$\mathcal{O}(1)$ COMPUTATION OF LEGENDRE POLYNOMIALS AND GAUSS-LEGENDRE NODES AND WEIGHTS FOR PARALLEL COMPUTING[*]

I. BOGAERT[†], B. MICHIELS[†] AND J. FOSTIER[‡]

Abstract. A self-contained set of algorithms is proposed for the fast evaluation of Legendre polynomials of arbitrary degree and argument $\in [-1,1]$. More specifically the time required to evaluate any Legendre polynomial, regardless of argument and degree, is bounded by a constant, i.e. the complexity is $\mathcal{O}(1)$. The proposed algorithm also immediately yields an $\mathcal{O}(1)$ algorithm for computing an arbitrary Gauss-Legendre quadrature node. Such a capability is crucial for efficiently performing certain parallel computations with high order Legendre polynomials, such as computing an integral in parallel by means of Gauss-Legendre quadrature and the parallel evaluation of Legendre series. In order to achieve the $\mathcal{O}(1)$ complexity, novel efficient asymptotic expansions are derived and used alongside known results. A C++ implementation is available from the authors that includes the evaluation routines of the Legendre polynomials and Gauss-Legendre quadrature rules.

Key words. Legendre Polynomial, Gauss-Legendre Quadrature, Fixed Complexity, Parallel Computing

1. Introduction.

The Legendre polynomials are given by

$$P_l(x) = \frac{1}{2^l l!}\frac{\mathrm{d}^x}{\mathrm{d}l^x}(x^2-1)^l. \tag{1.1}$$

They are orthogonal with respect to a constant weight function, such that

$$\int_{-1}^{1} P_l(x) P_{l'}(x)\,\mathrm{d}x = \frac{2}{2l+1}\delta_{l,l'}. \tag{1.2}$$

Legendre polynomials are used in a great variety of numerical techniques and applications in physics. For example, Legendre polynomials are used in spectral methods to solve various kinds of differential equations [1]. In principle, one could use any polynomial basis for this. However, many of these alternative polynomial bases (such as the monomials) are insufficiently linearly independent to allow a numerically stable use in spectral methods.

In physics, Legendre polynomials naturally appear when the Laplace and Helmholtz equations are solved in spherical coordinates by means of separation of variables. For the same reason, they are used in the solution of the hydrogen atom in quantum mechanics. Because of their intimate link with the spherical coordinate system, it is not surprising that Legendre polynomials also feature prominently in the addition theorems of the Multilevel Fast Multipole Algorithm (MLFMA) [2, 3, 4]. Indeed, the translation operator in the MLFMA for high frequencies is given by

$$T(\gamma) = \sum_{l=0}^{L} a_l P_l(\cos\gamma). \tag{1.3}$$

[*]This work was supported by a postdoctoral grant from the Fund for Scientific Research Flanders (FWO-Vlaanderen).

[†]Ghent University, Dept. of Information Technology (INTEC), Sint-Pietersnieuwstraat 41, 9000 Gent, Belgium

[‡]Ghent University, Dept. of Information Technology (INTEC), Gaston Crommenlaan 8, Bus 201, 9000 Gent, Belgium

The exact nature of the coefficients a_l will not be detailed here. For more information, we refer to [2]. What is important in (1.3) is the fact that L can be huge. Sums of the type (1.3), where the coefficients a_l are not necessarily related to the MLFMA, will be called Legendre series throughout this paper.

Another application of Legendre polynomials is in the construction of Gauss-Legendre quadrature rules. Because of their orthogonality, the zeros of the Legendre polynomial of degree l constitute a set of l Gauss-Legendre nodes that, when supplemented with appropriate weights, forms a quadrature rule that integrates exactly all polynomials of degree up to $2l-1$. If one wants to compute the integral of a very oscillatory function, l can become very large. One such instance is again in the MLFMA, where highly oscillatory radiation patterns are integrated with the translation operator introduced earlier.

The key motivation for this paper is that up till now, fast algorithms for the evaluation of Legendre series or Gauss-Legendre quadrature rules are all inherently sequential. Indeed, when evaluating the Legendre series (1.3), one usually starts with $P_0\left(x\right)=1$, $P_1\left(x\right)=x$ and applies the recurrence

$$(n+1)P_{n+1}\left(x\right)=(2n+1)xP_n\left(x\right)-nP_{n-1}\left(x\right), \tag{1.4}$$

to eventually compute all the $P_l\left(x\right)$. The complexity of this computation is clearly $\mathcal{O}\left(L\right)$. However, since the complexity of the summation is also inherently $\mathcal{O}\left(L\right)$, this is not problematic. Recursive computation of the Legendre polynomials only becomes a problem when a parallel paradigm is adopted. Indeed, suppose the coefficients a_l are approximately evenly distributed over the P computational nodes (CNs). These CNs can for example be the different cores of a multi-core CPU or even the large number of cores in a cluster. Then each CN only needs to calculate $\frac{L}{P}$ Legendre polynomials to finish its share of work. However, for the CN containing a_L, the recursive computation of $P_L\left(x\right)$ requires $\mathcal{O}\left(L\right)$ operations, no matter how large P is. Therefore, increasing the number of CNs does not decrease the time required for the computation, i.e. the algorithm has a very poor parallel efficiency. It is clear that this problem is immediately resolved if an algorithm is available that evaluates $P_L\left(x\right)$ in $\mathcal{O}\left(1\right)$ time. In light of the recent shift towards parallelism in computer hardware, such an algorithm definitely has great value.

A similar argument holds for the computation of Gauss-Legendre quadrature rules. Arguably, one of the best algorithms to compute these rules was given in [5], where the differential equation of the Legendre polynomials is leveraged to compute a Gauss-Legendre node from the previous one. This process requires L steps to compute all the L Gauss-Legendre nodes, hence the complexity is $\mathcal{O}\left(L\right)$. In a computational environment consisting of a single CPU, this algorithm is optimal. However, its inherently sequential nature again prevents an efficient parallel implementation. In order to be able to efficiently compute Gauss-Legendre quadrature rules, one needs to be able to compute every node on its own. In [6, 7], an approach satisfying this requirement is proposed. First, an asymptotic formula for the nodes is used to obtain a coarse estimate. A few Newton-Raphson iterations are then used to get the node with machine precision. However, this procedure requires the repeated evaluation of a Legendre polynomial, which makes the complexity for computing any node $\mathcal{O}\left(L\right)$ if the recurrence (1.4) is used. Computing $\frac{L}{P}$ nodes in this fashion is therefore an $\mathcal{O}\left(\frac{L^2}{P}\right)$ process. This is again reduced to $\mathcal{O}\left(\frac{L}{P}\right)$ if any Legendre polynomial can be evaluated in constant time.

In this paper a collection of formulas will be given that, when patched together, allows the $\mathcal{O}(1)$ evaluation of Legendre polynomials of arbitrary degree. For low degrees, a direct polynomial evaluation based on the tabulated zeros of the Legendre polynomials is used. For high degrees, two distinct asymptotic formulas are used to cover the range $x \in [-1,1]$. All these formulas have been implemented in a C++ software package called FastLegendre that is available from the authors. We do not claim that FastLegendre is faster than existing methods for abitrary degree. Indeed, the algorithms in FastLegendre are sometimes considerably more complicated than for example the recurrence (1.4), which means that the constant hidden in the $\mathcal{O}(1)$ complexity can still be substantial. However, for sufficiently high degree, FastLegendre should be faster than existing methods. To ensure the applicability of this software package in parallel computing, much attention has been payed to the robustness and numerical stability of all these formulas, especially when the degree is extremely large. This is the subject of Section 2. One of the measures taken to improve numerical stability is to compute $P_l(\cos\theta)$ with θ as the double precision argument instead of $P_l(x)$ with x stored in double precision. In [7], it is intuitively explained why this is effective. In this paper, a numerical stability study is performed to provide heuristic error bounds for both $P_l(x)$ and $P_l(\cos\theta)$. This convincingly shows that the latter is indeed much better than the former, through the availability of better error bounds. In Section 3, the evaluation formulas to actually compute the Legendre polynomials are given. One of these formulas is a newly derived asymptotic expansion. In Section 4, techniques are developed for computing arbitrary Gauss-Legendre nodes and their associated weights. Finally, in Section 5, some numerical results are presented to demonstrate the accuracy and $\mathcal{O}(1)$ complexity of the algorithms. These results show that the accuracy of the Legendre polynomials remains bounded by the derived bounds for degrees as high as $2^{51} = 2251799813685248$. The numerical results also show that the Gauss-Legendre nodes are evaluated to machine precision for degrees as high as $2^{46} = 70368744177664$.

2. Error Control.

When evaluating the error-controllability of a computational scheme, it is natural to look at the relative accuracy of the obtained result versus some exactly known reference result. However, since the Legendre polynomials have many real zeros in the evaluation interval $x \in [-1,1]$, the relative error can be very large even though the absolute error is small. This is a problem that also occurs for example in the sin and cos functions in standard libraries and is very hard to avoid in fixed-precision floating-point arithmetic. Given that no easy solution to this problem exists, we will instead use an error measure based on Bernstein's inequality [8]

$$|P_l(x)| \leq \frac{2}{\sqrt{\pi(2l+1)\sqrt{1-x^2}}}. \tag{2.1}$$

The right hand side turns out to be a very good approximation for the envelope of the Legendre polynomials, and is a strictly positive real function. However, it has a singularity at $x = \pm 1$. This singularity can be avoided by using the well-known fact that $|P_l(x)| \leq 1$ and introducing a new upper bound $g_l(x)$

$$|P_l(x)| \leq g_l(x) = \min\left[1, \frac{2}{\sqrt{\pi(2l+1)\sqrt{1-x^2}}}\right]. \tag{2.2}$$

The switch between the two upper bounds occurs at

$$x = \pm x_l = \pm\sqrt{1 - \frac{16}{\pi^2(2l+1)^2}}. \tag{2.3}$$

In the rest of this paper, with a mild abuse of the term 'relative error', the absolute error on the numerically evaluated Legendre polynomials divided by $g_l(x)$ will be called the relative error.

Another aspect of floating-point arithmetic is the fact that the precision is unavoidably limited by the roundoff error on the input. Indeed, the floating-point representation $\mathrm{fl}(x)$ of the input x only limits the absolute error of the input to [9]

$$\delta = |\mathrm{fl}(x) - x| \leq |x|\,\epsilon, \tag{2.4}$$

where ϵ is the machine precision (for double precision $\epsilon = 2.2204 \times 10^{-16}$). Put in another way, the input is already contaminated with error as soon as it is passed as a floating-point argument of a function. In many circumstances this input error is negligible when compared to other error sources. However, in the case of the Legendre polynomials, this can lead to problems because the zeros of the Legendre polynomials are quadratically clustered around the points $x = \pm 1$. This means that the distance between the zeros of the Legendre polynomials nearest to $x = \pm 1$ is, in the limit of large l, proportional to $\frac{1}{l^2}$. Therefore, in the extreme case where $\frac{1}{l^2}$ becomes of the order of ϵ, the precision of the input is not even good enough to tell between which two zeros of the Legendre polynomial the input falls. Clearly, this precludes obtaining any precision on the output. If l is large but smaller than this extreme case, the output will be inaccurate. This problem is a more dramatic version of the problem encountered when trying to evaluate for example $\cos\left(10^{17}\sqrt{2}\right)$ by means of double precision floating-point operations. In Matlab, the result 0.611536661466470 is obtained, while the correct result is $-0.06981231488858282216...$ Since this paper is explicitly aimed at very-high-degree Legendre polynomials, this problem needs to be addressed. In [7], it was proposed to compute $P_l(\cos\theta)$ with $\theta \in [0, \pi]$ instead of computing $P_l(x)$ with $x \in [-1, 1]$. In the following, a mathematical treatment of the output error incurred by the input error will be given, and it will be shown that the substitution $x = \cos\theta$ allows the derivation of much stronger error bounds for the error on the Legendre polynomial.

2.1. Error Incurred by the Input. To estimate the influence of the input error on the value of the output, i.e. the Legendre polynomial, we will assume that the input error is small enough such that the Legendre polynomial can be approximated as a linear function in the neighborhood around the input argument. It is then easily shown that the absolute error is approximately

$$|P_l(\mathrm{fl}(x)) - P_l(x)| \approx \delta\,|P_l'(x)| \leq \epsilon\,|xP_l'(x)|, \tag{2.5}$$

where

$$P_l'(x) = \frac{\mathrm{d}}{\mathrm{d}x}P_l(x) = \frac{1}{2}(l+1)P_{l-1}^{1,1}(x). \tag{2.6}$$

Here $P_\nu^{\alpha,\beta}(x)$ denotes a Jacobi polynomial [10]. To bound the absolute error, it would be useful to have an upper bound like (2.2) for the Jacobi polynomial $P_{l-1}^{1,1}(x)$. However, as can be read in [8], upper bounds of this type are still the topic of mathematical research. Therefore an *approximate* upper bound for $P_{l-1}^{1,1}(x)$ is proposed in equation (A.5) in Appendix A. It is approximate in the sense that it is not a strict upper bound. However, it closely tracks the behavior of $P_{l-1}^{1,1}(x)$ and is more than good enough for our estimation purposes. Using this approximate upper bound, the relative error can be approximately bounded by

$$\frac{|P_l(\mathrm{fl}(x)) - P_l(x)|}{g_l(x)} \lessapprox \epsilon \frac{|x|}{\sqrt{1-x^2}} \frac{l(l+1)}{l+\frac{1}{2}}. \tag{2.7}$$

for all $-x_{l-1}^{1,1} \leq x \leq x_{l-1}^{1,1}$ with $x_{l-1}^{1,1}$ defined in Appendix A. The maximum error occurs when $x = \pm x_l^{1,1}$ and for large l becomes

$$\epsilon l(l+1)\frac{\sqrt[3]{\pi}}{2} \approx 0.7322959437\ \epsilon l(l+1). \tag{2.8}$$

The same maximum error is found for the range $[x_{l-1}^{1,1}, x_l]$, while for $[x_l, 1]$ the maximum error is $\frac{1}{2}\epsilon l(l+1)$. Apparently, when the argument x is stored as a floating-point number and is sufficiently close to ± 1, all precision is lost when ϵl^2 becomes of order unity. This is consistent with the reasoning made earlier about the clustering of the zeros of the Legendre polynomials.

The picture dramatically improves when $P_l(\cos\theta)$ is computed instead of $P_l(x)$. Repeating the analysis done for $P_l(x)$, the absolute error on the result becomes

$$|P_l(\cos \mathrm{fl}(\theta)) - P_l(\cos\theta)| \leq \epsilon\, |\theta \sin\theta P_l'(\cos\theta)|\,, \tag{2.9}$$

For $-x_{l-1}^{1,1} \leq \cos\theta \leq x_{l-1}^{1,1}$, the relative error becomes

$$\frac{|P_l(\cos \mathrm{fl}(\theta)) - P_l(\cos\theta)|}{g_l(\cos\theta)} \lessapprox \epsilon\, |\theta|\, \frac{l(l+1)}{l+\frac{1}{2}}, \tag{2.10}$$

which is $\epsilon\theta l$ for large l. The same maximum error is found for the range $[x_{l-1}^{1,1}, x_l]$, while for $[x_l, 1]$ the maximum error is

$$\epsilon\, |\theta|\, \frac{1}{\pi} \frac{l(l+1)}{l+\frac{1}{2}}. \tag{2.11}$$

Clearly, for all $\theta \in [0, \pi]$ and for large l, the relative error approximately satisfies

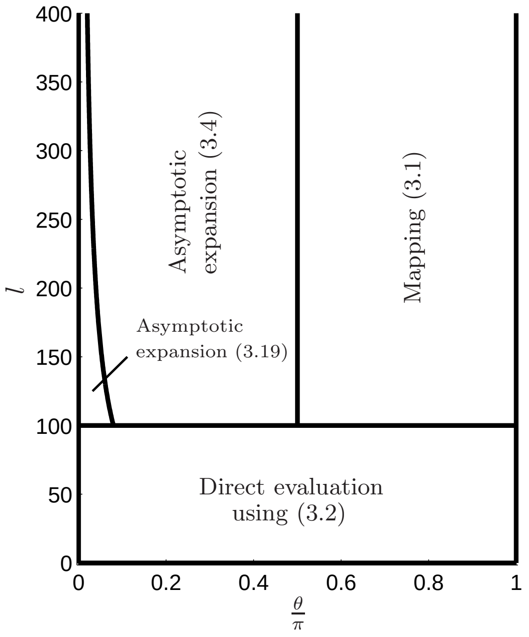
$$\frac{|P_l(\cos \mathrm{fl}(\theta)) - P_l(\cos\theta)|}{g_l(\cos\theta)} \lessapprox \epsilon\, |\theta|\, l. \tag{2.12}$$

This is a factor l better than if x is used as the argument, which can be huge if very high degrees are used. It should be pointed out that the algorithms to actually compute the Legendre polynomials should also be constructed such that they do not throw away this extra accuracy. In addition, they should keep other sources of errors (e.g. series truncation, etc.) in check. This will be handled in the next section.

3. Evaluation Formulas. In this section, four different formulas for the computation of $P_l(\cos\theta)$ will be given. The parameter ranges in which the four formulas are used are presented graphically in Figure 3.1. The first formula is a more or less direct evaluation of the polynomial and is used whenever $l \leq 100$. When l is larger than this bound, the functional equation

$$P_l(\cos\theta) = (-1)^l P_l(\cos(\pi - \theta))\,, \tag{3.1}$$

is used to map θ values in $[\frac{\pi}{2}, \pi]$ onto $[0, \frac{\pi}{2}]$. Therefore, the only remaining parameter range is $\theta \in [0, \frac{\pi}{2}]$ and $l > 100$. Two asymptotic expansions are leveraged to cover this range: the first is an asymptotic expansion found in [11] and is used when $(l+1)\sin\theta \geq 25$. The second is a novel asymptotic expansion, derived in Subsection 3.3. It is used when the first asymptotic expansion is not applicable. In the following, the various expansions are introduced and analyzed. In addition, criteria will be derived to determine which formula eventually gets used, depending on l and θ.



400
350
300
250
200
150
100
50
0
l
0
0.2
0.4
0.6
0.8
1
$\frac{\theta}{\pi}$
Asymptotic expansion (3.4)
Mapping (3.1)
Asymptotic expansion (3.19)
Direct evaluation using (3.2)

FIG. 3.1. *The patchwork of functions to evaluate $P_l(\cos\theta)$.*

3.1. Direct Evaluation for small l. For small l, there are a number of options for the evaluation of $P_l(\cos\theta)$. One option is using the recurrence relation (1.4). Another option is using the zeros of the Legendre polynomials, which leads to an evaluation method that leverages the following product representation

$$P_l(\cos\theta) = \begin{cases} C_l \prod_{n=1}^{\frac{l}{2}} \left(1 - \frac{\cos^2\theta}{x_{l,n}^2}\right) = \prod_{n=1}^{\frac{l}{2}} \left(1 - \frac{\sin^2\theta}{1-x_{l,n}^2}\right) & : l \text{ even} \\ C_l \cos\theta \prod_{n=1}^{\frac{l-1}{2}} \left(1 - \frac{\cos^2\theta}{x_{l,n}^2}\right) = \cos\theta \prod_{n=1}^{\frac{l-1}{2}} \left(1 - \frac{\sin^2\theta}{1-x_{l,n}^2}\right) & : l \text{ odd} \end{cases} \tag{3.2}$$

where $x_{l,n}$ is the nth positive zero of $P_l(x)$. The factor C_l is pre-computed using

$$C_l = \begin{cases} \frac{\sqrt{\pi}}{\Gamma\left(\frac{l}{2}+1\right)\Gamma\left(\frac{1-l}{2}\right)} & : l \text{ even} \\ lC_{l-1} & : l \text{ odd} \end{cases}, \tag{3.3}$$

and tabulated. Evaluating (3.2) using pre-computed and tabulated values for $\frac{1}{x_{l,n}^2}$ and $\frac{1}{1-x_{l,n}^2}$ requires fewer multiplications than the recurrence (1.4). In addition, it allows freedom in the sense that one can make the choice to use the product form based on the sines in certain circumstances, whereas the product form based on the cosines can be used in other circumstances. In the current version of FastLegendre, the sine-based form is used whenever $\left|\theta - \frac{\pi}{2}\right| \geq \frac{\pi}{4}$. The cosine-based form is used otherwise. This choice is motivated by the fact that the sine is more sensitive to the value of θ than the cosine in this region. Therefore, one can expect a higher accuracy in the final result. Another way to look at it is that using the cosine-based form is essentially the same as evaluating $P_l(x)$, i.e. the extra precision that was contained in the floating-point value of θ is lost by taking the cosine. Therefore, this evaluation method is best avoided when θ is near the boundaries of the range $[0, \pi]$.

3.2. Asymptotic Expansion for $(l+1)\sin\theta \geq 25$**.** Formula (8.21.5) in [12] is an asymptotic formula for the Legendre polynomials (this formula can also be found in [13] and [14]). After some manipulations, this formula can be rewritten as

$$P_l(\cos\theta) = \left(\frac{2}{\pi\sin\theta}\right)^{\frac{1}{2}} \sum_{m=0}^{M-1} C_{l,m} \frac{\cos\alpha_{l,m}(\theta)}{\sin^m\theta} + \xi_{l,M}(\theta), \tag{3.4}$$

with the residual $\xi_{l,M}(\theta)$ and

$$C_{l,m} = \frac{\Gamma^2\left(m+\frac{1}{2}\right)\Gamma(l+1)}{\pi 2^m \Gamma\left(l+m+\frac{3}{2}\right)\Gamma(m+1)}, \tag{3.5}$$

$$\alpha_{l,m}(\theta) = \left(l+m+\frac{1}{2}\right)\theta - \left(m+\frac{1}{2}\right)\frac{\pi}{2}. \tag{3.6}$$

The coefficients $C_{l,m}$ can be cheaply computed by means of

$$C_{l,m+1} = \frac{\left(m+\frac{1}{2}\right)^2}{2(m+1)\left(l+m+\frac{3}{2}\right)} C_{l,m}, \tag{3.7}$$

with starting value

$$C_{l,0} = \frac{\Gamma(l+1)}{\Gamma\left(l+\frac{3}{2}\right)}. \tag{3.8}$$

In Appendix B, a simple way to evaluate $C_{l,0}$ in $\mathcal{O}(1)$ operations is given, such that (3.4) can be evaluated in $\mathcal{O}(M)$ computations. To limit the number of computations to $\mathcal{O}(1)$, the parameter range in which (3.4) is used must be chosen such that the value of M is bounded by a constant M_0 whilst attaining accurate results. Here, this parameter range is defined by

$$(l+1)\sin\theta \geq R_0, \tag{3.9}$$

with R_0 a positive real constant. It will now be proven that the parameter R_0 can be chosen such that the error is controlled. To do this, the following upper bound on the magnitude of $\xi_{l,M}(\theta)$ will be used (see formula (8.21.6) in [12]):

$$|\xi_{l,M}(\theta)| < 2\left(\frac{2}{\pi\sin\theta}\right)^{\frac{1}{2}} \frac{C_{l,M}}{\sin^M\theta}. \tag{3.10}$$

Leveraging this upper bound, the relative error (as defined in Section 2) can be computed as

$$\frac{|\xi_{l,M}(\theta)|}{g_l(\cos\theta)} < \frac{C_{l,M}}{\sin^M\theta}\sqrt{4l+2} < (l+1)^M \frac{C_{l,M}}{R_0^M}\sqrt{4l+2}. \tag{3.11}$$

In the above, it was assumed that $R_0 > \frac{4}{\pi}$, such that $g_l(\cos\theta)$ is simply given by the upper bound in Bernstein's inequality (2.1). Substituting the explicit expression (3.5) for the coefficients $C_{l,M}$ yields

$$\frac{|\xi_{l,M}(\theta)|}{g_l(\cos\theta)} < (l+1)^M \frac{\Gamma^2\left(M+\frac{1}{2}\right)\Gamma(l+1)}{\pi 2^M \Gamma\left(l+M+\frac{3}{2}\right)\Gamma(M+1)} \frac{1}{R_0^M}\sqrt{4l+2}. \tag{3.12}$$

Using inequality (C.1) from Appendix C, the following upper bound on the relative error is easily obtained

$$\frac{|\xi_{l,M}(\theta)|}{g_l(\cos\theta)} < \frac{2}{\pi}\frac{\Gamma^2\left(M+\frac{1}{2}\right)}{2^M\Gamma(M+1)}\frac{1}{R_0^M} = \Delta_1(M, R_0). \tag{3.13}$$

The right hand side of (3.13), seen as a function of M, first converges and then starts to diverge when M becomes larger than approximately $2R_0$. Therefore, to get an accurate result from the asymptotic series (3.4), the minimum of the curve must simply reach below the wanted precision ϵ. Figure 3.2 shows the behavior of the curve for some choices of R_0. As can be seen, R_0 must be at least 17.3645 for double precision. In order to have fast convergence, it makes sense to choose a somewhat larger R_0 than this lower bound. In FastLegendre, the choice $R_0 = 25$ is made for this reason. In addition, this choice ensures that the asymptotic expansion derived in the next section also converges using only a limited number of terms. Given the curve in Figure 3.2 for $R_0 = 25$, it is clear that no more than around 17 terms are required to get double precision accuracy using (3.4). Since each term only requires $\mathcal{O}(1)$ work, the total complexity of evaluating (3.4) is also $\mathcal{O}(1)$.

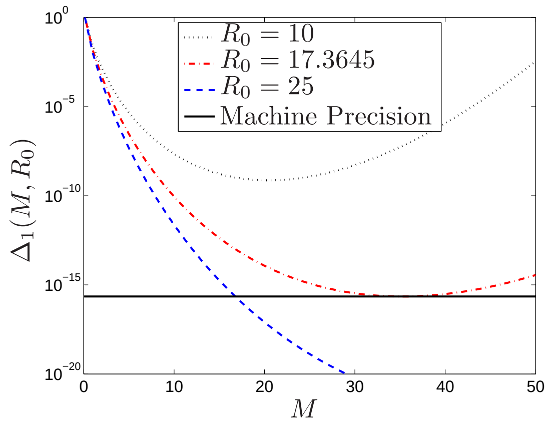


FIG. 3.2. *The convergence and subsequent divergence of* $\Delta_1(M, R_0)$ *as a function of* m. *To obtain sufficient precision with this expansion,* $\Delta_1(M, R_0)$ *must at some point become smaller than the machine precision. This requirement leads to the condition* $R_0 > 17.3645$.

3.3. Asymptotic Expansion for $(l+1)\sin\theta < 25$**.** Inequality (3.9) gives a sufficient condition for applicability of (3.4) in the evaluation of $P_l(\cos\theta)$. If this condition is not satisfied, another asymptotic expansion is required. In [15], an asymptotic expansion is given that converges uniformly for $0 \leq \theta \leq 2\pi(\sqrt{2}-1) - \chi_0$ with $0 < \chi_0 < 2\pi(\sqrt{2}-1)$. However, the terms in this expansion exhibit problematic numerical behavior when θ is close to zero, which is the case we are interested in.

Therefore, a novel expansion will be derived to handle this. Like the asymptotic expansion in [15], the novel expansion is a generalization of the well-known Mehler-Heine formula [16]

$$\lim_{l\to\infty} P_l\left(\cos\frac{z}{l}\right) = J_0\left(z\right). \tag{3.14}$$

This basically means that

$$P_l\left(\cos\theta\right) \approx J_0\left(l\theta\right), \tag{3.15}$$

when θ becomes small enough and l large enough.

Although approximation (3.15) is only accurate for very small values of $l\theta$, it is possible to find more accurate generalizations. To derive these, we will start from the Legendre differential equation

$$\sin\theta\frac{\mathrm{d}^2}{\mathrm{d}\theta^2}P_l\left(\cos\theta\right) + \cos\theta\frac{\mathrm{d}}{\mathrm{d}\theta}P_l\left(\cos\theta\right) + \sin\theta l(l+1)P_l\left(\cos\theta\right) = 0. \tag{3.16}$$

It will become clear later on that it is advantageous to do the substitution $l = v - \frac{1}{2}$, such that

$$l(l+1) = v^2 - \frac{1}{4}. \tag{3.17}$$

After this substitution, the Legendre differential equation only depends on the square of v. Subsequently, the substitution $\theta = \frac{y}{v}$ is done, which results in

$$\begin{aligned} v^2\sin\left(\frac{y}{v}\right)\frac{\mathrm{d}^2}{\mathrm{d}y^2}P_{v-\frac{1}{2}}\left(\cos\frac{y}{v}\right) + v\cos\left(\frac{y}{v}\right)\frac{\mathrm{d}}{\mathrm{d}y}P_{v-\frac{1}{2}}\left(\cos\frac{y}{v}\right) \\ + \sin\left(\frac{y}{v}\right)\left(v^2 - \frac{1}{4}\right)P_{v-\frac{1}{2}}\left(\cos\frac{y}{v}\right) = 0. \end{aligned} \tag{3.18}$$

Now, in order to find an asymptotic expansion for small θ, the magnitude of y will be understood to be bounded by a constant y_0. A solution of the form

$$P_{v-\frac{1}{2}}\left(\cos\frac{y}{v}\right) = \sum_{n=0}^{N-1}\frac{f_n(y)}{v^n} + \mathcal{O}\left(v^{-N}\right), \tag{3.19}$$

is proposed. Because y is bounded, the following Taylor expansions converge very quickly

$$\sin\left(\frac{y}{v}\right) = \sum_{p=0}^{P-1}\frac{(-1)^p}{(2p+1)!}\left(\frac{y}{v}\right)^{2p+1} + \mathcal{O}\left(v^{-2P-1}\right), \tag{3.20}$$

$$\cos\left(\frac{y}{v}\right) = \sum_{p=0}^{P-1}\frac{(-1)^p}{(2p)!}\left(\frac{y}{v}\right)^{2p} + \mathcal{O}\left(v^{-2P}\right). \tag{3.21}$$

When (3.19), (3.20) and (3.21) are substituted into (3.18), the left hand side can be written as a polynomial in v^{-1}. Making this polynomial zero means making all of its coefficients zero. It is easily proved that this leads to a set of differential equations given by

$$\mathcal{B}\left[f_n(y)\right] = \sum_{j=1}^{\left\lfloor\frac{n}{2}\right\rfloor}(-1)^{j+1}\mathcal{U}_j\left[f_{n-2j}(y)\right], \tag{3.22}$$

with the operators $\mathcal{B}$ and $\mathcal{U}_n$ defined as

$$\mathcal{B}\left[g(y)\right] = y\frac{\mathrm{d}^2}{\mathrm{d}y^2}g(y) + \frac{\mathrm{d}}{\mathrm{d}y}g(y) + yg(y), \tag{3.23}$$

$$\mathcal{U}_n\left[g(y)\right] = \frac{y^{2n+1}}{(2n+1)!}\left[\frac{\mathrm{d}^2}{\mathrm{d}y^2}g(y) + g(y)\right] + \frac{y^{2n}}{(2n)!}\frac{\mathrm{d}}{\mathrm{d}y}g(y) + \frac{y^{2n-1}}{4(2n-1)!}g(y). \tag{3.24}$$

The operator $\mathcal{B}$ can also be interpreted as $\mathcal{U}_0$ since $(-1)! = \Gamma(0) = \infty$. Equations (3.22) can be recursively solved. To do this, note that $y\mathcal{B}$ is the operator occurring in Bessel's differential equation with the index equal to zero [11]. Therefore, all the equations in (3.22) have the homogeneous solution

$$\text{Homogeneous}\left[f_n(y)\right] = c_n J_0(y) + d_n Y_0(y). \tag{3.25}$$

For $n \in [0,1]$ the particular solution is zero. In this case, the Weber function $Y_0(y)$ can be ruled out to represent the smooth Legendre polynomials, since it has a logarithmic singularity for $y = 0$. Note that the Weber function would be required if an asymptotic expansion of Legendre functions of the second kind was sought (see for example equation (14.15.12) in [11]). The coefficient c_0 can be determined to be 1 because of the fact that for $\theta = y = 0$

$$P_l(\cos 0) = 1, \forall l \in \mathbb{R}. \tag{3.26}$$

Therefore $f_0(0) = 1$, hence $c_0 = 1$ and

$$f_0(y) = J_0(y). \tag{3.27}$$

Note that this is compatible with the Mehler-Heine formula (3.14). A similar reasoning shows that $c_1 = 0$. As a consequence

$$f_1(y) = 0. \tag{3.28}$$

This shows that the differential equation for $f_3(y)$ is homogeneous which, using (3.26), leads to $f_3(y) = 0$. This reasoning can be repeated such that

$$f_{2n+1}(y) = 0, \ \forall n. \tag{3.29}$$

This result can be traced back to the substitution $l = v - \frac{1}{2}$ that was done earlier. Indeed, it can be checked that the functions $f_{2n+1}(y)$ are nonzero if the substitution is not done, i.e. if an asymptotic expansion in l is attempted instead of in v.

The functions with even index are more complicated because of the presence of the particular solution. However, it is still possible to compute them analytically. Up to twelfth order, these functions have been computed as

$$f_2(y) = \tfrac{1}{8}h_1(y) - \tfrac{1}{12}h_2(y), \tag{3.30a}$$

$$f_4(y) = \tfrac{11}{384}h_2(y) - \tfrac{7}{160}h_3(y) + \tfrac{1}{160}h_4(y), \tag{3.30b}$$

$$f_6(y) = \tfrac{173h_3(y)}{15360} - \tfrac{101h_4(y)}{3584} + \tfrac{671h_5(y)}{80640} - \tfrac{61h_6(y)}{120960}, \tag{3.30c}$$

$$f_8(y) = \tfrac{22931 h_4(y)}{3440640} - \tfrac{90497 h_5(y)}{3870720} + \tfrac{217 h_6(y)}{20480} - \tfrac{1261 h_7(y)}{967680} + \tfrac{1261 h_8(y)}{29030400}, \tag{3.30d}$$

$$\begin{aligned} f_{10}(y) = {} & \tfrac{1319183 h_5(y)}{247726080} - \tfrac{10918993 h_6(y)}{454164480} + \tfrac{1676287 h_7(y)}{113541120} - \tfrac{7034857 h_8(y)}{2554675200} \\ & + \tfrac{1501 h_9(y)}{8110080} - \tfrac{79 h_{10}(y)}{20275200}, \end{aligned} \tag{3.30e}$$

$$\begin{aligned} f_{12}(y) = {} & \tfrac{233526463 h_6(y)}{43599790080} - \tfrac{1396004969 h_7(y)}{47233105920} + \tfrac{2323237523 h_8(y)}{101213798400} - \tfrac{72836747 h_9(y)}{12651724800} \\ & + \tfrac{3135577 h_{10}(y)}{5367398400} - \tfrac{1532789 h_{11}(y)}{61993451520} + \tfrac{66643 h_{12}(y)}{185980354560}, \end{aligned} \tag{3.30f}$$

with

$$h_n(y) = y^n J_n(y), \ \forall n. \tag{3.31}$$

Using equations (3.30) in (3.19) with $N = 14$ yields the desired asymptotic expansion for small θ. This expansion will be used whenever $l > 100$ and $(l+1)\sin\theta < R_0$. This means that

$$y = v\theta = \left(l + \frac{1}{2}\right)\theta < \left(l + \frac{1}{2}\right) \arcsin\left(\frac{R_0}{l+1}\right). \tag{3.32}$$

In the limit of high l, the right hand side goes to the constant R_0. For finite l, it is slightly bigger than this limit and it attains its maximum for the minimal l, i.e. 101. Therefore $y < y_0 = 25.1336$ whenever (3.19) is used. To show that this expansion yields sufficiently accurate results, the difference

$$\Delta_2(v, y) = \left| P_{v-\frac{1}{2}}\left(\cos\frac{y}{v}\right) - \sum_{n=0}^{6} \frac{f_{2n}(y)}{v^{2n}} \right|, \tag{3.33}$$

was computed using Maple, with 120 digits of precision. It is plotted for various values of v in Figure 3.3 and for $y \in [0, y_0]$. Half-integer values of v are chosen because these correspond to integer values of l. In the figure, it can be clearly seen that the error is significantly below the machine precision for the entire y-range, which means that expansion (3.19) is sufficiently accurate. Also, this expansion becomes more and more accurate as l increases.

4. Computation of Quadrature Rules. The evaluation routines of the Legendre polynomials can be used to construct Gauss-Legendre quadrature rules. Suppose we want to construct a Gauss-Legendre quadrature rule consisting of l point-weight pairs $\{x_{l,k}, w_{l,k}\}$, $\forall k \in [0, l-1]$. Such a rule exactly integrates any polynomial of degree $2l - 1$ [11]. It is well-known that the points $x_{l,k}$ satisfy

$$P_l(x_{l,k}) = 0, \tag{4.1}$$

i.e. they are the l zeros of the Legendre polynomial of degree l. The weights associated with these points are given by (see 15.3.1 [12])

$$w_{l,k} = \frac{2(1 - x^2)}{\left[(n+1)P_{l+1}(x_{l,k})\right]^2}. \tag{4.2}$$

Clearly, once a zero is known, computing the associated weight is a computation with $\mathcal{O}(1)$ complexity if the Legendre polynomial is evaluated using the strategy from

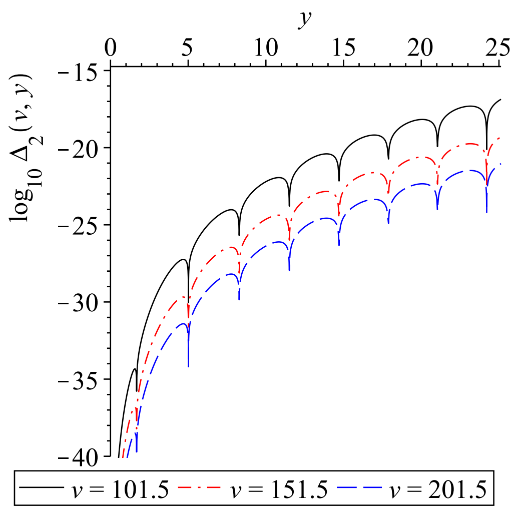


FIG. 3.3. *The error between the exact Legendre polynomial of degree $v - \frac{1}{2}$ and the asymptotic expansion (3.19).*

Section 3. Therefore, the only challenge remaining is the computation of $x_{l,k}$. In FastLegendre, this is done by first computing the $\theta_{l,k}$-values for which $P_l(\cos\theta_{l,k}) = 0$. Then, optionally, the cosine is taken to compute $x_{l,k}$. For $l \leq 100$, the zeros $\theta_{l,k}$ are tabulated. For $l > 100$, initial values

$$\theta_{l,k} \approx \pi \frac{4k+3}{4l+2}, \tag{4.3}$$

are polished with a few Newton-Raphson iterations to obtain the true value of $\theta_{l,k}$ with near machine precision. This approach is similar to the approaches from [6, 7] and requires only the evaluation of the Legendre polynomial and its derivative. Because the derivative of the Legendre polynomial is easily reduced to another evaluation of a Legendre polynomial

$$(1-x^2)P_l'(x) = lP_{l-1}(x) - lxP_l(x), \tag{4.4}$$

the computation of any single point-weight pair $\{x_{l,k}, w_{l,k}\}$ is an operation with $\mathcal{O}(1)$ complexity.

Due to the simple form of the initial values (4.3), a remarkable trick can be played which allows the computation of $\theta_{l,k}$ with an accuracy which is, in some ways, better than the machine precision. To show how this trick works, assume that $(l+1)\sin\theta > 25$. Then the evaluation will be done using asymptotic expansion (3.4). If we now define

$$\Delta\theta = \theta - \pi\frac{4k+3}{4l+2}, \tag{4.5}$$

then (3.4) can be rewritten into

$$P_l(\cos\theta) \approx (-1)^{k+1}\left(\frac{2}{\pi\sin\theta}\right)^{\frac{1}{2}} \sum_{m=0}^{M-1} C_{l,m} \frac{\sin\left[\left(l+\frac{1}{2}\right)\Delta\theta + m\left(\theta - \frac{\pi}{2}\right)\right]}{\sin^m\theta}. \tag{4.6}$$

It can be seen that the argument of the sine in the numerator inside the summation is $\mathcal{O}(1)$ if $\Delta\theta$ remains small enough, which is the case if one is searching for the Legendre zero $\theta_{l,k}$. Therefore, the Legendre polynomial will be evaluated with a precision close to the machine precision in this neighborhood. If we denote $\Delta\theta_{l,k} = \theta_{l,k} - \pi\frac{4k+3}{4l+2}$, this means that $\Delta\theta_{l,k}$, which is a small $\mathcal{O}\left(\frac{1}{l}\right)$ correction to the usually much larger $\pi\frac{4k+3}{4l+2}$, will be computed with a precision close to the machine precision. The relative error on this partly symbolical representation of $\theta_{l,k}$ is therefore

$$\frac{|\Delta\theta_{l,k}|\,\epsilon}{\pi\frac{4k+3}{4l+2} + \Delta\theta_{l,k}} \approx \mathcal{O}\left(\frac{\epsilon}{k}\right), \tag{4.7}$$

which can be significantly better that ϵ. Of course, from the moment that $\theta_{l,k}$ is actually computed by summing the correction $\Delta\theta_{l,k}$ with $\pi\frac{4k+3}{4l+2}$ and storing it in a double precision floating-point number, the extra accuracy is thrown away. However, all the complicated operations (i.e. the evaluation of the asymptotic expansion and the Newton-Raphson iteration), are finished by the time this summation is done. The fact that this summation is a very simple operation gives confidence that the computed $\theta_{l,k}$ will be very close to the best possible floating-point approximation.

5. Numerical Results. First the accuracy of the Legendre polynomials is investigated. Three values for θ were selected: $\frac{\pi}{3}$, 10^{-9} and $\pi - 10^{-9}$. The value $\frac{\pi}{3}$ demonstrates the accuracy of asymptotic expansion (3.4), while the value 10^{-9} is aimed at probing the accuracy of (3.19). Finally, the value $\pi - 10^{-9}$ is meant to show the influence of mapping (3.1) on the accuracy. Using these values as arguments, the relative error on $P_{2^p}(\cos\theta)$ was computed for $p \in [0, 51]$. Note that the highest degree Legendre polynomial that was checked, has the extremely high degree $2^{51} = 2251799813685248$. The exact values of the Legendre polynomial were computed in Maple using 50 digits of accuracy and then converted to double precision floating-point numbers for comparison with the values from FastLegendre. The results can be seen in Figure 5.1. The relative error on the Legendre polynomials was computed as proposed in Section 2, i.e. as in the left hand side of (2.12). The black curves in Figure 5.1 are the approximate upper bounds, given in the right hand side of (2.12), for $\theta = \frac{\pi}{3}$ and $\theta = 10^{-9}$. As can be seen, the relative error on the Legendre polynomials for these two θ values nicely stays below the corresponding error bound, except when the precision is bounded by the machine precision. For $\theta = \pi - 10^{-9}$, a similar error bound could also be plotted. However, this was omitted because the relative error for this θ value is already below the upper bound for $\theta = \frac{\pi}{3}$, hence it is surely below the bound for $\theta = \pi - 10^{-9}$. A curious phenomenon in the curve for $\theta = \pi - 10^{-9}$ is that, for low p, it remains close to machine precision and then increases quadratically until it hits its upper bound. This is easily explained by realizing that the Legendre polynomial is approximately

$$P_l(\cos\theta) = 1 - \frac{1}{4}l(l+1)\theta^2 + \mathcal{O}\left(\theta^4\right). \tag{5.1}$$

for $\theta << \frac{4}{l+\frac{1}{2}}$. When the argument is $\theta = \pi - 10^{-9}$, mapping (3.1) is used to reduce the argument to 10^{-9}. However, this reduced value is contaminated with $\mathcal{O}(\pi\epsilon)$ roundoff error. Therefore, the error on the Legendre polynomial can be estimated as

$$\left(1 - \frac{1}{4}l(l+1)\theta^2\right) - \left(1 - \frac{1}{4}l(l+1)(\theta+\pi\epsilon)^2\right) \approx \frac{1}{2}l(l+1)\pi\epsilon\theta \tag{5.2}$$

This error is smaller than ϵ whenever

$$\theta < \frac{2}{\pi l(l+1)}, \tag{5.3}$$

which matches with the results shown in Figure 5.1. When $\frac{4}{l+\frac{1}{2}} > \theta > \frac{2}{\pi l(l+1)}$, the error is larger than ϵ and rises quadratically. When θ is larger than the upper bound of this range, the error estimate (5.2) is no longer valid because the Taylor series (5.1) requires more terms to capture the behavior of the Legendre polynomial. Summarizing, Figure 5.1 shows that the behavior of the relative error can be completely described by reasonings based only on the error on the input, and the fact that the output precision is limited to machine precision. This can be interpreted as clear evidence that the algorithms used in FastLegendre do not add significant error to the error incurred by the rounding of the input θ, i.e. it is not possible to substantially improve the accuracy by improving the algorithms. The total loss of precision for degree 2^{51} cannot be solved without improving the error on the input.

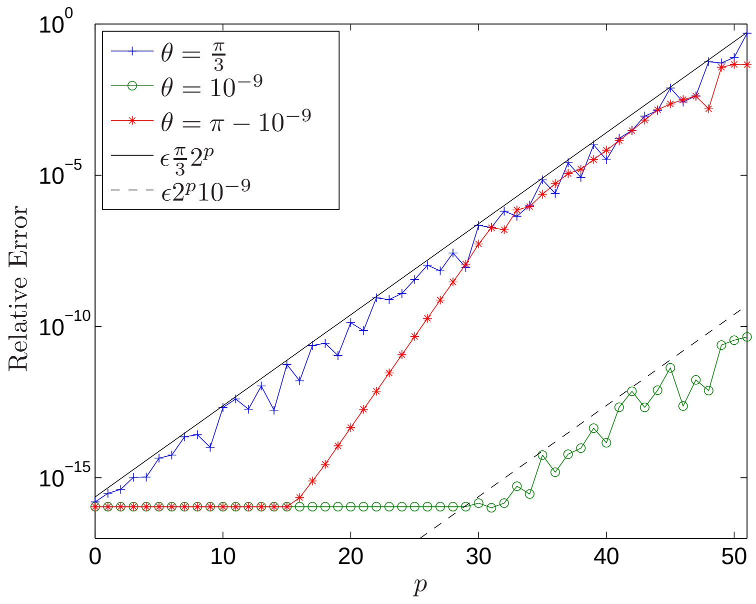


Fig. 5.1. *The relative error on $P_{2^p}(\cos\theta)$, evaluated using FastLegendre, for $\theta \in \{\frac{\pi}{3}, 10^{-9}, \pi - 10^{-9}\}$. The black curves are the approximate upper bounds for the relative error caused by the error on the input. Clearly, the error only increases proportionally with l. Also, the only circumstance in which the actual error is larger than the approximate upper bound is when the precision is bounded from below by the machine precision. Also note that the degree of the tested Legendre polynomial reaches the extremely high value* 2251799813685248 *when $p = 51$.*

Secondly, the accuracy on the Gauss-Legendre quadrature nodes will be investigated. To do this, the correction $\Delta\theta_{l,k}$ will be computed for $l = 2^p, \forall p \in [2, 51]$, and $k \in \left\{0, \frac{l}{4}\right\}$. The zero with $k = 0$ is always in the parameter range in which (3.19) is used. The $\frac{l}{4}$th zero is computed using the trick described in Section 4. The correction was also computed to high precision using Maple. The relative error between the two results is shown in Figure 5.2. Clearly, for the case $k = \frac{l}{4}$, the relative error on the correction $\Delta\theta_{l,k}$ is $\mathcal{O}(\epsilon)$. This allows us to immediately conclude that the error on $\theta_{l,k}$ will also be $\mathcal{O}(\epsilon)$. When $k = 0$, the error starts increasing when $p > 46$ or

$l > 2^{46} = 70368744177664$. This increase can be traced back to the fact that equation (4.4) starts to become numerically unstable for very high l. In principle, this could be solved by writing dedicated routines for computing the derivative of the Legendre polynomial. However, at this time, quadrature rules with $l > 2^{46}$ are not exactly commonly used, which is why such a solution has not been implemented.

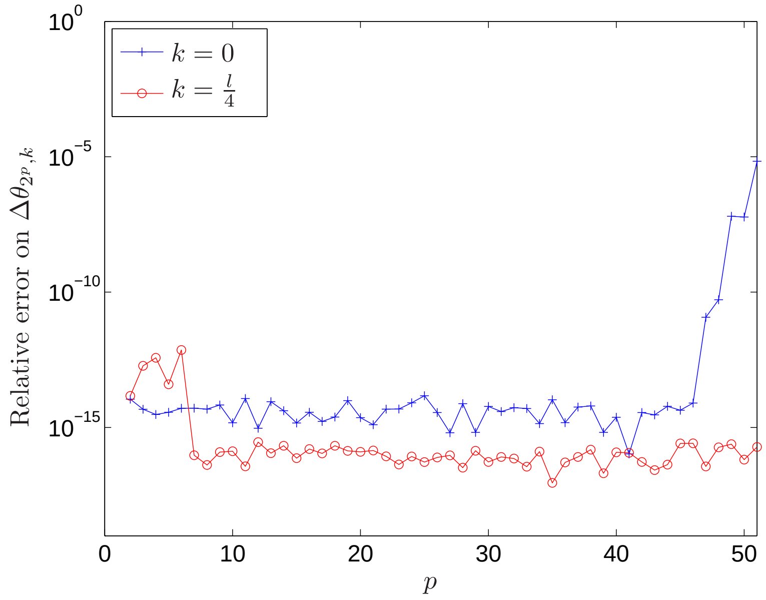


FIG. 5.2. *The relative error on the correction $\Delta\theta_{2^p,k}$ for $k \in \left\{0, \frac{l}{4}\right\}$.*

Finally, the Legendre polynomials and Gauss-Legendre quadrature rules are tested together by applying a quadrature rule with r points to the Legendre polynomial of degree $\frac{3}{2}r$

$$I_r = \int_{-1}^{1} P_{\frac{3}{2}r}(x)\,\mathrm{d}x = \sum_{k=0}^{r} w_{r,k} P_{\frac{3}{2}r}(\cos\theta_{r,k}) = 0. \tag{5.4}$$

In the test cases, only even r will be involved, such that $\frac{3}{2}r$ is always an integer. When no numerical error is present, the result of both the integral and the discrete summation is exactly zero, due to the orthogonality of the Legendre polynomials. However, in the finite precision case, any error on the Gauss-Legendre nodes or weights will result in a nonzero value for I_r, which makes the computation a suitable test for the accuracy of both the Legendre polynomials and the Gauss-Legendre quadrature rule. This test is exactly the same as the one used in [5] to test the accuracy of the quadrature rule. Table 5.1 shows the computed results for $r = 10^p\ \forall p \in [0, 9]$. Clearly, the integration is done very accurately and appears to be even more accurate than the results in [5]. The fact that I_r drops below the machine precision can be attributed to the fact that the magnitude of the Legendre polynomials in (5.4) drops with increasing r. This effect can be removed by multiplying I_r with the normalization factor $\sqrt{\frac{2r+1}{2}}$ (see equation (1.2)). The third column of Table 5.1 shows the result of this multiplication. The time required to compute the Gauss-Legendre nodes and do

r	$\lvert I_r \rvert$	$\lvert I_r \rvert \sqrt{\frac{2r+1}{2}}$	Computation time
10	$2.296 \ \ 10^{-17}$	$7.441 \ \ 10^{-17}$	< 0.01s
100	$2.206 \ \ 10^{-16}$	$2.211 \ \ 10^{-15}$	< 0.01s
1000	$2.502 \ \ 10^{-16}$	$7.916 \ \ 10^{-15}$	< 0.01s
10000	$1.356 \ \ 10^{-16}$	$1.356 \ \ 10^{-14}$	0.02s
100000	$8.107 \ \ 10^{-17}$	$2.563 \ \ 10^{-14}$	0.04s
1000000	$6.142 \ \ 10^{-17}$	$6.142 \ \ 10^{-14}$	0.39s
10000000	$1.481 \ \ 10^{-17}$	$4.684 \ \ 10^{-14}$	2.37s
100000000	$4.502 \ \ 10^{-18}$	$4.502 \ \ 10^{-14}$	22.08s
1000000000	$5.687 \ \ 10^{-18}$	$1.798 \ \ 10^{-13}$	200.41s

Table 5.1

The computed values for I_r and the time required for computing the Gauss-Legendre quadrature rule and performing the integration.

the integration is also given and shows that the complexity for computing a Legendre polynomial and Gauss-Legendre node is $\mathcal{O}(1)$. The computation was done on a laptop with an Intel(R) Core(TM) i7-2630QM CPU@2GHz. The availability of four cores with hyperthreading allowed the computation to be parallelized using simple multi-threading. Eight threads were used for the data in Table 5.1. This allowed a Gauss-Legendre quadrature rule with one billion points and the corresponding test integral I_r to be computed in under four minutes.

6. Conclusion.

A patchwork of formulas has been proposed to allow the evaluation of the Legendre polynomial $P_l(\cos\theta)\,\forall l \in \mathbb{N}, \forall \theta \in [0,\pi]$ in $\mathcal{O}(1)$ complexity. The argument $\cos\theta$ is chosen to improve the numerical stability of the evaluation. One of these formulas is a novel asymptotic expansion, which is derived in this paper. Rigorous or heuristic error bounds have been derived for every formula in this patchwork, and these bounds have been confirmed by numerical results comparing a C++ implementation with high-precision Maple results. This comparison was done for degrees as high as $2^{51} = 2251799813685248$. The capability to compute $P_l(\cos\theta)$ in $\mathcal{O}(1)$ complexity also yields an algorithm for computing an arbitrary Gauss-Legendre quadrature node (and its associated weight) in $\mathcal{O}(1)$ complexity. In addition, a technique has been devised to compute a Gauss-Legendre quadrature node as a small correction to a large analytically known quantity. Remarkably, it is possible to obtain this correction with a precision close to the machine precision. This algorithm has also been implemented and numerically validated. The fact that the algorithms presented in this paper are suitable for parallelization has allowed the computation of a Gauss-Legendre quadrature rule with one billion points in under four minutes on modest hardware. The C++ implementation, which includes the evaluation routines of the Legendre polynomials and Gauss-Legendre quadrature rules, is available from the authors.

Appendix A.

Here, an approximate upper bound of $P_n^{1,1}(x)$ will be proposed. It is inspired by an upper bound for Jacobi polynomials given in [8]

$$\left|P_n^{\alpha,\beta}(\cos\theta)\right| \leq \frac{\Gamma(q+1)\binom{n+q}{n}\left[n+\frac{\alpha+\beta+1}{2}\right]^{-q-\frac{1}{2}}}{\sqrt{\pi}\sin^{\alpha+\frac{1}{2}}\left(\frac{\theta}{2}\right)\cos^{\beta+\frac{1}{2}}\left(\frac{\theta}{2}\right)}, \tag{A.1}$$

where $q = \max(\alpha, \beta)$. Unfortunately, this inequality is only valid for $|\alpha| \le \frac{1}{2}$ and $|\beta| \le \frac{1}{2}$, which is not the case here: $\alpha = 1$ and $\beta = 1$. However, numerical experiments indicate that the right hand side of (A.1) still provides a very good approximation of the envelope of $P_n^{1,1}(x)$. Therefore, we propose the approximate upper bound

$$\left|P_n^{1,1}(\cos\theta)\right| \lessapprox \frac{n+1}{\sqrt{\pi}} \left[\frac{2}{\left(n+\frac{3}{2}\right)\sin\theta}\right]^{\frac{3}{2}}. \tag{A.2}$$

whenever $-x_n^{1,1} \le \cos\theta \le x_n^{1,1}$ with

$$x_n^{1,1} = \sqrt{1 - \left[\frac{2}{\left(n+\frac{3}{2}\right)\sqrt[3]{\pi}}\right]^2}. \tag{A.3}$$

Outside of this range, the easily proved upper bound

$$\left|P_n^{1,1}(\cos\theta)\right| \le n+1, \tag{A.4}$$

is used to avoid the singularity. Summarizing these results, the following approximate upper bound is obtained

$$\left|P_n^{1,1}(\cos\theta)\right| \lessapprox \min\left[n+1, \frac{n+1}{\sqrt{\pi}} \left[\frac{2}{\left(n+\frac{3}{2}\right)\sin\theta}\right]^{\frac{3}{2}}\right]. \tag{A.5}$$

Figure A.1 shows the magnitude of the Jacobi polynomial $P_{27}^{1,1}(\cos\theta)$ and its associated approximate upper bound. Clearly, equation (A.5) provides a very good description of the global behavior of the Jacobi polynomial.

Appendix B. Here, a method to compute

$$C_{n,0} = \frac{\Gamma(n+1)}{\Gamma\left(n+\frac{3}{2}\right)}, \tag{B.1}$$

with an $\mathcal{O}(1)$ complexity will be given. First introduce an auxiliary function $\tau(x)$ as

$$\tau(x) = \sqrt{x}C_{x-\frac{3}{4},0} = \sqrt{x}\frac{\Gamma\left(x+\frac{1}{4}\right)}{\Gamma\left(x+\frac{3}{4}\right)}. \tag{B.2}$$

Numerical experiments show that this auxiliary function converges to 1 when $x \to \infty$, i.e. it does not go to infinity. It is therefore reasonable to attempt to construct an asymptotic series for $\tau(x)$ by computing the Taylor series of $\tau\left(\frac{1}{t}\right)$ around $t = 0$. This procedure has been performed with the symbolic mathematics package Maple, and it turns out that all the computed Taylor series coefficients are finite. The final result is

$$\begin{aligned}\tau(x) \approx 1 - \frac{1}{64x^2} + \frac{21}{8192x^4} - \frac{671}{524288x^6} + \frac{180323}{134217728x^8} \\ - \frac{20898423}{8589934592x^{10}} + \frac{7426362705}{1099511627776x^{12}} + \mathcal{O}\left(\frac{1}{x^{14}}\right).\end{aligned} \tag{B.3}$$

Note that only the even powers of x occur, which is nice from a computational point of view.

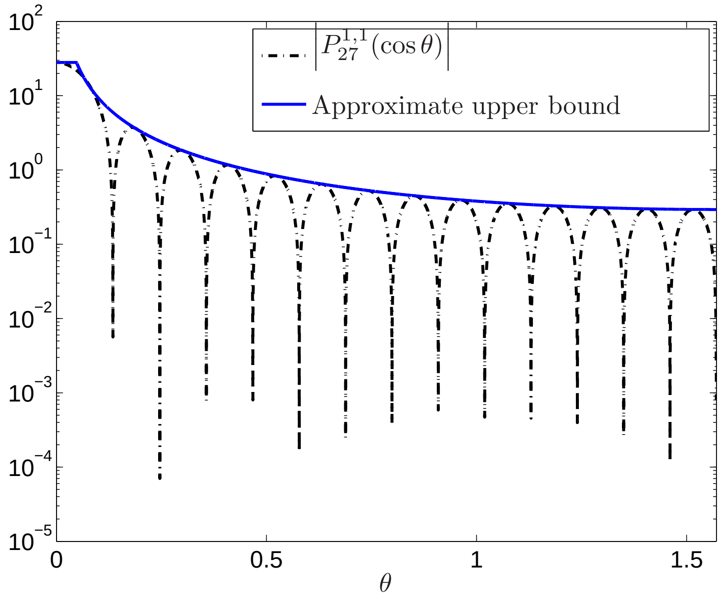


FIG. A.1. *The magnitude of the Jacobi polynomial $P_{27}^{1,1}(\cos\theta)$, and its approximate upper bound, as proposed in equation (A.5). Only the range $[0, \frac{\pi}{2}]$ is shown because the rest of the range is just a mirrored version of this plot.*

It is clear that, since equation (B.3) is a series around $x = \infty$, (B.3) will only be accurate for sufficiently large x. Numerical tests show that for all $x > 10.09475$, the relative error on $\tau(x)$ is below the double precision accuracy $\epsilon = 2.2204 \times 10^{-16}$. This means that $C_{n,0}$ can be evaluated as

$$C_{n,0} = \frac{1}{\sqrt{n+\frac{3}{4}}}\tau\left(n+\frac{3}{4}\right), \tag{B.4}$$

whenever $n > 9$. Since n is an integer, the values of $C_{n,0}$ for $n \in [0, 9]$ can be simply tabulated.

Appendix C. The following inequality will be proved

$$(l+1)^{M}\frac{\Gamma(l+1)}{\Gamma\left(l+M+\frac{3}{2}\right)}\sqrt{4l+2} < 2, \tag{C.1}$$

for $l, M \in \mathbb{N}$. It is convenient to start with proving (C.1) for $M = 0$. Indeed, by using Gautschi's Inequality (formula (5.6.4) from [11]) with $x = l + \frac{1}{2}$ and $s = \frac{1}{2}$, it is easily shown that

$$\sqrt{l+\frac{1}{2}} < \frac{\Gamma\left(l+\frac{1}{2}+1\right)}{\Gamma\left(l+\frac{1}{2}+\frac{1}{2}\right)}. \tag{C.2}$$

From this, it immediately follows that

$$\frac{\Gamma(l+1)}{\Gamma\left(l+\frac{3}{2}\right)}\sqrt{4l+2} < 2. \tag{C.3}$$

Also, by means of the functional equation of the Gamma function, it is easily shown that

$$\frac{(l+1)^M}{\Gamma\left(l+M+\frac{3}{2}\right)} = \frac{1}{\Gamma\left(l+\frac{3}{2}\right)} \cdot \frac{l+1}{l+\frac{3}{2}} \cdot \frac{l+1}{l+\frac{5}{2}} \cdot \ \cdots \ \cdot \frac{l+1}{l+M+\frac{3}{2}} < \frac{1}{\Gamma\left(l+\frac{3}{2}\right)}. \tag{C.4}$$

Using inequalities (C.4) and (C.3), inequality (C.1) is proved:

$$(l+1)^M \frac{\Gamma(l+1)}{\Gamma\left(l+M+\frac{3}{2}\right)} \sqrt{4l+2} < \frac{\Gamma(l+1)}{\Gamma\left(l+\frac{3}{2}\right)} \sqrt{4l+2} < 2. \tag{C.5}$$

REFERENCES

[1] C. Canuto, M. Hussaini, A. Quarteroni, and T. Zang, *Spectral methods: fundamentals in single domains.* Springer, 2006.

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

[3] I. Bogaert, J. Peeters, and F. Olyslager, "A nondirective plane wave MLFMA stable at low frequencies," *IEEE Transactions on Antennas and Propagation*, vol. 56, no. 12, pp. 3752–3767, December 2008.

[4] I. Bogaert and F. Olyslager, "A low frequency stable plane wave addition theorem," *Journal of Computational Physics*, vol. 228, no. 4, pp. 1000–1016, March 2009.

[5] A. Glaser, X. Liu, and V. Rokhlin, "A fast algorithm for the calculation of the roots of special functions," *SIAM Journal on Scientific Computing*, vol. 29, no. 4, pp. 1420–1438, 2007.

[6] E. Yakimiw, "Accurate computation of weights in classical gauss-christoffel quadrature rules," *J. Comput. Phys.*, vol. 129, pp. 406–430, December 1996.

[7] P. N. Swarztrauber, "On computing the points and weights for gauss-legendre quadrature," *SIAM J. Sci. Comput.*, vol. 24, pp. 945–954, March 2002.

[8] W. Gautschi, "How Sharp is Bernsteins Inequality for Jacobi Polynomials?" *Electronic Transactions on Numerical Analysis*, vol. 36, pp. 1–8, 2009.

[9] D. Goldberg, "What every computer scientist should know about floating-point arithmetic," *ACM Comput. Surv.*, vol. 23, pp. 5–48, March 1991.

[10] M. Abramowitz and I.A. Stegun, *Handbook of Mathematical Functions with Formulas, Graphs and Mathematical Tables*, ser. Advanced Mathematics. New York: Dover Publications, Inc., 1965.

[11] F. W. J. Olver, D. W. Lozier, R. F. Boisvert, and C. C. W., *NIST Handbook of Mathematical Functions.* Cambridge University Press, 2010.

[12] G. Szegö, *Orthogonal Polynomials.* Providence, Rhode Island: American Mathematical Society, 1978.

[13] T.-J. Stieltjes, "Sur les Polynômes de Legendre," *Annales de la faculté des sciences de Toulouse, premiere série*, vol. 4, no. 2, pp. 1–17, 1890.

[14] F. W. J. Olver, *Asymptotics and Special Functions*, 2nd ed. A.K. Peters/CRC Press, 1997.

[15] G. Szegö, "Über Einige Asymptotische Entwicklungen der Legendreschen Funktionen," *Proc. London Math. Soc.*, vol. s2-36, no. 1, pp. 427–450, 1934.

[16] E. Heine, *Handbuch der Kugelfunktionen: Theorie und Anwendung.* Berlin: Georg Reimer, 1861.