SIAM J. Sci. Comput. © 2014 Society for Industrial and Applied Mathematics
Vol. 36, No. 3, pp. A1008–A1026

ITERATION-FREE COMPUTATION OF GAUSS–LEGENDRE QUADRATURE NODES AND WEIGHTS[*]

I. BOGAERT[†]

Abstract. Gauss–Legendre quadrature rules are of considerable theoretical and practical interest because of their role in numerical integration and interpolation. In this paper, a series expansion for the zeros of the Legendre polynomials is constructed. In addition, a series expansion useful for the computation of the Gauss–Legendre weights is derived. Together, these two expansions provide a practical and fast iteration-free method to compute individual Gauss–Legendre node-weight pairs in $\mathcal{O}(1)$ complexity and with double precision accuracy. An expansion for the barycentric interpolation weights for the Gauss–Legendre nodes is also derived. A C++ implementation is available online.

Key words. Legendre polynomial, Gauss–Legendre quadrature, asymptotic series, parallel computing

AMS subject classifications. 65D32, 26C10, 34E05

DOI. 10.1137/140954969

1. Introduction.

The zeros of the Legendre polynomials [20] are of fundamental importance because they constitute the nodes of Gauss–Legendre (GL) quadrature rules, which are optimal integration rules for polynomial functions [2, 3, 4]. Tabulated GL rules with a small number of points are routinely used for the numerical approximation of integrals. However, applications using GL rules with many (hundreds, thousands, or even more) points are becoming more widespread. For example, GL quadrature rules are useful for the computation of weights occurring in the barycentric Lagrange interpolation formula [11, 23] on the GL nodes. Also, many-point GL quadrature rules are used in spectral methods and to discretize integrals in the three-dimensional (3-D) multilevel fast multipole method (see [24, p. 81]). For such applications, tabulation of the GL rules may no longer be practical and their computation becomes necessary.

Because of the lack of closed-form analytical formulas for the nodes, many different methods have been proposed to numerically compute GL quadrature nodes. An excellent overview of these methods has been given in [10]. For example, the Golub–Welsch algorithm [8] can be used for general Gaussian quadrature rules. However, for the specific case of GL quadrature, the current state of the art is to use asymptotic expansions of the Legendre polynomials together with local zero finding algorithms to find the GL nodes. The asymptotic expansions are then once again used to compute the GL weights. Both in [1] (for GL quadrature) and [10] (for the more general Gauss–Jacobi quadrature), this approach was used to allow the computation of individual GL nodes and weights in $\mathcal{O}(1)$ complexity. This implies an $\mathcal{O}(n)$ complexity for the entire n-point GL rule, which is on par with the algorithm by Glaser, Liu, and Rohklin [7]. The advantage of the asymptotic expansion methods is that the node-weight pairs can be computed independently from each other, which

[*]Submitted to the journal's Methods and Algorithms for Scientific Computing section January 29, 2014; accepted for publication March 11, 2014; published electronically May 22, 2014. This work was supported by a postdoctoral grant from the Fund for Scientific Research Flanders (FWO-Vlaanderen).
http://www.siam.org/journals/sisc/36-3/95496.html

[†]Department of Information Technology (INTEC), Ghent University, Sint-Pietersnieuwstraat 41, 9000 Gent, Belgium (ignace.bogaert@ugent.be).

is critical for parallelization purposes. In fact, the part of [1] that is concerned with the computation of the Legendre polynomials was specifically motivated by such a parallelization effort: the scalable parallel computation of the translation operator in the 3-D multilevel fast multipole method [16, 15]. With current hardware becoming faster mainly by adding more CPU cores, it is clear that parallelization potential becomes ever more important. This also holds true for GL quadrature rules.

In this work, these asymptotic expansion methods will be developed further by providing expansions for the computation of the GL nodes and weights themselves, instead of for the Legendre polynomials. Clearly, this avoids the use of a zero finding algorithm, resulting in an iteration-free computation of the GL nodes. For the weights, no iterations were needed in the first place, but special attention was required for the computation of the derivative of the Legendre polynomials, especially near the boundaries of the domain [10]. The expansion proposed in this work passes through the same computations but, because these are done symbolically, nearly all the cancellations and other numerical difficulties are removed from the final expansion. Finally, an expansion is derived for the barycentric interpolation weights associated with the Legendre nodes.

Remarkably simple expansions are obtained, allowing for a very fast and accurate evaluation, as will be evidenced by the numerical results. An added advantage is that the proposed expansions are uniform, i.e., the same code runs for all relevant nodes and weights in a GL quadrature rule. This is in contrast with [1] and [10], where two separate expansions for the Legendre polynomials were used to cover the full range $[-1, 1]$. In a parallel computing environment, this can be very advantageous for achieving a good load balancing. Indeed, in [10] it was reported that the boundary expansion is three orders of magnitude slower than the internal expansion. When the GL nodes are uniformly divided over P parallel processes, this will lead to some of them requiring more time than others, therefore making poor use of computational resources. This issue is solved by the expansions proposed in this paper. A final advantage is that the evaluation of Bessel functions, necessary to evaluate the Legendre polynomials near the edges of the domain $[-1, 1]$, is no longer required in the expansions provided here. A C++ code implementing the formulas from this paper is available online [12].

The paper is organized as follows. In section 2, the expansion for the GL nodes will be derived, followed be the expansion for the weights in section 3. In section 4, auxiliary asymptotic expansions related to the Bessel functions are derived, along with a brief discussion of the numerical handling of the node and weight expansions. Subsequently, section 5 shows the accuracy and speed of the proposed approach by means of numerical examples in a double precision C++ implementation. Finally, section 6 details how the techniques proposed in this paper are extended to the computation of barycentric interpolation weights for the GL nodes.

2. Expansion for the GL nodes.

The GL nodes will be denoted as $x_{n,k}$ $\forall k \in [1, n]$, ordered such that $x_{n,k+1} < x_{n,k}$ $\forall k \in [1, n-1]$. Due to the even/odd symmetry of the Legendre polynomials of even/odd degree, respectively, the following reflection property holds:

$$x_{n,n-k+1} = -x_{n,k}. \tag{2.1}$$

A corollary is

$$x_{2n-1,n} = 0. \tag{2.2}$$

Properties (2.1) and (2.2) ensure that all GL nodes can be computed from those with $k \in [1, \lfloor \frac{n}{2} \rfloor]$.

As explained in [1], the nodes exhibit quadratic clustering near the points ± 1 for large n. This leads to the conclusion that, in a fixed-precision floating point format, it is numerically advantageous to compute not $x_{n,k}$ but rather $\theta_{n,k}$ such that

$$x_{n,k} = \cos \theta_{n,k}. \tag{2.3}$$

In addition to having numerical advantages, using the $\theta_{n,k}$ form of the GL nodes fits more closely with the asymptotic series that will be leveraged, which is why the rest of this paper will be focused on computing $\theta_{n,k}$. If needed, the node $x_{n,k}$ can be cheaply obtained by means of (2.3).

The process of finding the asymptotic expansion for $\theta_{n,k}$ requires an asymptotic expansion of the Legendre polynomials. In subsection 2.1, this expansion will be discussed. Then, in subsection 2.2, an initial approximation for $\theta_{n,k}$ is refined by means of the Lagrange inversion theorem (see [20, subsection 1.10(vii)]). This approach can be used to get arbitrarily high-order corrections to the initial approximation, such that the error on the GL nodes can be controlled for n sufficiently large. For small n, the expansions cannot be used and a tabulation strategy similar to [1] is adopted.

2.1. Expansion for the Legendre polynomials. The starting point of the derivation is the following series expansion for the Legendre polynomials:

$$P_n(\cos\theta) = q(\theta) \sum_{\nu=0}^{\infty} a_\nu(\theta) J_\nu \left(\frac{\theta}{v_n} \right) v_n^\nu \tag{2.4}$$

with $J_\nu(\cdot)$ the Bessel function of the first kind of order ν and

$$v_n = \frac{1}{n + \frac{1}{2}}, \;\; q(\theta) = \sqrt{\frac{\theta}{\sin\theta}}. \tag{2.5}$$

This expansion was proposed in [5], where the functions $a_\nu(\theta)$ are defined as follows:

$$a_\nu(\theta) = (2\theta)^\nu \frac{\Gamma(\nu + \frac{1}{2})}{\sqrt{\pi}} \psi_\nu(\theta). \tag{2.6}$$

The functions $\psi_\nu(\theta)$ are the series expansion coefficients of

$$\left[1 + \sum_{m=2}^{\infty} t^{m-1} \phi_m(\theta) \right]^{-\frac{1}{2}} = \sum_{\nu=0}^{\infty} \psi_\nu(\theta) t^\nu \tag{2.7}$$

with

$$\phi_\nu(\theta) = \frac{1}{\nu!} (2\theta)^{1-\nu} \frac{J_{\nu-\frac{1}{2}}(\theta)}{J_{\frac{1}{2}}(\theta)}. \tag{2.8}$$

The first few $a_\nu(\theta)$ are given by

$$\begin{aligned} a_0(\theta) &= 1, \\ a_1(\theta) &= \frac{1}{8} \frac{\theta\cos\theta - \sin\theta}{\theta \sin\theta}, \\ a_2(\theta) &= \frac{1}{128} \frac{6\theta\sin\theta\cos\theta - 15\sin^2\theta + \theta^2(9 - \sin^2\theta)}{\theta^2 \sin^2\theta}, \\ a_3(\theta) &= \frac{5}{1024} \frac{((\theta^3 + 21\theta)\sin^2\theta + 15\theta^3)\cos\theta - ((3\theta^2 + 63)\sin^2\theta - 27\theta^2)\sin\theta}{\theta^3 \sin^3\theta}, \end{aligned} \tag{2.9}$$

and more can be computed with relative ease using computer algebra software. It is worthwhile to point out the similarities between expansion (2.4) and the boundary expansion used in [10, equation (3.12)]:

$$P_n(\cos\theta) \approx q(\theta)\left[J_0\left(\frac{\theta}{v_n}\right)\sum_{m=0}^{M} A_m(\theta)v_n^{2m} + \theta J_1\left(\frac{\theta}{v_n}\right)\sum_{m=0}^{M-1} B_m(\theta)v_n^{2m+1}\right]. \tag{2.10}$$

Indeed, when the Bessel functions in (2.4) are replaced with Bessel functions of zeroth and first order (by means of the recurrence relation), a form very similar to (2.10) is obtained. In addition, the coefficients in (2.10) that are known explicitly (i.e., $A_0(\theta)$, $B_0(\theta)$, and $A_1(\theta)$; see equation (3.13) in [10]) are exactly reproduced. Whether these expansions are truly identical is beyond the scope of this paper, but it seems likely that the derivations in the rest of this paper could have also been based on (2.10) instead of (2.4). The reason for choosing (2.4) is the fact that the functions $a_\nu(\theta)$ are much easier to compute. Indeed, only derivatives are needed to compute $a_\nu(\theta)$, while a complicated recurrence containing indefinite integrations links the coefficients $A_m(\theta)$ and $B_m(\theta)$.

According to [5], series (2.4) converges uniformly in the usual sense in any interval $\theta \in [0, \theta_0 - \epsilon]$ with $\theta_0 = 2(\sqrt{2}-1)\pi \approx 2.6026$ and $0 < \epsilon < \theta_0$. More importantly, it converges uniformly in the asymptotic sense (for large n) in any interval $0 \le \theta \le \pi - \epsilon'$ with $0 < \epsilon < \pi$. Crucially, this region contains the closed interval $\theta \in [0, \frac{\pi}{2}]$. This is important because, since k may be restricted to $[1, \lfloor\frac{n}{2}\rfloor]$ without loss of functionality, $\theta_{n,k}$ will always be in the interval $[0, \frac{\pi}{2}]$. Therefore, series (2.4) will be uniformly valid in the neighborhood of all $\theta_{n,k}$ of interest.

2.2. Lagrange inversion theorem. It is clear that, for very large n, it is clear that only the first term of (2.4) contributes significantly to the Legendre polynomial:

$$P_n(\cos\theta) = q(\theta)J_0\left(\frac{\theta}{v_n}\right) + \mathcal{O}(v_n). \tag{2.11}$$

Therefore, it makes sense to approximate the zeros of the Legendre polynomial in terms of the zeros of the Bessel function:

$$\theta_{n,k} \approx v_n j_{0,k} = \alpha_{n,k}. \tag{2.12}$$

This idea is of course not new. In fact, it was further refined in [19, p. 469], where the following result was given:

$$\theta_{n,k} = \alpha_{n,k} + v_n^2 \frac{\alpha_{n,k}\cot\alpha_{n,k} - 1}{8\alpha_{n,k}} + \alpha_{n,k}\mathcal{O}\left(v_n^4\right). \tag{2.13}$$

The goal of this work is to find a version of this expression with an error term containing a higher exponent, such that higher-order convergence is achieved. This will be done by means of the Lagrange inversion theorem (see [20, subsection 1.10(vii)]).

To apply the Lagrange inversion theorem, a Taylor series expansion of $q^{-1}(\theta)$ $P_n(\cos\theta)$ around the point $\alpha_{n,k}$ will be constructed:

$$q^{-1}(\theta)P_n(\cos\theta) = \sum_{p=0}^{\infty} f_p(\alpha_{n,k}, v_n)\frac{(\theta - \alpha_{n,k})^p}{p!}, \tag{2.14}$$

where

$$f_p(\theta, v_n) = \frac{\partial^p}{\partial \theta^p} \sum_{\nu=0}^{\infty} a_\nu(\theta) J_\nu\left(\frac{\theta}{v_n}\right) v_n^\nu. \tag{2.15}$$

The evaluation of these derivatives gives rise to very long expressions containing Bessel functions of all orders. However, as mentioned, any Bessel function can be replaced with a combination of Bessel functions of zeroth and first order by means of the recurrence relations. Crucially, the expansion point is $\alpha_{n,k} = v_n j_{0,k}$, which means that the zeroth-order Bessel function vanishes. Hence, all Bessel functions can be replaced with a rational function of $\alpha_{n,k}$ and v_n, times $\mathcal{J}_k = J_1(j_{0,k})$:

$$J_2\left(\frac{\alpha_{n,k}}{v_n}\right) = \mathcal{J}_k \frac{2v_n}{\alpha_{n,k}}, \tag{2.16}$$

$$J_3\left(\frac{\alpha_{n,k}}{v_n}\right) = \mathcal{J}_k \frac{8v_n^2 - \alpha_{n,k}^2}{\alpha_{n,k}^2}, \tag{2.17}$$

$$J_4\left(\frac{\alpha_{n,k}}{v_n}\right) = \mathcal{J}_k \frac{48v_n^3 - 8\alpha_{n,k}^2 v_n}{\alpha_{n,k}^3}, \tag{2.18}$$

$$J_5\left(\frac{\alpha_{n,k}}{v_n}\right) = \mathcal{J}_k \frac{384v_n^4 - 72\alpha_{n,k}^2 v_n^2 + \alpha_{n,k}^4}{\alpha_{n,k}^4}. \tag{2.19}$$

For higher-order Bessel functions, similar expressions hold. Using this knowledge, the Taylor coefficients (2.15) can be evaluated up to any order in v_n. For example,

(2.20)
$$f_0(\alpha_{n,k}, v_n) = \mathcal{J}_k \left[a_1(\alpha_{n,k}) v_n + \left(\frac{2a_2(\alpha_{n,k})}{\alpha_{n,k}} - a_3(\alpha_{n,k})\right) v_n^3 + \mathcal{O}\left(v_n^5\right)\right],$$

(2.21)
$$f_1(\alpha_{n,k}, v_n) = \mathcal{J}_k \left[-a_0(\alpha_{n,k}) v_n^{-1} + \left(a_1'(\alpha_{n,k}) + a_2(\alpha_{n,k}) - \frac{a_1(\alpha_{n,k})}{\alpha_{n,k}}\right) v_n^1 + \mathcal{O}\left(v_n^3\right)\right].$$

Here, $a_1'(\theta)$ is the derivative of $a_1(\theta)$ with respect to θ. Now, with the help of the Lagrange inversion theorem, series (2.14) can be inverted. For example, a third-order expansion is given by

$$\theta_{n,k} = \alpha_{n,k} - \frac{f_0}{f_1} - \frac{f_2 f_0^2}{2f_1^3} + \frac{\left(f_3 f_1 - 3f_2^2\right) f_0^3}{6f_1^5} + \mathcal{O}\left(f_0^4\right), \tag{2.22}$$

but higher-order expansions can be generated at will. The arguments of the functions $f_p(\theta, v_n)$ have been omitted to avoid overburdening the notation. Using only the first-order term in f_0, and the series (2.20) and (2.21), the following is obtained:

$$\begin{aligned} \theta_{n,k} &= \alpha_{n,k} + v_n^2 \frac{a_1(\alpha_{n,k})}{a_0(\alpha_{n,k})} + \mathcal{O}\left(v_n^4\right), \\ &= \alpha_{n,k} + v_n^2 \frac{1}{8} \frac{\alpha_{n,k}\cos\alpha_{n,k} - \sin\alpha_{n,k}}{\alpha_{n,k}\sin\alpha_{n,k}} + \mathcal{O}\left(v_n^4\right). \end{aligned} \tag{2.23}$$

As expected, this result is identical to (2.13).

Now, the main novelty in using the Lagrange inversion theorem (2.22) is that arbitrary-order expansions can be derived. Though the manipulations are in principle quite simple, the expressions soon become extremely cumbersome to handle by hand. Fortunately, modern computer algebra software can be used instead. By these means, all terms up to 11th order in v_n have been computed, and it turns out that all odd-order terms in this range are equal to zero. Therefore, the result can be summarized as

$$\theta_{n,k} = \sum_{m=0}^{5} F_m\left(\alpha_{n,k}, \frac{\cos\alpha_{n,k}}{\sin\alpha_{n,k}}\right) v_n^{2m} + \mathcal{O}\left(v_n^{12}\right) \tag{2.24}$$

with the functions F_m explicitly known as

$$F_0(x,u) = x, \tag{2.25}$$

$$F_1(x,u) = \frac{1}{8}\frac{ux-1}{x}, \tag{2.26}$$

$$F_2(x,u) = \frac{1}{384}\frac{6x^2(1+u^2)+25-u(31u^2+33)x^3}{x^3}. \tag{2.27}$$

The functions F_3, F_4, and F_5 are also known explicitly, but these are listed in Appendix A because of their length.

The relative accuracy of (2.24) was tested in a multiple precision computing environment, using 100 decimal digits of accuracy. The following error was computed:

$$\Delta_M^\theta(n) = \sup_{k\in[1,\lfloor\frac{n}{2}\rfloor]}\left|\frac{\sum_{m=0}^{M} F_m\left(\alpha_{n,k}, \frac{\cos\alpha_{n,k}}{\sin\alpha_{n,k}}\right) v_n^{2m}}{\theta_{n,k}^{\text{Exact}}} - 1\right|, \tag{2.28}$$

where v_n brings in the dependence on n through (2.5). The results are plotted in Figure 1, along with the machine precision in the IEEE 754 double format, i.e., $\epsilon_{\text{mach}} = 2.2204 \times 10^{-16}$. As can be seen, the four-term approximation ($M = 3$) yields machine precision from $n \geq 62$. Therefore, if the values $\theta_{n,k}$ have been tabulated for $n \leq 100$, the two highest-order terms of (2.24) can be omitted. If a smaller look-up table is desired, adding the two highest-order terms can be advantageous because it increases the convergence rate considerably, as evidenced by the graph for $M = 5$, where machine precision is achieved for $n \geq 21$.

3. Expansion for the GL weights.

As is well known [6, 10], the weights $w_{n,k}$ of a GL quadrature rule can be directly computed from the nodes:

$$w_{n,k} = \frac{2}{\left[\frac{\mathrm{d}}{\mathrm{d}\theta}P_n(\cos\theta)\right]^2_{\theta=\theta_{n,k}}}. \tag{3.1}$$

It is clear that an expansion for the derived Legendre polynomial is sufficient to compute the weight with ease. To get such an expansion, the Taylor series (2.14) will be used to compute the derivative:

$$\frac{\mathrm{d}}{\mathrm{d}\theta}P_n(\cos\theta) = \frac{\mathrm{d}}{\mathrm{d}\theta}q(\theta)\sum_{p=0}^{\infty} f_p(\alpha_{n,k}, v_n)\frac{(\theta-\alpha_{n,k})^p}{p!} + q(\theta)\sum_{p=0}^{\infty} f_{p+1}(\alpha_{n,k}, v_n)\frac{(\theta-\alpha_{n,k})^p}{p!}. \tag{3.2}$$

Now $q(\theta)$ as well will be expanded in a Taylor series around $\alpha_{n,k}$:

$$q(\theta) = q(\alpha_{n,k})\sum_{r=0}^{\infty} Q_r(\alpha_{n,k})(\theta-\alpha_{n,k})^r \tag{3.3}$$

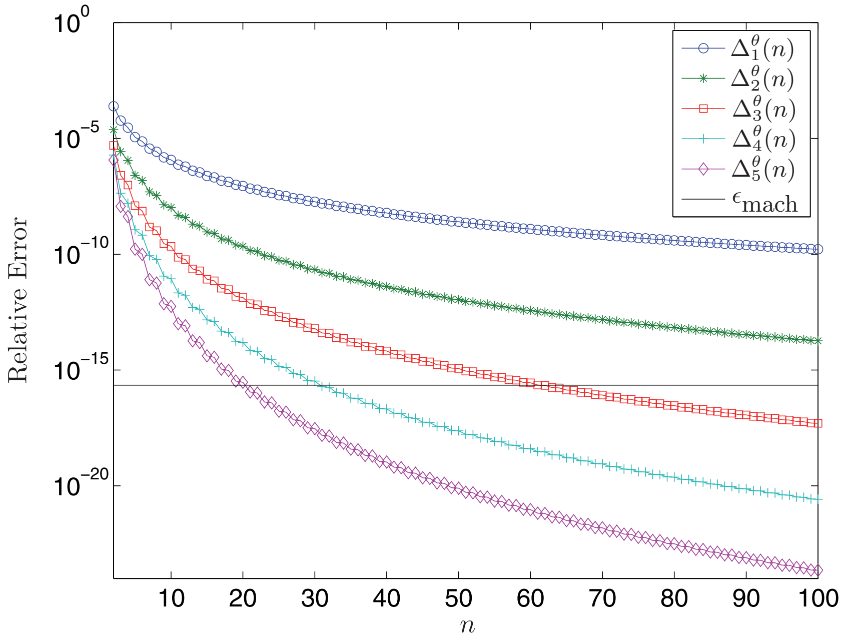

FIG. 1. *The convergence, as a function of the Legendre degree n, of the relative error on the GL nodes* (2.28) *for a varying number of terms M. The horizontal black line is the machine precision in the double format, i.e.,* 2.2204×10^{-16}.

with the first two terms given by

$$Q_0(\alpha_{n,k}) = 1, \tag{3.4}$$

$$Q_1(\alpha_{n,k}) = \frac{1}{2}\frac{\sin\alpha_{n,k} - \alpha_{n,k}\cos\alpha_{n,k}}{\alpha_{n,k}\sin\alpha_{n,k}}. \tag{3.5}$$

Again, higher-order terms are omitted for brevity but can be computed easily. Series (3.3) allows the immediate construction of the series for the derivative of $q(\theta)$:

$$\frac{\mathrm{d}}{\mathrm{d}\theta}q(\theta) = q(\alpha_{n,k})\sum_{r=0}^{\infty}(r+1)Q_{r+1}(\alpha_{n,k})(\theta - \alpha_{n,k})^r. \tag{3.6}$$

Substituting this into (3.2) yields

$$\begin{aligned}\frac{\mathrm{d}}{\mathrm{d}\theta}P_n(\cos\theta) = q(\alpha_{n,k})\sum_{r=0}^{\infty}\sum_{p=0}^{\infty}&[(r+1)Q_{r+1}(\alpha_{n,k})f_p(\alpha_{n,k}, v_n)\\ &+ Q_r(\alpha_{n,k})f_{p+1}(\alpha_{n,k}, v_n)]\,\frac{(\theta-\alpha_{n,k})^{r+p}}{p!}.\end{aligned} \tag{3.7}$$

Finally, θ has to be replaced with the series for the nodes (2.24). Not surprisingly, this again leads to very lengthy expressions that can in practice be handled only by means of computer algebra software. In addition, it is advantageous to directly compute an expansion for the square of (3.7) to avoid computing the square root in $q(\alpha_{n,k})$. This leads to even longer expressions. Nevertheless, these computations can all be carried out, leading to the following result:

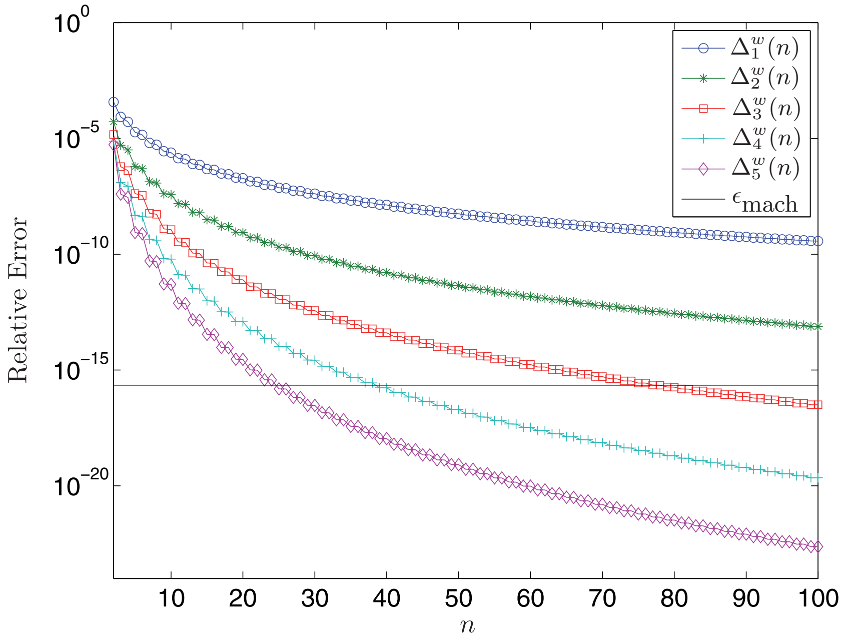

FIG. 2. *The convergence, as a function of the Legendre degree n, of the relative error on the GL weights* (3.12) *for a varying number of terms M. The horizontal black line is the machine precision in the double format.*

$$\frac{2}{w_{n,k}} = \frac{\mathcal{J}_k^2}{v_n^2}\frac{\alpha_{n,k}}{\sin\alpha_{n,k}}\left[\sum_{m=0}^{5} W_m\left(\alpha_{n,k}, \frac{\cos\alpha_{n,k}}{\sin\alpha_{n,k}}\right)v_n^{2m} + \mathcal{O}\left(v_n^{12}\right)\right] \tag{3.8}$$

with

$$W_0\left(x,u\right) = 1, \tag{3.9}$$

$$W_1\left(x,u\right) = \frac{1}{8}\frac{ux + x^2 - 1}{x^2}, \tag{3.10}$$

$$W_2\left(x,u\right) = \frac{1}{384}\frac{81 - 31ux - (3-6u^2)x^2 + 6ux^3 - (27+84u^2+56u^4)x^4}{x^4}. \tag{3.11}$$

Again, the functions W_3, W_4, and W_5 are also known but are listed in Appendix B because of their length.

In a way similar to the nodes, the relative accuracy of the weights (3.8) was tested using 100 decimal digits of accuracy. The following error was computed:

$$\Delta_M^w(n) = \sup_{k\in[1,\lfloor\frac{n}{2}\rfloor]}\left|\frac{\frac{\mathcal{J}_k^2}{v_n^2}\frac{\alpha_{n,k}}{\sin\alpha_{n,k}}\sum_{m=0}^{M} W_m\left(\alpha_{n,k}, \frac{\cos\alpha_{n,k}}{\sin\alpha_{n,k}}\right)v_n^{2m}}{w_{n,k}^{\text{Exact}}} - 1\right|. \tag{3.12}$$

Figure 2 shows the obtained convergence results. It can be seen that, using an identical M, the convergence for the weights is only slightly slower than for the nodes.

4. Auxiliary expansions.

To be able to use expansions (2.24) and (3.8), it is necessary to compute the zeros $j_{0,k}$ of the zeroth-order Bessel function. Also, $\mathcal{J}_k$, i.e., the first-order Bessel function evaluated in this zero, has to be computed. Asymptotic expansions for both these quantities will be given. Finally, it will also be explained how the functions F_m and W_m are computed in practice.

4.1. Computing $j_{0,k}$ and $\mathcal{J}_k$. For the computation of $j_{0,k}$, McMahon's asymptotic expansion for large zeros (equation (10.21.19) in [20]) is used:

$$j_{0,k} = a_k + \frac{1}{8a_k} - \frac{124}{3(8a_k)^3} + \frac{120928}{15(8a_k)^5} - \frac{401743168}{105(8a_k)^7} + \frac{1071187749376}{315(8a_k)^9} + \cdots \tag{4.1}$$

with $a_k = \pi(k - \frac{1}{4})$. Since (4.1) is an asymptotic expansion, it can be used only for sufficiently large k. For this paper, $j_{0,k}$ is tabulated for $k \in [1, 20]$ and computed by means of (4.1) if $k > 20$.

To compute $\mathcal{J}_k$, another asymptotic expansion is used, i.e., the large argument expansion for the Bessel function (equation (10.17.3) in [20]) itself:

$$J_1(x) = -\sqrt{\frac{2}{\pi x}}\left[\cos\left(x + \frac{\pi}{4}\right) - \frac{3}{8x}\cos\left(x - \frac{\pi}{4}\right) + \frac{15}{128x^2}\cos\left(x + \frac{\pi}{4}\right) + \cdots\right]. \tag{4.2}$$

Each separate term of this asymptotic expansion can be expanded into a Taylor series around the point $x = a_k$. For example, the first term leads to

$$\sqrt{\frac{2}{\pi x}}\cos\left(x + \frac{\pi}{4}\right) = \sqrt{\frac{2}{\pi a_k}}(-1)^k\left[1 - \frac{(x - a_k)}{2a_k} + (3 - 4a_k^2)\frac{(x - a_k)^2}{8a_k^2} + \cdots\right]. \tag{4.3}$$

The advantage of specifically choosing a_k as the expansion point is that all the trigonometric functions from (4.2) can be replaced with either zero or a power of minus one. Finally, McMahon's expansion for the zero (4.1) can be substituted in the Taylor series-expanded (4.2), yielding an expansion for $\mathcal{J}_k = J_1(j_{0,k})$:

$$\mathcal{J}_k = (-1)^{k+1}\sqrt{\frac{2}{\pi a_k}}\left[1 - \frac{7}{96a_k^4} + \frac{151}{320a_k^6} - \frac{230665}{43008a_k^8} + \frac{9239111}{92160a_k^{10}} + \cdots\right]. \tag{4.4}$$

Finally, it should be noted that only $\mathcal{J}_k^2$ appears in the expansion for the weights (3.8). Therefore, it is possible to avoid the square root and the factor $(-1)^{k+1}$ by immediately expanding $\mathcal{J}_k^2$:

$$\mathcal{J}_k^2 = \frac{1}{\pi a_k}\left[2 - \frac{7}{24a_k^4} + \frac{151}{80a_k^6} - \frac{172913}{8064a_k^8} + \frac{461797}{1152a_k^{10}} - \frac{171497088497}{15206400a_k^{12}} + \cdots\right]. \tag{4.5}$$

Using (4.1) and (4.5), it becomes possible to evaluate expansions (2.24) and (3.8) without the need for explicitly evaluating Bessel functions. Because evaluating Bessel functions is computationally expensive [21], this presents a significant advantage.

4.2. Computing F_m and W_m. The functions F_m and W_m occurring in expansions (2.24) and (3.8) can be problematic to evaluate numerically because of significant numerical cancellation occurring for $\alpha_{n,k}$ near zero. Therefore, special care must be taken to avoid this problem. In the following, the focus will be on the functions F_m, but W_m can be treated in a very similar way.

As can be seen from (2.25) through (2.27) and (A.2) through (A.4), the functions $F_m(\phi, \cot\phi)$ are analytic for $\phi \in\ ]-\pi, \pi[$. Because these functions need only be evaluated in the range $\phi \in [0, \frac{\pi}{2}]$, they are good candidates to be approximated by means of Chebyshev interpolants [14, 22]. However, the presence of a singularity at $\phi = r\pi \ \forall r \in \{\pm 1, \pm 2, \pm 3, \dots\}$ slows down the convergence of the Chebyshev series. This is why it was chosen to instead construct a Chebyshev series for the functions

$$G_m(\phi) = \frac{1}{\phi}\left[\frac{\sin\phi}{\phi}\right]^{2m-1} F_m(\phi, \cot\phi) \ \forall m \in \{1,2,3,4,5\}, \tag{4.6}$$

which are analytic on the entire complex plane. Additionally, it turns out that $G_m(\phi)$ is an even function of ϕ. Therefore, the interpolant was constructed for $G_m(\sqrt{\phi})$. Numerical experiments have subsequently shown that a Chebyshev interpolant of degree 6 is sufficient for obtaining approximately double precision on the nodes if $n > 100$. For the evaluation of W_m, an analogous approach leads to Chebyshev interpolants of degree 9 for $m = 1$ and 8 for $m > 1$.

5. Numerical results for double precision.

The results depicted in Figures 1 and 2 were generated using a floating point format with 100 digits of precision. Because such formats are not widely used in scientific computing, expansions (2.24) and (3.8) will also be tested when they are implemented in native double precision. The main difference with the 100-digits test is that expansions (2.24) and (3.8) are essentially exact in double precision. Therefore, the main source of error in the result is due to the effects of rounding error, the magnitude of which will now be investigated.

As mentioned, expansions (2.24) and (3.8) are useful only for n sufficiently large, such that tabulated values should be used for small n. Because the tables from [1] can be reused here, tabulation has been chosen for all $n \leq 100$. With this choice, $M = 3$ in (2.24) and (3.8) is sufficient for obtaining machine precision. It is worthwhile to point out that the expansions from this paper can accommodate a considerably smaller n. In fact, the convergence plots 1 and 2 indicate that $n > 30$ is large enough for machine precision if $M = 5$. Therefore, the size of the look-up table can be considerably reduced, though this incurs a cost for evaluating additional terms in the expansions. Alternatively, an increased number of terms can also be used to increase the accuracy of the expansion if the implementation is done in a more precise floating-point format. In such a case, the tabulation may be necessary for larger n.

The expansions for the nodes and weights have been implemented in C++. Subsequently, for $n \in [101, 500]$, the nodes $\theta_{n,k}$ and weights $w_{n,k}$ were computed $\forall k \in [1, \lfloor \frac{n}{2} \rfloor]$. For comparison, the same values were computed to 30-digit precision using Maple and rounded to the nearest double. This double value was henceforth considered the "best possible" double precision value. The error between the C++ and rounded Maple values was subsequently measured as the number of units in the last place (ulp; see [17]) that the two double values differ. Expressing the error in ulps is an excellent way to measure the relative error without referring explicitly to the machine precision. For example, 1 and $1 + \epsilon_{\text{mach}}$ differ by exactly one ulp.

Figures 3 and 4 graphically show the distribution of the errors on the nodes and weights for the range $n \in [101, 200]$. The dominance of the colors white and green means that most of the values are either exactly the best possible double precision value or differ by at most one ulp.

Tables 1 and 2 give more quantitative information on how often errors occur as a function of how large they are for the entire computed range $n \in [101, 500]$. As can be seen, all nodes and weights are correct up to at most 3 and 5 ulps, respectively, for this range. On average, weights contain around 0.8 ulp of error, while the nodes contain around 0.5 ulp of error, which is an excellent accuracy.

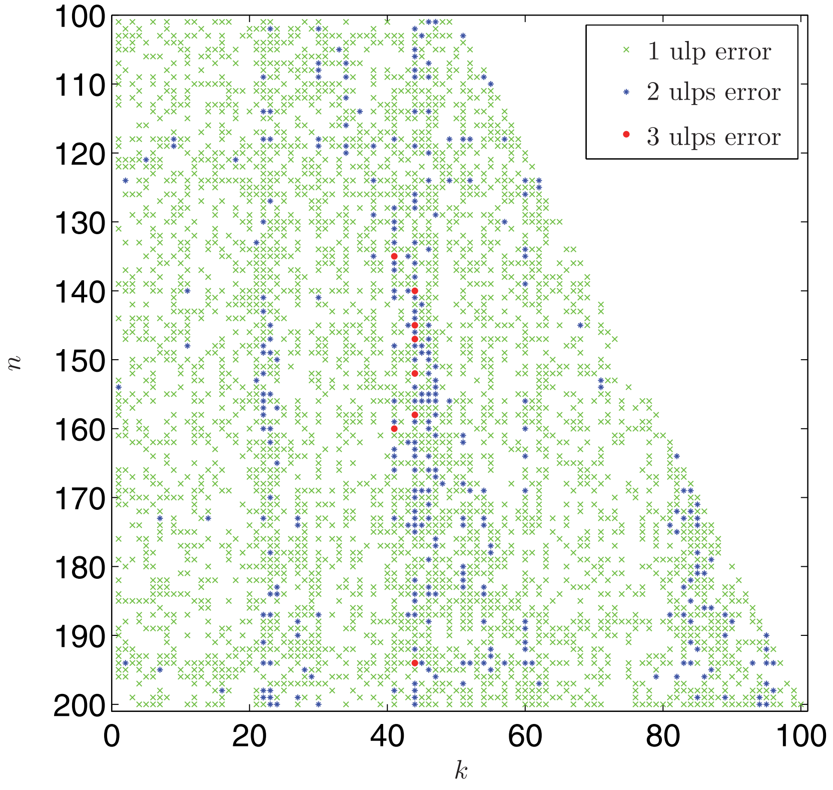

FIG. 3. *The error, measured in ulps, on the nodes $\theta_{n,k}$ in the range $n \in [101, 200], k \in [1, \lfloor \frac{n}{2} \rfloor]$. Nodes without error are not shown.*

In addition, the proposed expansions can also be implemented using the 80-bit extended precision format (sometimes called "long double") that is usually available on x86 processors. Tests for such an implementation have shown that (again for the range $n \in [101, 500]$) both nodes and weights differ from the Maple result by at most 1 ulp. However, this accuracy gain comes at the cost of a run time that is around 50 percent longer.

Finally, the speed of the C++ implementation has been tested. Table 3 lists the run times to compute all nodes and weights for $k \in [1, \lceil \frac{n}{2} \rceil]$ and n a power of 10. The computation was done on a laptop with an Intel Core i7-2630QM CPU@2GHz. Parallelization of the computation over the four CPU cores was done using either four or eight threads. When eight threads are used, hyperthreading allows some performance gains, but not as much as by adding four "real" CPU cores. As can be seen, computing a GL quadrature rule with 1 million points takes only a few tens of milliseconds, which is an order of magnitude faster than the iterative method from [1]. This speed comparison is justified because the exact same hardware, programming language, and compiler were used.

6. Expansion for the barycentric interpolation weights.

The barycentric interpolation formula [13], specialized for the case where the interpolation points are the GL nodes, is given by

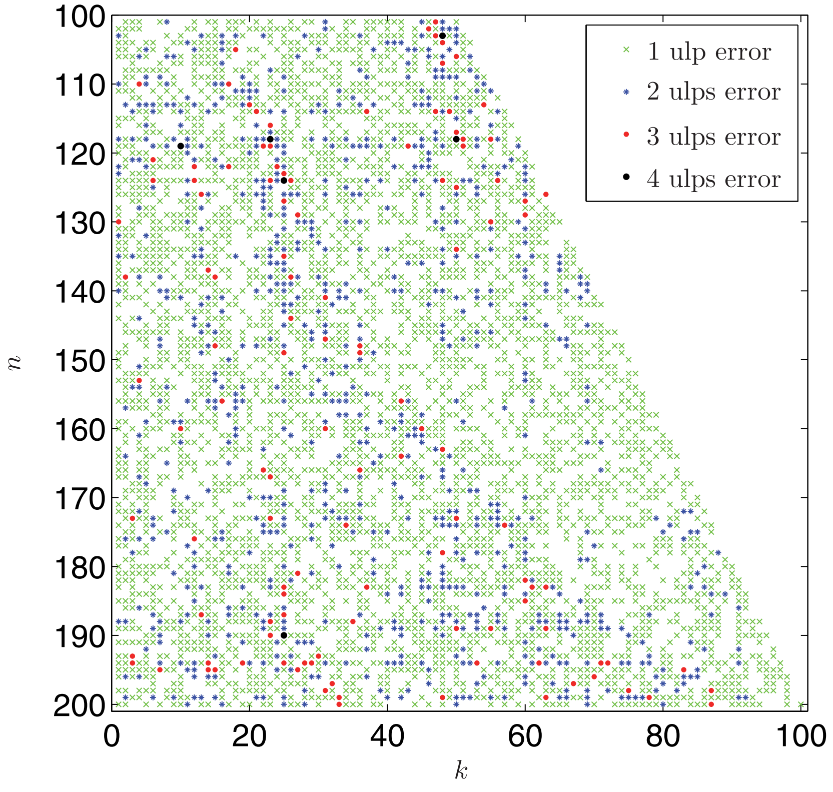

Fig. 4. *The error, measured in ulps, on the weights $w_{n,k}$ in the range $n \in [101, 200], k \in [1, \lfloor \frac{n}{2} \rfloor]$. Weights without error are not shown.*

Table 1

The number of times an N-ulp error was encountered for the nodes $\theta_{n,k}$ with $N \in \{0, 1, 2, 3\}$. No errors larger than three ulps were found.

n range	0 ulp	1 ulps	2 ulps	3 ulps
101 → 150	1727	1309	110	4
151 → 200	2430	1769	197	4
201 → 250	2850	2576	224	0
251 → 300	3585	3035	267	13
301 → 350	4417	3393	329	11
351 → 400	4667	4268	458	7
401 → 450	5229	4894	522	5
451 → 500	6200	5336	363	1

$$g(x) = \frac{\sum_{k=1}^{n} \frac{\lambda_{n,k} g(x_{n,k})}{x - x_{n,k}}}{\sum_{k=1}^{n} \frac{\lambda_{n,k}}{x - x_{n,k}}}, \tag{6.1}$$

where $g(x)$ is any polynomial of degree n (or lower) that needs to be interpolated. Each set of interpolation points has its own set of so-called barycentric weights. For the GL nodes, these will be denoted as $\lambda_{n,k}$.

The barycentric interpolation formula is of great practical importance because it exhibits very advantageous numerical stability properties [18, 13]. In addition, it

Table 2
The number of times an N-ulp error was encountered for the weights $w_{n,k}$ with $N \in \{0,1,2,3,4,5\}$. No errors larger than five ulps were found.

n range	0 ulp	1 ulp	2 ulps	3 ulps	4 ulps	5 ulps
$101 \to 150$	1237	1486	366	56	5	0
$151 \to 200$	1864	2056	416	63	1	0
$201 \to 250$	2053	2665	798	110	21	3
$251 \to 300$	2806	3257	738	90	8	1
$301 \to 350$	3616	3726	747	55	5	1
$351 \to 400$	3624	4368	1229	168	11	0
$401 \to 450$	3704	5060	1630	246	10	0
$451 \to 500$	4419	5687	1569	202	20	3

Table 3
The run time for computing an n-point GL quadrature rule, with large n, using parallelization over four CPU cores. The run times with eight threads are lower because of hyperthreading. The run times for the FastLegendre package [1] *are also displayed.*

	This work		FastLegendre [1]	
n	4 threads	8 threads	4 threads	8 threads
1000	< 0.01s	< 0.01s	< 0.01s	< 0.01s
10000	< 0.01s	< 0.01s	0.01s	< 0.01s
100000	< 0.01s	< 0.01s	0.04s	0.05s
1000000	0.02s	0.02s	0.33s	0.28s
10000000	0.18s	0.14s	2.31s	1.88s
100000000	1.68s	1.11s	21.4s	17.2s
1000000000	16.4s	11.0s	193s	157s

allows the interpolation process to be done with an $\mathcal{O}(n)$ computational complexity. Therefore, the practical importance of the barycentric interpolation formula warrants a discussion of how the techniques of this paper can be applied to compute the barycentric interpolation weights associated with the GL nodes.

It is clear that the barycentric weights $\lambda_{n,k}$ are only determined up to a constant factor. According to Theorem 3.1 in [23] or the more general theory in [9], this factor can be chosen such that the barycentric interpolation weights are given by

$$\lambda_{n,k} = \frac{1}{P_n'(x_{n,k})} = (-1)^{k+1}\sqrt{\frac{(1-x_{n,k}^2)w_{n,k}}{2}}, \tag{6.2}$$

$$= (-1)^{k+1}\sin\theta_{n,k}\sqrt{\frac{w_{n,k}}{2}}. \tag{6.3}$$

For increasing n, formula (6.2) becomes increasingly numerically unstable for points $x_{n,k}$ near the edges of the interval $[-1,1]$. However, the rewritten form (6.3) is numerically stable if it is used for $k \in [1, \lceil \frac{n}{2} \rceil]$ and the reflection property

$$\lambda_{n,k} = (-1)^{n+1}\lambda_{n,n-k+1} \tag{6.4}$$

is used for the remaining values of k. This method for computing $\lambda_{n,k}$ works, but it requires the evaluation of both the quadrature node and weight before the computation can be completed.

Alternatively, a series representation for $\lambda_{n,k}$ itself can be derived:

$$\lambda_{n,k} = \frac{v_n}{\mathcal{J}_k}\sqrt{\frac{\sin^3\alpha_{n,k}}{\alpha_{n,k}}}\left[\sum_{m=0}^{5}\Lambda_m\left(\alpha_{n,k}, \frac{\cos\alpha_{n,k}}{\sin\alpha_{n,k}}\right)v_n^{2m} + \mathcal{O}\left(v_n^{12}\right)\right] \tag{6.5}$$

with

$$\Lambda_0(x,u) = 1, \tag{6.6}$$

$$\Lambda_1(x,u) = \frac{3u^2x^2 - 3ux - (x^2-1)(u^2+1) - u^2}{16x^2}, \tag{6.7}$$

$$\Lambda_2(x,u) = \frac{44ux+4u^3x^3+22ux^3-4u^4x^4+7u^2x^2+4u^2x^4-8x^2+21x^4-51}{512x^4}. \tag{6.8}$$

The functions Λ_3, Λ_4, and Λ_5 are listed in Appendix C because of their length. To evaluate the factor $\mathcal{J}_k$ in (6.5), expansion (4.4) can be used, which nicely takes care of the sign changes in $\lambda_{n,k}$. Because the barycentric weights are only determined up to a factor, one could in principle remove the common factor v_n in expansion (6.5), thereby saving one multiplication for each barycentric weight.

To test the accuracy of expansion (6.5), the following error was computed:

$$\Delta_M^\lambda(n) = \sup_{k\in[1,\lfloor\frac{n}{2}\rfloor]}\left|\frac{\frac{v_n}{\mathcal{J}_k}\sqrt{\frac{\sin^3\alpha_{n,k}}{\alpha_{n,k}}}\sum_{m=0}^{M}\Lambda_m\left(\alpha_{n,k}, \frac{\cos\alpha_{n,k}}{\sin\alpha_{n,k}}\right)v_n^{2m}}{\lambda_{n,k}^{\text{Exact}}} - 1\right|. \tag{6.9}$$

Figure 5 shows a plot of this error, computed with 100 decimal digits of precision, as a function of n for various M. The convergence behavior is very similar to that of the

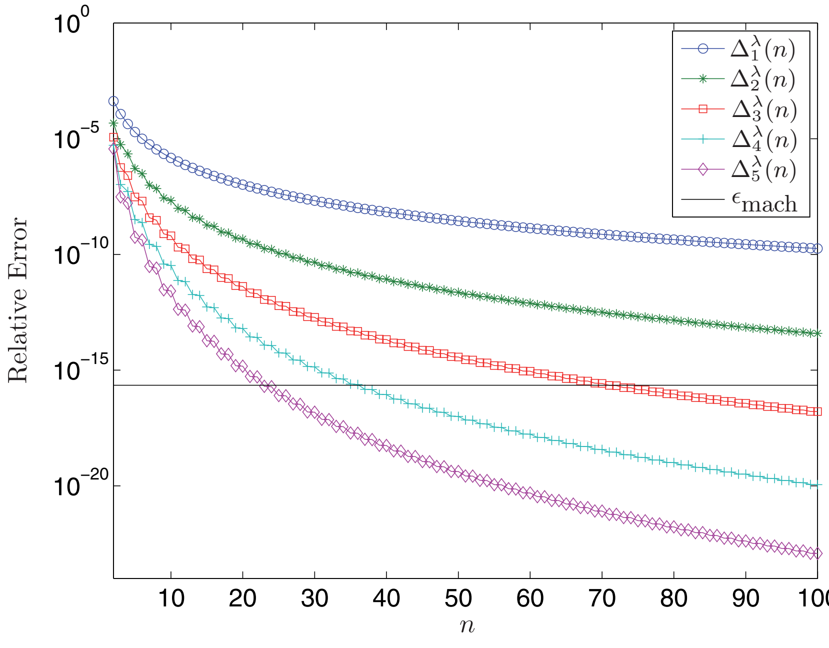

FIG. 5. *The convergence, as a function of the Legendre degree n, of the relative error on the barycentric interpolation weights* (6.9) *for a varying number of terms M. Again, the horizontal black line is the machine precision in the double format.*

GL nodes (2.24) and weights (3.8). Clearly, when n is large enough, expansion (6.5) allows the accurate evaluation of $\lambda_{n,k}$ without first calculating both $\theta_{n,k}$ and $w_{n,k}$. Therefore, it replaces two expansions with one, which may make it useful in certain speed-critical applications.

7. Conclusion. Expansions for the computation of individual Gauss–Legendre quadrature nodes and weights have been derived. These avoid both iterative zero-finding methods and the evaluation of Bessel functions. Because of this, the computation of Gauss–Legendre quadrature rules can be performed an order of magnitude faster than previously possible without sacrificing accuracy, as shown by the numerical results. Each Gauss–Legendre node-weight pair is computed independently from the others, which is required for parallelization purposes. In addition, the same expansion is used for all node-weight pairs, which is expected to be beneficial for the load balancing in parallel computing. Finally, an expansion is given for the barycentric interpolation weights associated with the Legendre nodes. A C++ implementation of the formulas from this paper is available online [12]. Future research may allow similar results for Gauss–Jacobi, Gauss–Hermite and Gauss–Laguerre quadrature rules.

Appendix A. Here, the functions $F_m(x,u)$ that appear in the expansion (2.24) will be given for $m \in \{3,4,5\}$. Since the expressions are too long to fit on one line they will be represented as follows:

$$F_m(x,u) = R_{m,0}(u) + R_{m,2m-1}(u)x^{-(2m-1)} + (1+u^2)\sum_{p=1}^{2m-3} R_{m,p}(u)x^{-p}. \tag{A.1}$$

The functions $R_{m,p}(u)$ are polynomials in u, given by

$$\begin{aligned} R_{3,0}(u) &= \frac{u(2595+6350u^2+3779u^4)}{15360},\ R_{3,1}(u) = -\frac{31u^2+11}{1024}, \\ R_{3,2}(u) &= \frac{u}{512},\ R_{3,3}(u) = -\frac{25}{3072},\ R_{3,5}(u) = -\frac{1073}{5120}, \end{aligned} \tag{A.2}$$

for $F_3(x,u)$. For $F_4(x,u)$, the following expressions hold:

$$\begin{aligned} R_{4,0}(u) &= -\frac{6277237u^6+14682157u^4+10808595u^2+2407755}{3440640}u, \\ R_{4,1}(u) &= \frac{3779u^4+3810u^2+519}{24576},\ R_{4,5}(u) = \frac{1073}{40960},\ R_{4,4}(u) = -\frac{25}{12288}u, \\ R_{4,3}(u) &= \frac{787u^2+279}{49152},\ R_{4,2}(u) = -\frac{21+31u^2}{4096}u,\ R_{4,7}(u) = \frac{375733}{229376}. \end{aligned} \tag{A.3}$$

Finally, for $F_5(x,u)$, the expressions are

$$\begin{aligned} R_{5,0}(u) &= LL\frac{u}{82575360}(6282767956u^6+415542645 \\ &\quad + 6710945598u^4+2092163573u^8+2935744980u^2), \\ R_{5,1}(u) &= -\frac{6277237u^6+10487255u^4+4632255u^2+343965}{3932160}, \\ R_{5,2}(u) &= +\frac{15178u^2+11337u^4+4329}{196608}u,\ R_{5,5}(u) = -\frac{100539u^2+35659}{1966080}, \\ R_{5,3}(u) &= -\frac{96335u^4+97122u^2+13227}{1179648},\ R_{5,4}(u) = \frac{778u^2+527}{98304}u, \\ R_{5,6}(u) &= \frac{41753}{5898240}u,\ R_{5,7}(u) = -\frac{375733}{1835008}, R_{5,9}(u) = -\frac{55384775}{2359296}. \end{aligned} \tag{A.4}$$

Appendix B. Here, the functions $W_m(x,u)$ that appear in the expansion of the weights (3.8) will be given for $m \in \{3,4,5\}$. These functions will be represented as

$$W_m(x,u) = \sum_{p=0}^{2m} Q_{m,p}(u) x^{-p}. \tag{B.1}$$

Again, the functions $Q_{m,p}(u)$ are polynomials in u. For $W_3(x,u)$, these are given by

$$\begin{gathered}
Q_{3,0}(u) = \frac{187}{96}u^4 + \frac{295}{256}u^2 + \frac{151}{160}u^6 + \frac{153}{1024}, \\
Q_{3,1}(u) = -\frac{119}{768}u^3 - \frac{35}{384}u^5 - \frac{65}{1024}u, \\
Q_{3,2}(u) = \frac{5}{512} + \frac{7}{384}u^4 + \frac{15}{512}u^2, \;\; Q_{3,3}(u) = \frac{1}{512}u^3 - \frac{13}{1536}u, \\
Q_{3,4}(u) = -\frac{7}{384}u^2 + \frac{53}{3072}, \; Q_{3,5}(u) = \frac{3749}{15360}u, \; Q_{3,6}(u) = -\frac{1125}{1024}.
\end{gathered} \tag{B.2}$$

For $W_4(x,u)$, the polynomials are

$$\begin{aligned}
Q_{4,0}(u) &= -\frac{21429}{32768} - \frac{27351}{1024}u^4 - \frac{3626248438009}{338228674560}u^8 - \frac{36941}{4096}u^2 - \frac{669667}{23040}u^6, \\
Q_{4,1}(u) &= \frac{8639}{6144}u^3 + \frac{2513}{8192}u + \frac{7393}{3840}u^5 + \frac{997510355}{1207959552}u^7, \\
Q_{4,2}(u) &= -\frac{1483}{8192}u^2 - \frac{1909}{6144}u^4 - \frac{1837891769}{12079595520}u^6 - \frac{371}{16384}, \\
Q_{4,3}(u) &= \frac{355532953}{6039797760}u^5 + \frac{1849}{18432}u^3 + \frac{675}{16384}u, \\
Q_{4,4}(u) &= -\frac{1183}{24576}u^2 - \frac{147456121}{4831838208}u^4 - \frac{1565}{98304}, \\
Q_{4,5}(u) &= -\frac{19906471}{6039797760}u^3 + \frac{6823}{245760}u, \;\; Q_{4,7}(u) = -\frac{76749336551}{42278584320}u, \\
Q_{4,6}(u) &= \frac{149694043}{2415919104}u^2 - \frac{156817}{1474560}, \;\; Q_{4,8}(u) = \frac{568995840001}{48318382080}.
\end{aligned} \tag{B.3}$$

Finally, for $W_5(x,u)$,

$$\begin{aligned}
Q_{5,0}(u) &= \frac{97620617026363819}{487049291366400}u^{10} + \frac{202966472595331}{300647710720}u^8 \\
&\quad + \frac{17266857}{20480}u^6 + \frac{22973795}{49152}u^4 + \frac{3401195}{32768}u^2 + \frac{1268343}{262144}, \\
Q_{5,1}(u) &= -\frac{65272472659909}{5411658792960}u^9 - \frac{2717368577869}{75161927680}u^7 \\
&\quad - \frac{4729993}{122880}u^5 - \frac{548439}{32768}u^3 - \frac{612485}{262144}u, \\
Q_{5,2}(u) &= \frac{26455}{262144} + \frac{6324614896949}{3607772528640}u^8 + \frac{45578037433}{9663676416}u^6 + \frac{52739}{12288}u^4 + \frac{93673}{65536}u^2, \\
Q_{5,3}(u) &= -\frac{181651}{196608}u^3 - \frac{40779010513}{32212254720}u^5 - \frac{63001776779}{115964116992}u^7 - \frac{19795}{98304}u, \\
Q_{5,4}(u) &= \frac{9477}{262144} + \frac{2101713105}{4294967296}u^4 + \frac{56281}{196608}u^2 + \frac{184730261873}{773094113280}u^6, \\
Q_{5,5}(u) &= -\frac{29273066033}{96636764160}u^3 - \frac{488659}{3932160}u - \frac{38212677741}{214748364800}u^5,
\end{aligned} \tag{B.4}$$

$$Q_{5,6}(u) = \frac{39817}{491520} + \frac{370339107271}{2319282339840}u^4 + \frac{996334037}{4026531840}u^2,$$
$$Q_{5,7}(u) = \frac{16514308061}{1352914698240}u^3 - \frac{3258170891}{15032385536}u,$$
$$Q_{5,8}(u) = \frac{3354565639447}{2705829396480} - \frac{335149450411}{721554505728}u^2,$$
$$Q_{5,9}(u) = \frac{1230657354291011}{48704929136640}u, \ Q_{5,10}(u) = -\frac{553063956480229}{2576980377600}.$$

Appendix C. Here, the functions $\Lambda_m(x,u)$ that appear in the expansion of the barycentric interpolation weights (6.5) will be given for $m \in \{3,4,5\}$. These functions will be represented as

$$\Lambda_m(x,u) = \sum_{p=0}^{2m} V_{m,p}(u)x^{-p}. \tag{C.1}$$

For $\Lambda_3(x,u)$, these are given by

$$\begin{aligned}
V_{3,0}(u) &= -\frac{3353}{6144}u^4 - \frac{671}{8192} - \frac{1663}{4096}u^2 - \frac{3329}{15360}u^6,\\
V_{3,1}(u) &= -\frac{5}{2048}u^5 - \frac{47}{8192}u - \frac{7}{2048}u^3,\\
V_{3,2}(u) &= \frac{1}{4096}u^4 + \frac{5}{8192}u^2 + \frac{43}{8192},\\
V_{3,3}(u) &= -\frac{227}{12288}u - \frac{85}{24576}u^3, \ V_{3,4}(u) = -\frac{149}{8192}u^2 + \frac{145}{8192},\\
V_{3,5}(u) &= -\frac{11861}{40960}u, \ V_{3,6}(u) = \frac{4343}{8192}.
\end{aligned} \tag{C.2}$$

For $\Lambda_4(x,u)$, the result is

$$\begin{aligned}
V_{4,0}(u) &= \frac{784535322707}{225485783040}u^8 + \frac{671835}{65536}u^4 + \frac{254427}{65536}u^2 + \frac{834761}{81920}u^6 + \frac{180323}{524288},\\
V_{4,1}(u) &= -\frac{163721}{491520}u^5 - \frac{11291}{131072}u - \frac{58637}{196608}u^3 - \frac{1472643103}{12079595520}u^7,\\
V_{4,2}(u) &= -\frac{340156231}{24159191040}u^6 - \frac{6919}{196608}u^4 - \frac{359}{65536} - \frac{6773}{262144}u^2,\\
V_{4,3}(u) &= \frac{4812749}{4026531840}u^5 + \frac{665}{393216}u^3 + \frac{81}{32768}u, \ V_{4,8}(u) = -\frac{557722275841}{96636764160},\\
V_{4,4}(u) &= -\frac{5251}{786432} + \frac{645241}{9663676416}u^4 - \frac{89}{98304}u^2, \ V_{4,7}(u) = \frac{196876210151}{84557168640}u,\\
V_{4,5}(u) &= \frac{143536039}{12079595520}u^3 + \frac{126871}{1966080}u, \ V_{4,6}(u) = -\frac{18107}{245760} + \frac{232305329}{2684354560}u^2.
\end{aligned} \tag{C.3}$$

Finally, for $\Lambda_5(x,u)$, the result is

(C.4)
$$\begin{aligned}
V_{5,0}(u) = &-\frac{72332722205864599}{974098582732800}u^{10} - \frac{2819116250606113}{10823317585920}u^8 - \frac{8039693539}{23592960}u^6\\
&- \frac{20898423}{8388608} - \frac{199700277}{4194304}u^2 - \frac{209379309}{1048576}u^4,
\end{aligned}$$

$$
\begin{aligned}
V_{5,1}(u) &= \frac{424956643859}{150323855360}u^9 + \frac{38943483325}{4227858432}u^7 + \frac{56864403}{5242880}u^5 + \frac{5629107}{1048576}u^3 + \frac{7600695}{8388608}u,\\
V_{5,2}(u) &= \frac{3800059265783}{21646635171840}u^8 + \frac{197398703969}{386547056640}u^6 + \frac{3200215}{6291456}u^4 + \frac{794451}{4194304}u^2 + \frac{135159}{8388608},\\
V_{5,3}(u) &= \frac{32340284483}{579820584960}u^7 + \frac{14729450399}{96636764160}u^5 + \frac{5152877}{37748736}u^3 + \frac{82767}{2097152}u,\\
V_{5,4}(u) &= \frac{3800093599}{171798691840}u^6 + \frac{8571856867}{154618822656}u^4 + \frac{171581}{4194304}u^2 + \frac{35673}{4194304},\\
V_{5,5}(u) &= -\frac{542701}{62914560}u - \frac{264058673}{48318382080}u^3 - \frac{3753744449}{966367641600}u^5,\\
V_{5,6}(u) &= -\frac{7512205357}{4638564679680}u^4 + \frac{354037729}{77309411328}u^2 + \frac{5843227}{188743680},\\
V_{5,7}(u) &= -\frac{356277323747}{676457349120}u - \frac{14479810427}{150323855360}u^3,\\
V_{5,8}(u) &= -\frac{19052843388127}{21646635171840}u^2 + \frac{7340675446247}{10823317585920},\\
V_{5,9}(u) &= -\frac{103175042361049}{3044058071040}u,\\
V_{5,10}(u) &= \frac{182195245670473}{1717986918400}.
\end{aligned}
$$

REFERENCES

[1] I. Bogaert, B. Michiels, and J. Fostier, *O(1) computation of Legendre polynomials and Gauss–Legendre nodes and weights for parallel computing*, SIAM J. Sci. Comput., 34 (2012), pp. 83–101.

[2] C. F. Gauss, *Methodus Nova Integralium Valores per Approximationem Inveniendi*, Gottingen, 1814.

[3] C. G. J. Jacobi, *Über Gauss neue Methode, die Werthe der Integrale näherungsweise zu finden*, J. Reine Angew. Math., 1 (1826), pp. 301–307.

[4] C. G. J. Jacobi, *Über 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.

[5] G. Szegö, *Über Einige Asymptotische Entwicklungen der Legendreschen Funktionen*, Proc. London Math. Soc., s2-36 (1934), pp. 427–450.

[6] G. Szegö, *Orthogonal Polynomials*, AMS, Providence, RI, 1978.

[7] 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.

[8] G. H. Golub and J. H. Welsch, *Calculation of Gauss quadrature rules*, Math. Comp., 23 (1969), pp. 221–230.

[9] H. Wang, D. Huybrechs, and S. Vandewalle, *Explicit Barycentric Weights for Polynomial Interpolation in the Roots or Extrema of Classical Orthogonal Polynomials*, arXiv:1202.0154, 2012.

[10] N. Hale and A. Townsend, *Fast and accurate computation of Gauss–Legendre and Gauss–Jacobi quadrature nodes and weights*, SIAM J. Sci. Comput., 35 (2013), pp. A652–A674.

[11] N. Hale and L. N. Trefethen, *Chebfun and numerical quadrature*, Sci. China Math., 55 (2012), pp. 1749–1760.

[12] I. Bogaert, *Fast Gauss-Legendre Quadrature Rules*, http://sourceforge.net/projects/fastgausslegendrequadrature.

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

[14] R. C. Li, *Near Optimality of Chebyshev interpolation for elementary function computations*, IEEE Trans. Comput., 53 (2004), pp. 678–687.

[15] B. Michiels, I. Bogaert, J. Fostier, and D. De Zutter, *A well-scaling parallel algorithm for the computation of the translation operator in the MLFMA*, IEEE Trans. Antennas Propag., 62 (2014), pp. 2679–2687.

[16] B. Michiels, I. Bogaert, J. Fostier, and D. De Zutter, *A weak scalability study of the parallel computation of the translation operator in the MLFMA*, in Proceedings of the International Conference on Electromagnetics in Advanced Applications, Turin, Italy, 2013.

[17] N. J. Higham, *Accuracy and Stability of Numerical Algorithms*, 2nd ed., SIAM, Philadelphia, 2002.

[18] N. J. Higham, *The numerical stability of barycentric Lagrange interpolation*, IMA J. Numer. Anal., 24 (2004), pp. 547–556.

[19] F. W. J. Olver, *Asymptotics and Special Functions*, Academic, New York, 1974.

[20] 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.

[21] R. Piessens, *Chebyshev series approximations for the zeros of the Bessel functions*, J. Comput. Phys., 53 (1984), pp. 188–192.

[22] L. N. Trefethen, *Computing numerically with functions instead of numbers*, Math. Comput. Sci., 1 (2007), pp. 9–19.

[23] H. Wang and S. Xiang, *On the Convergence Rates of Legendre Approximation*, Math. Comp., 81 (2012), pp. 861–877.

[24] W. C. Chew, J. Jin, E. Michielssen, and J. Song, *Fast and Efficient Algorithms in Computational Electromagnetics*, Artech House, London, 2001.