, \ldots, f_n$ at a distinct set of points $x_1, \ldots, x_n$, and the barycentric weights $\{v_k\}$

$$v_k = C \Bigg/ \prod_{j\neq k}^{n} (x_k - x_j) \,,$$

where the constant $C$ is arbitrary, the barycentric interpolation formula [6]

$$p(x) = \sum_{k=1}^{n} \frac{v_k f_k}{x - x_k} \Bigg/ \sum_{k=1}^{n} \frac{v_k}{x - x_k}$$

gives the unique polynomial interpolant throughout the data $\{x_k, f_k\}_{k=1}^n$. An equivalent formulation of the barycentric weights is [48, Chap. 5]

$$v_k = C \,/ l'(x_k) \,, \qquad l(x) = \prod_{k=1}^{n} (x - x_k).$$

Thus if the $x_k$ are the roots of the Jacobi polynomial $P_n^{(\alpha,\beta)}(x)$, the derivative values $P_n^{(\alpha,\beta)}(x_k)$ needed to determine the corresponding barycentric weights can be computed in exactly the same way as for the quadrature weights [29, 50]. As such, we now have a fast, accurate, and stable method [30] of evaluating Jacobi interpolants, even at millions of points.

**Software.** MATLAB code for the Gauss–Legendre and Gauss–Jacobi algorithms described in this paper can be found in Chebfun's `legpts` and `jacpts` functions, respectively [46, revision 2337 and above]. We hope to soon have a software library QUADPTS [28] which contains both MATLAB and C implementations, a Python interface, and ultimately the other extensions described above.

**6. Conclusion.** We have presented an algorithm which, for any $n \geq 100$, computes the $n$-point Gauss–Legendre or Gauss–Jacobi nodes and weights in a total of $\mathcal{O}(n)$ operations. The algorithm is easily vectorized, making it efficient in array-based languages like MATLAB, and can be easily parallelized if required. Furthermore,

we have demonstrated that the algorithm is extremely accurate, so that nodes and weights computed to absolute and relative accuracies of almost machine precision, respectively. MATLAB code is available as a standalone package [28], and is also available through Chebfun's `legpts` and `jacpts` commands [46].

We hope this new approach will remove the artificial limit on how researchers feel they can use Gauss quadratures, and open up a fascinating window into numerical algorithms built on asymptotic formulae.

**Note added in proof.** Shortly after submitting this paper for publication the authors became aware of recent work by Bogaert, Michiels, and Fostier [7] which also presents a method for computing the Gauss–Legendre nodes and weights based upon asymptotic expansions. The approach proposed in [7] is similar to ours in two respects: asymptotic expansions are used for the fast evaluation of a Legendre polynomial and Newton's method is used for computing the Gauss–Legendre nodes. Bogaert, Michiels, and Fostier also suggest using both interior and exterior asymptotic formulae, but while their interior formula is equivalent to the one used here in (3.9), they derive an alternative expansion for the boundary near $\pm 1$. Their paper is motivated by a fast evaluation scheme for Legendre series, and as such is not concerned with extending the method to Gauss–Jacobi quadratures. Excellent accuracy in the nodes is demonstrated for remarkably high degree quadratures, but little direct evidence of the accuracy in the weights is given.

**Acknowledgments.** The authors would like to thank Andreas Glaser and Vladimir Rokhlin for supplying a Fortran implementation of the GLR algorithm, Iain Smears for translating the French papers of Stieltjes and Darboux, and Nick Trefethen and Stefan Güttel for initial discussions.

## REFERENCES

[1] M. Abramowitz and I. A. Stegun, *Handbook of Mathematical Functions*, Dover, New York, 1964.

[2] B. K. Alpert and V. Rokhlin, *A fast algorithm for the evaluation of Legendre expansions*, SIAM J. Sci. Statist. Comput., 12, (1991), pp. 158–179.

[3] X.-X. Bai and Y.-Q. Zhao, *A uniform asymptotic expansion for Jacobi polynomials via uniform treatment of Darboux's method*, J. Approx. Theory, 148 (2007), pp. 1–11.

[4] D. Bailey, K. Jeyabalan, and L. S. Xiaoye, *A comparison of three high-precision quadrature schemes*, Experiment. Math., 14 (2005), pp. 317–329.

[5] P. Baratella and L. Gatteschi, *The bounds for the error term of an asymptotic approximation of Jacobi polynomials*, in Orthogonal Polynomials and Their Applications, Lecture Notes in Math. 1329, 1988, pp. 203–221.

[6] J.-P. Berrut and L. N. Trefethen, *Barycentric Lagrange interpolation*, SIAM Rev., 46 (2004), pp. 501–517.

[7] I. Bogaert, B. Michiels, and J. Fostier, $\mathcal{O}(1)$ *computation of Legendre polynomials and Gauss–Legendre nodes and weights for parallel computing*, SIAM J. Sci. Comput., 34 (2012), pp. C83–C101.

[8] M. Branders, R. Piessens, and M. Demeue, *Rational approximations for zeros of Bessel functions*, J. Comput. Phys., 42 (1981), pp. 403–405.

[9] W. Bühring, *An asymptotic expansion for a ratio of products of gamma functions*, Internat. J. Math. Math. Sci., 24 (2000), pp. 505–510.

[10] J. Burkardt, *J. Burkardt's software collection*, http://people.sc.fsu.edu/~jburkardt/c_src/legendre_rule_fast/legendre_rule_fast.html (2009).

[11] G. Darboux, *Mémoire sur l'approximation des fonctions de très-grands nombres, et sur une classe étendue de développements en série*, J. Math. Pures Appl. (3), 4 (1878), pp. 5–56.

[12] A. de Moivre, *The Doctrine of Chances, or, A Method of Calculating the Probability of Events in Play*, W. Pearson, London, 1718.

[13] D. Elliott, *Uniform asymptotic expansions of the Jacobi polynomials and an associated function*, Math. Comp., 25 (1971), pp. 309–315.
[14] M. Galassi et al., *GNU Scientific Library Reference Manual*, 3rd ed., GSL version 1.15, http://www.gnu.org/software/gsl/ (2011).
[15] L. Gatteschi, *Una nuova rappresentazione asintotica dei polinomi di Jacobi*, Rend. Sem. Mat. Univ. Politec. Torino, 27 (1967-68), pp. 165–184.
[16] L. Gatteschi and G. Pittaluga, *An asymptotic expansion for the zeros of Jacobi polynomials*, in Mathematical Analysis. Teubner-Texte Math., 79 (1985), pp. 70–86.
[17] C. F. Gauss, *Methodus Nova Integralium Valores per Approximationem Inveniendi*, Gottingen, Germany, 1814.
[18] W. Gautschi, *Algorithm* 726: *ORTHPOL; a package of routines for generating orthogonal polynomials and Gauss-type quadrature rules*, ACM Trans. Math. Software, 20 (1994), pp. 21–62.
[19] W. Gautschi, *A survey of Gauss–Christoffel quadrature formulae*, in E. B. Christoffel: The Influence of His Work in Mathematics and the Physical Sciences, Birkhäuser, Basel, 1981, pp. 72–147.
[20] W. Gautschi, *Gauss–Radau formulae for Jacobi and Laguerre weights functions*, Math. Comput. Simulation, 54 (2000), pp. 403–412.
[21] W. Gautschi and C. Giordano, *Luigi Gatteschi's work on asymptotics of special functions and their zeros*, Numer. Algorithms, 49 (2008), pp. 11–31.
[22] H. Gerber, *First hundred zeros of $J_0(x)$ accurate to* 19 *significant figures*, Math. Comp., 18 (1964), pp. 319–322.
[23] A. Glaser, X. Liu, and V. Rokhlin, *A Fast Algorithm for the Calculation of the Roots of Special Functions*, SIAM J. Sci. Comput., 29 (2007), pp. 1420–1438.
[24] G. H. Golub and J. H. Welsch, *Calculation of Gauss quadrature rules*, Math. Comp., 23 (1969), pp. 221–230.
[25] E. Grosse, `gaussq.f` GO Quadrature Library, http://www.netlib.org/go/gaussq.f (1983).
[26] M. Gu and S. C. Eisenstat, *A divide-and-conquer algorithm for the symmetric tridiagonal eigenproblem*, SIAM J. Matrix Anal. Appl., 16 (1995), pp. 172–191.
[27] E. Hahn, *Asymptotik bei Jacobi-polynomen und Jacobi-funktionen*, Math. Z., 171 (1980), pp. 201–226.
[28] N. Hale and A. Townsend, `quadpts` *Library*, https://github.com/nickhale/quadpts (2012).
[29] N. Hale and L. N. Trefethen, *Chebfun and numerical quadrature*, Sci. China Ser. A, 55 (2012), pp. 1749–1760.
[30] N. J. Higham, *The numerical stability of barycentric Lagrange interpolation*, IMA J. Numer. Anal., 24 (2004), pp. 547–556.
[31] C. G. J. Jacobi, *Üeber Gauss neue Methode, die Werthe der Integrale näherungsweise zu finden*, J. Reine Angew. Math., 1 (1826), pp. 301–307.
[32] C. G. J. Jacobi, *Üeber eine besondere Gattung algebraischer Functionen, die aus der Entwicklung der Function* $(1 - 2xz + z^2)^{1/2}$ *entstehen*, J. Reine Angew. Math., 2 (1827), pp. 223–226.
[33] F. Lether, *On the construction of Gauss–Legendre quadrature rules*, J. Comput. Appl. Math., 4 (1978), pp. 47–52.
[34] NAG Fortran Library Version 23, The Numerical Algorithms Group, Oxford, UK, 2012.
[35] F. W. J. Olver, D. W. Lozier, R. F. Boisvert, and C. W. Clark, *NIST Handbook of Mathematical Functions*, Cambridge University Press, Cambridge, UK, 2010.
[36] F. W. J. Olver, *The asymptotic solution of linear differential equations of the second order for large values of a parameter*, R. Soc. Lond. Philos. Trans. Ser. A Math. Phys. Eng. Sci., 247 (1954), pp. 307–327.
[37] F. W. J. Olver, *A paradox in asymptotics* SIAM J. Math. Anal., 1 (1970), pp. 533–534.
[38] F. W. J. Olver, *Asymptotics and Special Functions*, Academic, New York, 1974.
[39] K. Petras, *On the computation of the Gauss–Legendre quadrature form with a given precision*, J. Comput. Appl. Math., 112 (1999), pp. 253–267.
[40] O. Perron, *Über das Verhallen einer ausgearteten hypergeometrischen Reiche bei unbegrenztem Wachstum eines Parameters* J. Reine Angew. Math., 151 (1921), pp. 63–87.
[41] R. Piessens, *Chebyshev series approximations for the zeros of the Bessel functions*, J. Comput. Phys., 53 (1984), pp. 188–192.
[42] W. H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery, *Numerical Recipes: The Art of Scientific Computing*, Cambridge University Press, Cambridge, UK, 2007.
[43] T. J. Stieltjes, *Sur les polynômes de Legendre*, Ann. Fac. Sci. Toulouse, 4 (1890), G1–G17.
[44] P. N. Swarztrauber, *On computing the points and weights for Gauss–Legendre quadrature*, SIAM J. Sci. Comput., 24 (2003), pp. 945–954.

[45] G. Szegő, *Orthogonal Polynomials*, AMS, Providence, RI, 1939.
[46] L. N. Trefethen et al., Chebfun Version 4.2, The Chebfun Development Team, http://www.maths.ox.ac.uk/chebfun (2011).
[47] F. Tisseur, *Newton's method in floating point arithmetic and iterative refinement of generalized eigenvalue problems*, SIAM J. Matrix Anal. Appl., 22 (2001), pp. 1038–1057.
[48] L. N. Trefethen, *Approximation Theory and Approximation Practice*, SIAM, Philadelphia, to appear.
[49] F. G. Tricomi, *Sugli zeri dei polinomi sferici ed ultrasferici*, Ann. Mat. Pura Appl., 31 (1950), pp. 93–97.
[50] H. Wang and S. Xiang, *On the convergence rates of Legendre approximation*, Math. Comp., 81 (2012), pp. 861–877.
[51] H. S. Wilf, *Mathematics for the Physical Sciences*, Wiley, New York, 1962.
[52] R. Wong and Y.-Q. Zhao, *Estimates for the error term in a uniform asymptotic expansion of the Jacobi polynomials*, Anal. Appl., 1 (2003), pp. 231–241.
[53] S. Xiang and F. Bornemann, *On the convergence rates of Gauss and Clenshaw–Curtis quadrature for functions of limited regularity*, SIAM J. Numer. Anal., 50 (2012), pp. 2581–2587.
[54] E. Yakimiw, *Accurate Computation of weights in classical Gauss–Christoffel quadrature rules*, J. Comput. Phys., 129 (1996), pp. 406–430.