JOURNAL OF COMPUTATIONAL PHYSICS **129,** 406–430 (1996)
ARTICLE NO. 0258

Accurate Computation of Weights in Classical Gauss–Christoffel Quadrature Rules

E. Yakimiw

Recherche en prévision numérique, 2121 Route Transcanadienne, nord, Dorval, Québec, Canada H9P 1J3

Received June 20, 1996

For many classical Gauss–Christoffel quadrature rules there does not exist a method which guarantees a uniform level of accuracy for the Gaussian quadrature weights at all quadrature nodes unless the nodes are known exactly. More disturbing, some algebraic expressions for these weights exhibit an excessive sensitivity to even the smallest perturbations in the node location. This sensitivity rapidly increases with high order quadrature rules. Current uses of very high order quadratures are common with the advent of more powerful computers, and a loss of accuracy in the weights has become a problem and must be addressed. A simple but efficient and general method for improving the accuracy of the computation of the quadrature weights is proposed. It ensures a high level of accuracy for these weights even though the nodes may carry a significant large error. In addition, a highly efficient root-finding iterative technique with superlinear converging rates for computing the nodes is developed. It uses solely the quadrature polynomials and their first derivatives. A comparison of this method with the eigenvalue method of Golub and Welsh implemented in most standard software libraries is made. The proposed method outperforms the latter from the point of view of both accuracy and efficiency. The Legendre, Lobatto, Radau, Hermite, and Laguerre quadrature rules are examined.

1. INTRODUCTION

The advent of powerful computers has led over the last three decades to extensive research into quadrature, a numerical technique for evaluating integrals. Methods for constructing efficient classical Gauss–Christoffel quadrature rules and for computing the associated Christoffel numbers have been known for some time (see, for instance, Hildebrand [1], Gautschi [2], and for a comprehensive list of references Davis and Rabinowitz [3]). To this day, research activities related to this field and its applications are still carried on in several important areas (Gautschi [4], Milovanovic [5], Mastroianni and Monegato [6], Gautschi and Li [7]). This paper is mainly concerned with the computation of the Christoffel numbers or the quadrature weights for high order classical Gauss–Christoffel quadrature rules, where all the nodes are real and simple, the quadrature polynomials satisfy a three-term recurrence relation and a second-order differential equation. It is known that the sensitivity of the weights to small perturbations in the node location is not uniform over all the nodes. Some algebraic expressions suggested in the literature for computing the weights performed better than others. This has been noticed in Gautschi [8], Lether [9], and more recently in Nehrkorn [10] and Yakimiw [11]. The sensitivity of the Gaussian weights to the accuracy of the nodes is not a major problem for low order quadrature rules. However, for relatively high order quadrature rules, certain expressions for the weights are so sensitive to even small round-off errors that they are practically useless for computing the weights. Very high order Legendre quadrature rules are currently used in global atmospheric spectral models of high resolution (Ritchie *et al.* [12]). Spectral modellers are also well aware of the extreme sensitivity to the node precision of certain analytic expressions for computing the Gauss–Legendre quadrature weights near the polar regions. In high resolution atmospheric spectral models, a loss of accuracy in the weights significantly impacts the quality of long time integrations of complex atmospheric fluid flows and thus the sensitivity problem must be dealt with. Hence in general, it is incorrect to presume that nodes which are accurate to machine precision will invariably produce weights of the same level of accuracy. A similar although less severe accuracy problem is also present in the eigenvalue method originally proposed by Golub and Welsh [13], where the weights are calculated from the first eigenvector component. This aspect will be examined in some detail.

The main purpose of this paper is to describe a general method for deriving in classical Gauss–Christoffel quadrature rules algebraic expressions for the weights which are more robust and largely independent of perturbations in the node location. Furthermore, a simple but highly efficient technqiue for accurately computing the nodes is included. This technique is essentially an extension of Lether's [9] suggestion and is based on the work of Ostrowski [14] and Traub [15]. The general method is then applied to various classical Gauss–Christoffel quadrature rules. Because of its particular interest to atmospheric spectral modellers, the Gauss–Legendre quadrature rule is exam-

0021-9991/96 $18.00

ined in great detail. To assess the value of the method, a comparison of the results with those obtained with the eigenvalue method is carried out.

The organization of the paper is as follows. Section 2.1 briefly recalls the essentials of the eigenvalue method of Golub and Welsh for the computation of the nodes and the weights in classical Gauss–Christoffel quadrature rules. It is followed, in Section 2.2, by a detailed description of a root-finding iteration technique which exhibits superlinear convergence rates. Then, a general rule for deriving robust analytic expressions for the Gaussian weights is developed in Section 2.3. An implementation of this technique in the Gauss–Legendre quadrature rule appear in Section 3, together with a thorough comparison with the eigenvalue method in Section 3.3. In the Appendix, the proposed method is applied to derive formulas for many other classical Gauss–Christoffel quadrature rules, in particular, for Lobatto, Radau, Hermite, and Laguerre quadratures. In the conclusion (Section 4), we summarize and comment on the results.

2. CLASSICAL GAUSS–CHRISTOFFEL QUADRATURE RULES

Hereafter, we refer to the classical Gauss–Christoffel quadrature rules as those which share the following properties: (i) the quadrature nodes are all real and simple; (ii) the quadrature polynomials satisfy a three-term recurrence relation and obey a second-order differential equation. Two alternate methods for the computation of the weights in these quadrature rules are discussed in this paper. These two methods are referred to as the eigenvalue method and the root-finding method. Both techniques are described in many classic textbooks on numerical integration (See Davis and Rabinowitz [3], for instance.)

2.1. *The Eigenvalue Method*

The eigenvalue method which was originally proposed by Golub and Welsh [13] is based on the diagonalization of a symmetric tridiagonal Jacobi matrix generated from a three-term recurrence relation satisfied by the quadrature polynomials. An efficient procedure for diagonalizing the Jacobian matrix makes use of the implicitly shifted QL algorithm described in Wilkinson and Reinsch [16]. The eigenvalues of the matrix are just the quadrature nodes and the weights are simply related to the corresponding eigenvectors. More precisely, given a set of orthogonal polynomials $\{p_i\}_{i=0}^{n}$ obeying a three-term recurrence relation

$$xp_{i-1}(x) = a_i p_i(x) + b_i p_{i-1}(x) + a_{i-1} p_{i-2}(x),$$

one can deduce a matrix relation

$$x\mathbf{P}(x) = \mathbf{J}\mathbf{P}(x) + a_n p_n(x)\mathbf{E},$$

where

$$\mathbf{P}(x) = [p_0(x), p_1(x), p_2(x), ..., p_{n-1}(x)]^{\mathrm{T}},$$

and $\mathbf{E}$ $[0, 0, 0, ..., 1]^{\mathrm{T}}$. In this relation, $\mathbf{J}$ is the symmetric tridiagonal Jacobi matrix whose elements are the Gram–Schmidt coefficients appearing in the three-term recurrence relation. Hence, if x_k is a zero of $p_n(x)$, there follows

$$x_k\mathbf{P}(x_k) = \mathbf{J}\mathbf{P}(x_k),$$

and the eigenvalues of $\mathbf{J}$ are the requested quadrature nodes. Making use of the Christoffel–Darboux identity and the fact that $p_n(x_k) = 0$, the Gaussian weights are then expressed in terms of the eigenvectors of the Jacobian matrix. In particular, the weights can be computed from the first component of the orthonormal eigenvectors of $\mathbf{J}$.

The eigenvalue method of Golub and Welsh is a powerful and efficient technique specially suited for classical Gauss–Christoffel quadrature rules of low order. Since the three-term recurrence relations for the quadrature polynomials are precisely known, the coefficients of the Jacobi matrix can be evaluated to machine precision. If this is not the case, the method may be subject to numerical instabilities, thus multiple-precision arithmetic may be required in order to obtain meaningful results (Gautschi [8]). The eigenvalue method of Golub and Welsh is implemented in most standard numerical software libraries. For example, we obtained a revised version of the original Golub and Welsh subroutines from the *Netlib* software library. Numerical results obtained with this code are used for comparison in Section 3.3.

2.2. *The Root-Finding Method*

The root-finding method is an alternative for locating the nodes in Gauss–Christoffel quadrature rules since these nodes are the zeros of the associated quadrature polynomials. The method essentially rests on the general theory for the solution of equations and systems of equations elaborated by respected mathematicians such as Cauchy, Chebyshev, Euler, Fourier, Gauss, Lagrange, Laguerre, and Newton. A comprehensive description of the theory can be found in the textbooks of Ostrowski [14] and Traub [15].

Using a first approximation of the zero location of a function, the root-finding method aims at improving the initial guess by means of an iterative procedure involving the function itself and its derivatives. The whole technique relies on the assumption that the successive iterates will eventually converge to the exact location of the required zero. This convergence is secured, provided the function is sufficiently smooth in a certain neighborhood of the zero (i.e., avoiding rapidly oscillating functions or singular functions, the derivatives of which are either undefined or

zero) and that the zero initial guess location is close enough to its exact location. Amongst other reasons, the rate of convergence of the method closely depends on the particular value of the first derivative of the iterative function itself. The optimum theoretical convergence rate of an iterative scheme is always reached in the very close vicinity of the zero itself. A well-known example of a root-finding iterative scheme is the second-order Newton–Raphson rule. Near the zero of the function, the rate of convergence of the scheme is quadratic. (For a formal discussion of the convergence properties of the iterative functions, see Ostrowski [14, Chapter 7].) In the Gauss–Legendre quadrature rule application in Section 3, we shall examine in greater detail how the convergence properties surface and can be improved.

Implementations of iterative schemes with superlinear convergence rates are rather limited, owing to the apparent complexity brought upon by the computation of higher order derivatives of the function. For this reason, the Newton–Raphson scheme is usually chosen for its simplicity and its well-defined properties. Some exceptions of higher order iterative schemes are reported, for example, in the textbook of Davis and Rabinowitz [3]. In his original paper, Lether [9] develops a fifth-order iterative function for the Gauss–Legendre quadrature rule. The main merit of this paper is to have successfully implemented the fundamental and general theory of iterative methods for the solution of equations to the construction of superlinear converging iterative functions in classical Gauss–Christoffel quadrature rules. Furthermore, Lether suggests a straightforward substitution technique for eliminating higher order derivatives owing to the fact that the quadrature polynomials obey a second-order differential equation. Unfortunately, for high order iterative functions, the author himself admits that the algebra in this technique rapidly becomes prohibitive. In the following, we shall indicate how to circumvent this difficulty. Using fundamental properties shared by the iterative functions, we shall establish a general method for deriving high order iterative functions involving at most the quadrature polynomials and their first derivatives of the classical Gauss–Christoffel quadrature rules.

In this paper we shall be concerned with a real single-valued function $f(x)$ of a real variable x which possesses a certain number of continuous derivatives in the neighborhood of a zero α. After Traub [15], recall the following theorem.

THEOREM 2.2.1. *If the first derivative $f'(x)$ of $f(x)$ does not vanish in an interval about a zero α of $f(x)$ and if the lth derivative $f^{(l)}(x)$ is continuous in this interval, then $f(x)$ has an inverse $g(x)$ and $g^{(l)}(x)$ is continuous in the interval.*

Assume $\alpha = x + Dx$, where Dx is a relatively small deviation from α and α is such that $f(\alpha) = 0$ and $f'(\alpha) \neq 0$. Furthermore, let $f^{(j)}(x) = d^j f(x)/dx^j$ represents the jth derivative of $f(x)$. Finally, let us define a function f_j such that

$$f_j = \frac{f^{(j)}(x)}{f'(x)}, \quad j = 0, 1, 2, ... \tag{1}$$

Notice that $f_1 = 1$. (Hereafter, we shall use $f'(x) \equiv f^{(1)}(x)$ to express the first derivative of $f(x)$ and we shall drop the explicit x dependency whenever this is convenient.) In particular, we have

$$f_j' = f_{j+1} - f_2 f_j, \quad j = 0, 1, 2, ... \tag{2}$$

A Taylor series expansion of $f(\alpha)/f'(x)$ about α in terms of Dx gives

$$\sum_{j=0}^{\infty} \frac{f_j}{j!} (Dx)^j = 0. \tag{3}$$

If $f(x)$ meets the conditions of Theorem 2.2.1, then (3) can be inverted,

$$Dx = -\sum_{j=1}^{\infty} \frac{A_j}{j!} f_0^j, \tag{4}$$

where f_0 is defined according to (1) and the coefficients A_j are given by the recurrence relation

$$\begin{aligned} A_1 &= 1 \\ A_j &= (j-1) f_2 A_{j-1} - A_{j-1}', \quad j = 2, 3, ... \end{aligned} \tag{5}$$

In particular,

$$\begin{aligned} A_1 &= 1 \\ A_2 &= f_2 \\ A_3 &= 3f_2^2 - f_3 \\ A_4 &= 15f_2^3 - 10f_2 f_3 + f_4 \\ A_5 &= 105f_2^4 - 105f_2^2 f_3 + 10f_3^2 + 15f_2 f_4 - f_5 \\ A_6 &= 945f_2^5 - 1260f_2^3 f_3 + 280f_2 f_3^2 + 210f_2^2 f_4 \\ &\quad - 21f_2 f_5 - 35f_3 f_4 + f_6. \\ &\cdots \end{aligned} \tag{6}$$

(One can show that the sum of the coefficients in A_j is $(j-1)!$. See Traub [15]) An alternative method for deducing A_j in terms of the derivatives of f is through the identity

$$A_j = (-1)^j f_0^j \sum_r (-1)^r (j-1+r)! \prod_{i=0}^{j} \left(\frac{f_i}{i!}\right)^{\beta_i} \frac{1}{\beta_i!},$$

where the sum is taken over all nonnegative integers β_i satisfying the equations

$$\sum_{i=2}^{j} (i-1)\beta_i = j-1, \quad r = \sum_{j=2}^{j} \beta_j .$$

Finally, if we define $B_j = A_j f_0^j$, then the functions B_j obey the recurrence relation

$$\begin{aligned} B_1 &= f_0 \\ B_j &= (j-1)B_{j-1} - f_0 B'_{j-1}, \quad j = 2, 3, ... \end{aligned} \tag{7}$$

It is important to realize that Eq. (4) essentially states that, in principle, the correction Dx to x can be computed exactly without any need to iterate. In effect, the ultimate level of accuracy of Dx is limited by the machine precision. Furthermore, the series (4) is more or less severely truncated which further reduces its accuracy. An iteration procedure is thus required in order to determine the location of the zero to the machine precision.

Let a kth-order iterative scheme function be defined as

$$\{Dx\}_k = -\sum_{j=1}^{k-1} \frac{A_j}{j!} f_0^j, \quad k = 2, 3, ... \tag{8}$$

For instance, the iterative function $\{Dx\}_2 = -f_0$ leads to nothing but the second-order Newton–Raphson iterative scheme. If $f(x)$ is a polynomial of order n and $n \gg 1$, the third order iterative scheme defined according to (8) can be shown to reduce to the cubically convergent Laguerre scheme (Wilkinson [17], Fox and Mayers [18]). The function $\{Dx\}_k$ in (8) satisfies the recurrence relation

$$\begin{aligned} \{Dx\}_k &= \{Dx\}_{k-1} - \frac{A_{k-1}}{(k-1)!} f_0^{k-1} \\ &= \{Dx\}_{k-1} - \frac{f_0}{k-1}(1 + \{Dx\}'_{k-1}), \quad k = 3, 4, ..., \end{aligned} \tag{9}$$

where $\{Dx\}'_{k-1} \equiv d(\{Dx\}_{k-1})/dx$. The general iterative method using (4) can be related to many particular techniques and has sometimes been referred to as the reversion of the Taylor series expansion method (Wilkinson [17]) or the inverse interpolation method (Fox and Mayers [18], Traub [15]). It needs to be mentioned here that Gerlach [19] has recently described yet another method for constructing superlinear converging iterative functions. The method is basically a variant of (4). Indeed, let X be defined such as

$$X = \sum_{j=2}^{\infty} \frac{A_j}{j!} f_0^{j-1}.$$

If $X^2 < 1$, relation (4) can be written as

$$Dx = -\frac{f_0}{1-X}.$$

Hence, superlinear converging iterative functions can be derived. In particular, for $m = 2$ in Gerlach's convention, we obtain

$$Dx = -\frac{f_0}{1 - \frac{1}{2} f_2 f_0} \simeq -f_0 \left(1 + \frac{1}{2} f_2 f_0\right)$$

which is equivalent to the third-order scheme according to (8). However, for $m = 3$, applying his method to the iterative function

$$Dx = -f_0 \Big/ \left[1 - \frac{1}{2} f_2 f_0 - \frac{1}{6}\left(\frac{3}{2} f_2^2 - f_3\right) \frac{f_0^2}{1 - \frac{1}{2} f_2 f_0}\right]$$

where f_0, f_2, f_3 are defined according to (1). From this expression it is clear that higher order iterative functions are just a series expansion in terms of the iterative function of the preceding order. However, it is possible to show that this expression for Dx is numerically as good as the fifth-order scheme but certainly no better than the sixth-order iterative functions. Gerlach's technique allows reproducing superlinear converging iterative functions, however, at the cost of undesirable complexity. For higher m values, the algebra becomes rather tedious and simply impracticable unless the functions whose zeros are sought are simple. This contrasts with the method based on (4). Given relations (4) and (5), it is relatively straightforward to construct iterative schemes with superlinear convergence rates, provided one is willing to compute the higher derivatives, as indicated in (6). This, however, does not constitute an outstanding numerical problem, provided $f(x)$ satisfies a second-order differential equation. Indeed, if

$$f''(x) = a_2(x) f'(x) + b_2(x) f(x)$$

or, according to the adopted convention defined in (1),

$$f_2 = a_2(x) + b_2(x) f_0; \tag{10}$$

then, the following relation holds:

$$f_j = a_j(x) f_{j-1} + b_j(x) f_{j-2}, \quad j = 2, 3, ... \tag{11}$$

There are obvious advantages in using (4) for computing higher derivatives. Nevertheless, one may choose to express these derivatives in terms of ascending powers of f_0, as in Lether [9]. As we mentioned earlier, for high order

iterative schemes, the method of direct substitution suggested by Lether is cumbersome and prone to errors. (In fact, the erroneous factor C in Lether's formula has been corrected in the next section.) In addition, the method is simply unworkable in certain classical Gauss–Christoffel quadrature rules (e.g., Radau quadrature rule).

The hereafter proposed simpler method makes use of one of the fundamental intrinsic properties shared by the iterative functions (4). According to Traub [15, Theorem 2.6, and Chap. 5), if the properties of a basic sequence are known, then many properties of any iteration function may be deduced. It is these properties that we exploit to derive simple superlinear convergent schemes. In particular, using the relation (5), it can easily be demonstrated that (4) obeys the following property:

$$\frac{d(Dx)}{dx} = -1. \tag{12}$$

It is clear then that one can construct various equivalent iterative functions with superlinear convergence rates, provided property (12) is satisfied. This freedom of choice will be examined in greater detail in the Legendre and Lobatto quadrature rules. For now, consider an expression of (4) in terms of f_0 such that

$$Dx = -\sum_{j=1}^{\infty} \frac{d_j}{j!} f_0^j, \tag{13}$$

where the coefficients d_j are some relatively simple functions of x to be determined but are not functions of $f(x)$ or its derivatives. If (13) has to satisfy condition (12) in some neighborhood of the nodes, then the recurrence relation for the coefficients d_j,

$$\begin{aligned} d_1 &= 1 \\ d_{j+1} &= -d_j' + ja_2 d_j + j(j-1)b_2 d_{j-1}, \quad j = 1, 2, ..., \end{aligned} \tag{14}$$

holds, assuming (11) is true. Applying this relation to various classical quadrature polynomials allows us to easily derive iterative schemes of any desired order with superlinear convergence rates solely in terms of very simple functions and of f_0. A practical example of this method is given in Section 3 for the Gauss–Legendre quadrature rule. Other results for the Lobatto, Radau, Hermite, and Laguerre quadrature rules are reported in the Appendix.

2.3. *Computation of Gaussian Quadrature Weights*

The computation of the Gaussian weights calls upon some analytic expressions involving the quadrature polynomials which must be evaluated at the node locations. The precision of these weights more or less depends on the accuracy of the quadrature nodes itself. In fact, this dependency does create serious numerical problems in many classical Gauss–Christoffel quadrature rules. The eigenvalue method is not exempt from this peculiar problem. Indeed, in this approach, where the weight function is related to the first component of the eigenvectors, a loss of accuracy always occurs when the Gaussian weights are relatively very small; besides, the small nodes themselves lose accuracy in this method. The main reason for this deficiency is that the accuracy of the nodes and the weights directly depends on the accuracy of the initial Jacobi matrix elements. Fortunately, these matrix elements are obtained from exact recurrence relations and their level of precision is determined according to the floating point arithmetic used in their computation. In double precision, for instance, the initial matrix elements are accurate to at least 15 significant digits. However, the eigenvalues (or the nodes) and the corresponding eigenvectors (hence, the weights) are only accurate to 15 decimal figures. Whatever the resulting loss of accuracy, it is certainly not to the benefit of using higher order quadratures to evaluate integrals more precisely.

In the root-finding method, the accuracy problem of the weights can be even more acute, depending on the analytic expressions chosen for their computation. For very high order quadrature rules, some expressions are so sensitive to small round-off errors in the location of the nodes that they produce completely erroneous values. Sensitivity to small perturbations of the node locations has been observed previously (Lether [9], Nehrkorn [10]) and is known to exist in many classical Gauss–Christoffel quadrature rules (Yakimiw [11]). Basically, Yakimiw showed that large error in the weights due to small deviations in the node location is bound to be generated when a singularity; hence an abrupt variation of the weight functions happens to be too close to the position of the nodes. A typical example of such behavior is depicted in Fig. 1 for the Gauss–Legendre quadrature of orders $n = 6$, where the weight is

$$w(x) = \frac{2(1-x^2)}{[nP_{n-1}(x)]^2}. \tag{15}$$

Such a formula is widely used in spectral atmospheric models (Ritchie *et al.* [12]) and can be derived from the Christoffel–Darboux identity and the recurrence relation for the ordinary Legendre polynomials. The heavy line in Fig. 1 represents th Gauss–Legendre weight function $w(x)$ in terms of x in the half-interval [0, 1]. The Gaussian weights are obtained at the nodes, the locations of which are indicated by the long-dash vertical lines. The dotted line plots the ordinary Legendre polynomials $P_6(x)$ to a constant. One can see that a small deviation or error in the position of the node closest to $x = 1$ will necessarily

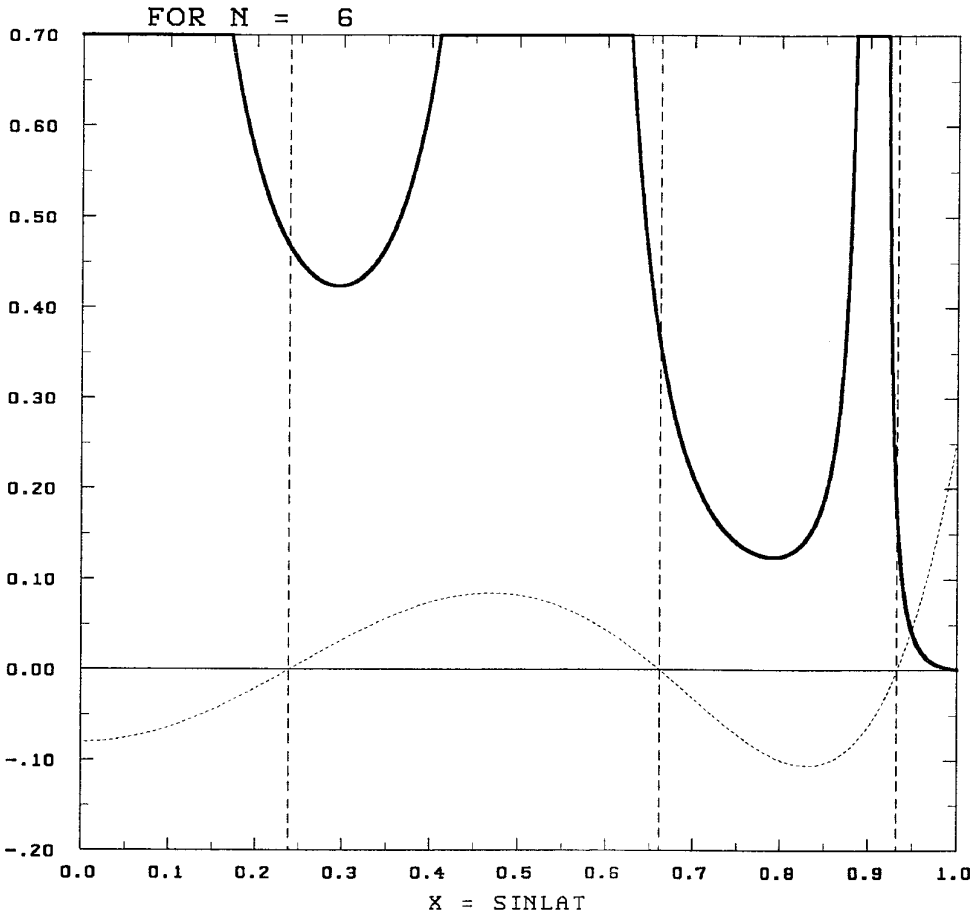

FIG. 1. Gauss–Legendre quadrature weight function (15) for $n = 6$. The heavy line represents the Gauss–Legendre weight function $w(x)$ given by (15) for $n = 6$. The long dash vertical lines indicate the x-position of the quadrature nodes (the zeros of the ordinary Legendre polynomials of degree $n = 6$ whose behavior is represented by the dotted line). The weights and the nodes are symmetric with respect to $x = 0$ in the interval $[-1, 1]$. Notice the extremely sharp variation of $w(x)$ at the nodes closest to the end-point of the interval.

induce a large error in the weight at that location. Some numerical results when using (15) for the computation of the Gaussian weights are reported in column 3 of Tables III and IV for the Gauss–Legendre quadrature rule of order $n = 92$ and $n = 384$, respectively (These are typical values which can actually be used in numerical climate simulation and weather forecast spectral models.). The first Gaussian weights in that column are accurate to 11 and 9 significant digits, respectively, even though the nodes used to compute these weights are accurate to 16 significant digits! The control values are given in column 2 of these tables. The problem of accuracy for the weights to round-off errors of the nodes closest to the end-points of the interval increases so rapidly with n that expression (15) fails at some point to produce the correct values, unless multiple precision arithmetic is called upon.

An alternative expression for the Gauss–Legendre weights is written in terms of the first derivative of the quadrature polynomials.

$$w(x) = \frac{2(1 - x^2)}{[(1 - x^2)P'_n(x)]^2}. \tag{16}$$

The function is plotted in Fig. 2 and exhibits a less abrupt variation in the vicinity of the nodes. This is even greater for the nodes closest to the end-point.

Another well-known formula for the computation of the weights (e.g., Eq. (19) in Lether [9]) depicts a similar behavior in the vicinity of the nodes, as shown in Fig. 3. In order to alleviate the sensitivity of the Gaussian weights to the node location, simple expressions were derived in (Yakimiw [11]). In what follows we propose a general method for constructing analytic expressions for the Gaussian weights that are more robust within a significantly large neighborhood of the nodes.

First assume that $f(x)$ is a quadrature polynomial satisfying the condition stated in Theorem 2.2.1 (with $f'(x) \neq 0$ in the neighborhood of the nodes) and obeys a second-order differential such that (10) holds. Moreover,

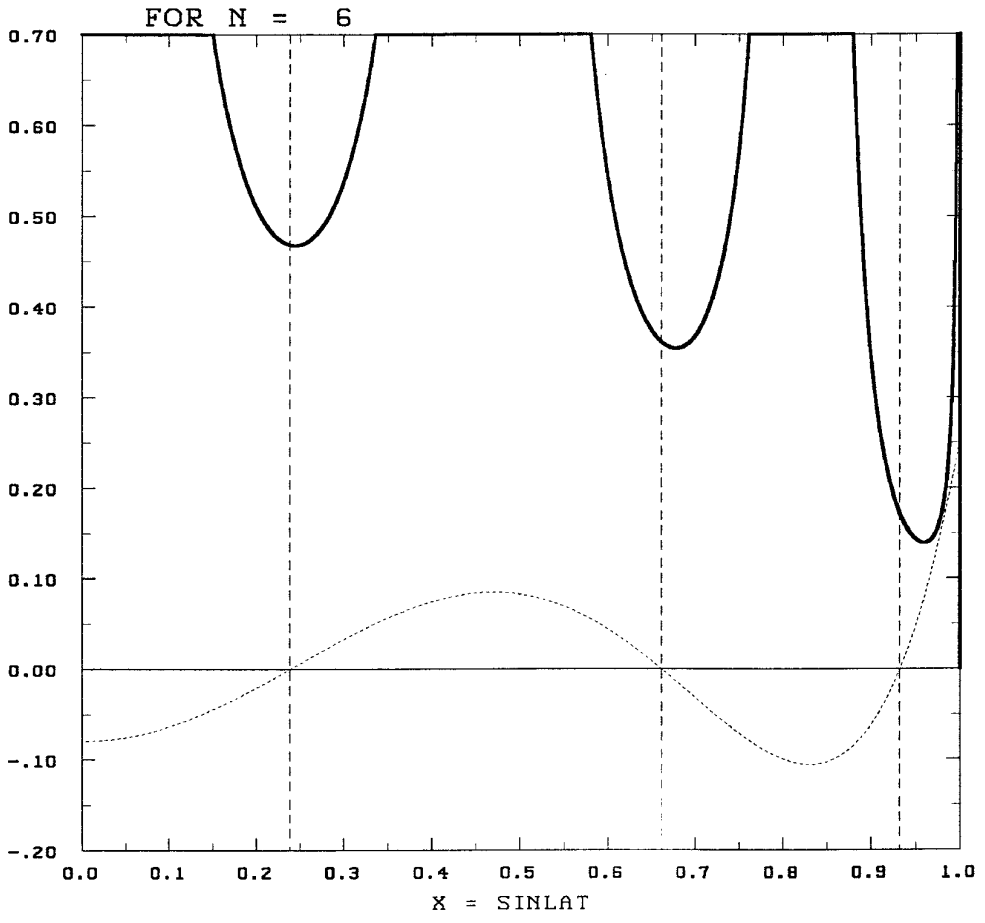

FIG. 2. Gauss–Legendre quadrature weight function (16) for $n = 6$. Same as in Fig. 1, except that the Gauss–Legendre weight function $w(x)$ given by (16) for $n = 6$. Notice the less abrupt variation of $w(x)$ at the nodes closest to the end-point of the interval compared to the variation shown in Fig. 1 in that neighborhood.

consider a weight function of the form

$$w(x) = \frac{k}{[d_0 f'(x)]^2}, \tag{17}$$

where k is some known constant and d_0 is a simple function of x. (For the Gauss–Legendre quadrature, $k = 2$ and $d_0 = \sqrt{1 - x^2}$). Then, a valid expression for computing the Gaussian weights is

$$w(x) = k \Big/ \left[f'(x) \sum_{j=0}^{\infty} \frac{d_j}{j!} f_0^j \right]^2, \tag{18}$$

where $f_0 = f(x)/f'(x)$ as defined in (1) and d_j is a simple function of x. (Notice that the d_j are different from those given previously in (14)). The purpose of (18) is to be able to modify the behavior of the weight function without changing its value in the vicinity of the nodes. This happens precisely when

$$\frac{dw(x)}{dx} = 0. \tag{19}$$

This condition readily implies that the coefficients d_j in (18) must satisfy the recurrence relation

$$\begin{aligned} d_{j+1} &= -d_j' + (j-1)a_2 d_j + j(j-2)b_2 d_{j-1}, \\ j &= 0, 1, 2, ..., \end{aligned} \tag{20}$$

where (10) has been used. By requesting that the weight function obeys (19), it is very simple to derive analytic expressions for computing the weights in the classical Gauss–Christoffel quadrature rules that are practically independent of small errors which may be associated with the nodes.

Retaining the first two terms in (18) leads to the results reported in Yakimiw [11]. For nodes accurate to machine precision, inclusion of these two terms is all that is needed to compute the weight to their highest accuracy. If the nodes carry a relatively large error, higher order terms must be incorporated. In the next section, we implement this method to construct weight functions for the Legendre quadratures that include the first six terms. Analogous

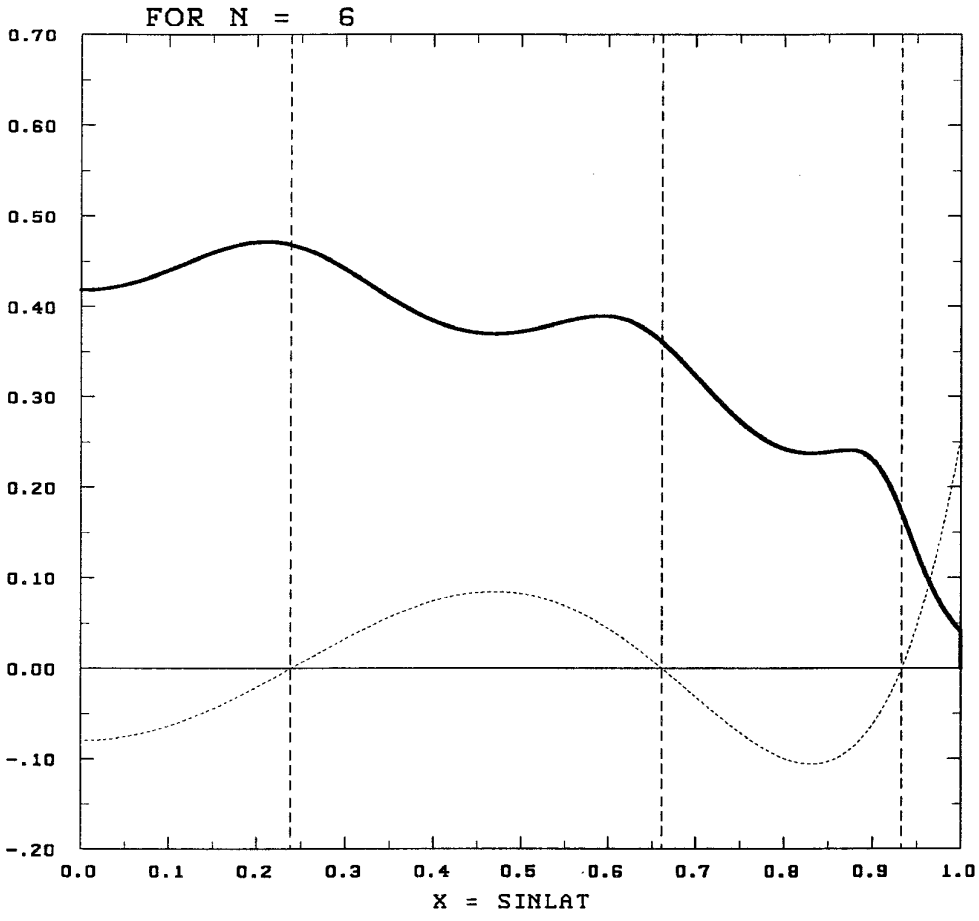

FIG. 3. Gauss–Legendre quadrature weight function for $n = 6$. Same as in Fig. 1, except that the Gauss–Legendre weight function $w(x)$ given by Eq. (19) in Lether [9], for $n = 6$. The behavior of the function $w(x)$ in the neighborhood of the nodes is very similar to the behavior of $w(x)$ given by (16) and shown in Fig. 2.

results for other Gauss–Christoffel quadrature rules are included in the Appendix. Finally, one must realize that it is possible to construct classes of weight functions other than (18) having similar properties to those derived in this paper provided condition (19) is imposed.

3. APPLICATIONS

In this section we exploit the method proposed in the previous section for deriving superlinear converging iterative schemes which enable us to compute the nodes accurately and efficiently. We then derive robust analytic expressions for the weights which satisfy condition (19). We examine the Gauss–Legendre quadrature rule in great detail in order to compare the results with the eigenvalue method of Golub and Welsh.

3.1. *Gauss–Legendre Quadrature*

The Gauss–Legendre quadrature rule is defined by the relation

$$\int_{-1}^{1} F(x)\,dx = \sum_{i=1}^{n} w(\alpha_i)F(\alpha_i), \tag{21}$$

where the α_i are the zeros of the ordinary Legendre polynomials $P_n(x)$ of degree n. Following the convention adopted in the previous section, we have $f(x) = P_n(x)$ and

$$f_j = \frac{P_n^{(j)}(x)}{P_n'(x)}, \quad j = 0, 1, 2, \ldots.$$

The ordinary Legendre polynomials obey the recurrence relations and identities

$$\begin{gathered} P_0(x) = 1, \quad P_1(x) = x, \\ (n+1)P_{n+1}(x) = (2n+1)xP_n(x) - nP_{n-1}(x), \\ c^2P_n'(x) = n[P_{n-1}(x) - xP_n(x)], \\ f_2 = \frac{2x}{c^2} - \frac{\overline{n}}{c^2}f_0 \equiv a_2 + b_2 f_0, \end{gathered} \tag{22}$$

where, for convenience, we have defined $c^2 = 1 - x^2$ and $\overline{n} = n(n+1)$. According to the adopted convention (1), higher order derivatives of the ordinary Legendre polyno-

mials satisfy the relation

$$f_j = \frac{2(j-1)x}{c^2} f_{j-1} + \frac{[(j-1)(j-2) - \bar{n}]}{c^2} f_{j-2}, \quad j = 2, 3, ... \tag{23}$$

Using (23) to evaluate the derivatives in (6) and replacing their values in (4), it is simple and straightforward to derive superlinear converging iterative schemes for computing the zeros of $P_n(x)$. To obtain instead a series expansion of Dx solely in terms of f_0 as in (13), we proceed as follows. In order to simplify the mathematical derivation, write (13) in the equivalent form,

$$Dx = -c^2 \sum_{j=1}^{\infty} \frac{\delta_j}{j!} \left(\frac{f_0}{c^2}\right)^j, \tag{24}$$

so that

$$d_j = \frac{\delta_j}{c^{2(j-1)}}, \quad d_j' = \frac{\delta_j'}{c^{2(j-1)}} + \frac{2(j-1)x\delta_j}{c^{2j}}.$$

Translated in terms of δ_j, recurrence relation (14) becomes

$$\begin{aligned} \delta_1 &= 1 \\ \delta_{j+1} &= -c^2\delta_j' + 2x\delta_j - j(j-1)\bar{n}c^2\delta_{j-1}, \quad j = 1, 2, ..., \end{aligned} \tag{25}$$

where, for the ordinary Legendre polynomials, we have used the fact that $a_2 = 2x/c^2$ and $b_2 = -\bar{n}/c^2$, according to (22) with $\bar{n} = n(n+1)$. Thus, the first five terms in (24) are

$$\begin{aligned} \delta_1 &= 1 \\ \delta_2 &= 2x \\ \delta_3 &= 2[2 - (3+\bar{n})c^2] = 2[(3+\bar{n})x^2 - (1+\bar{n})] \\ \delta_4 &= 4x[2 - (6+5\bar{n})c^2] = 4x[(6+5\bar{n})x^2 \\ &\quad - (4+5\bar{n})] \\ \delta_5 &= 4[4 - 2(15+16\bar{n})c^2 + (30+43\bar{n}+6\bar{n}^2)c^4] \\ &= 4[(30+43\bar{n}+6\bar{n}^2)x^4 - 6(2\bar{n}^2+9\bar{n}+5)x^2 \\ &\quad + (3\bar{n}+4)(2\bar{n}+1)]. \end{aligned} \tag{26}$$

It can be shown that the first four terms exactly reproduce the results given in Lether [9] (after correcting for a missing x in the expression of C in Lether).

It has been mentioned in Section 2.2 that it is possible to construct series expansions for Dx other than (24), provided that the new expression obeys condition (12). For instance, one such series expansion could be in terms of the ratio $P_n(x)/P_{n-1}(x)$. However, such a choice should be avoided since the zero of $P_{n-1}(x)$ are increasingly closer to the zeros of $P_n(x)$ near the end-points of the interval $[-1, 1]$ as n increases; hence, the stability properties of the series Dx deteriorate significantly in these regions. On the other hand, the following function is a good, if not a better, candidate for producing a highly stable series expansion for Dx with superlinear convergence rates:

$$\frac{cP_n(x)}{c^2P_n'(x) - xP_n(x)} = \frac{cf_0}{c^2 - xf_0}. \tag{27}$$

Defining $cF(x) = c^2P_n'(x) - xP_n(x)$, we obtain

$$Dx = -\sum_{j=1}^{\infty} \frac{\delta_j}{j!} \left(\frac{P_n}{F}\right)^j. \tag{28}$$

For (28) to satisfy (12), the coefficients δ_j must obey the recurrence relation

$$\begin{aligned} \delta_1 &= c \\ \delta_{j+1} &= -c\delta_j' - \frac{jx}{c}\delta_j - j(j-1)\frac{(\bar{n}c^2+1)}{c^2}\delta_{j-1}, \\ j &= 1, 2, ... \end{aligned} \tag{29}$$

In particular,

$$\begin{aligned} \delta_1 &= c, \\ \delta_2 &= 0, \\ \delta_3 &= -\frac{2(\bar{n}c^2+1)}{c}, \\ \delta_4 &= \frac{4x}{c^2}(\bar{n}c^2+2), \\ \delta_5 &= \frac{4}{c^3}[3\bar{n}c^4(2\bar{n}+1) + 2c^2(4\bar{n}+5) + 6], \quad \text{etc.} \end{aligned} \tag{30}$$

This leads to a simple but highly efficient iteration function Dx,

$$\{Dx\}_6 = -c^2h\left\{1 + \frac{h^2}{3}\left[A + \frac{h}{2}\left(B + C\frac{h}{5}\right)\right]\right\},$$

where $h = f_0/(c^2 - xf_0)$,

$$\begin{aligned} A &= -(\bar{n}c^2+1), \quad B = x(\bar{n}c^2+2), \\ C &= 3\bar{n}c^4(2\bar{n}+1) + 2c^2(4\bar{n}+5) + 6. \end{aligned}$$

The same function F will be shown to be also a good candidate for producing a robust analytic expression for $w(x)$. It is interesting to plot the function Dx given by (24) in terms of x and to picture the effect of adding more

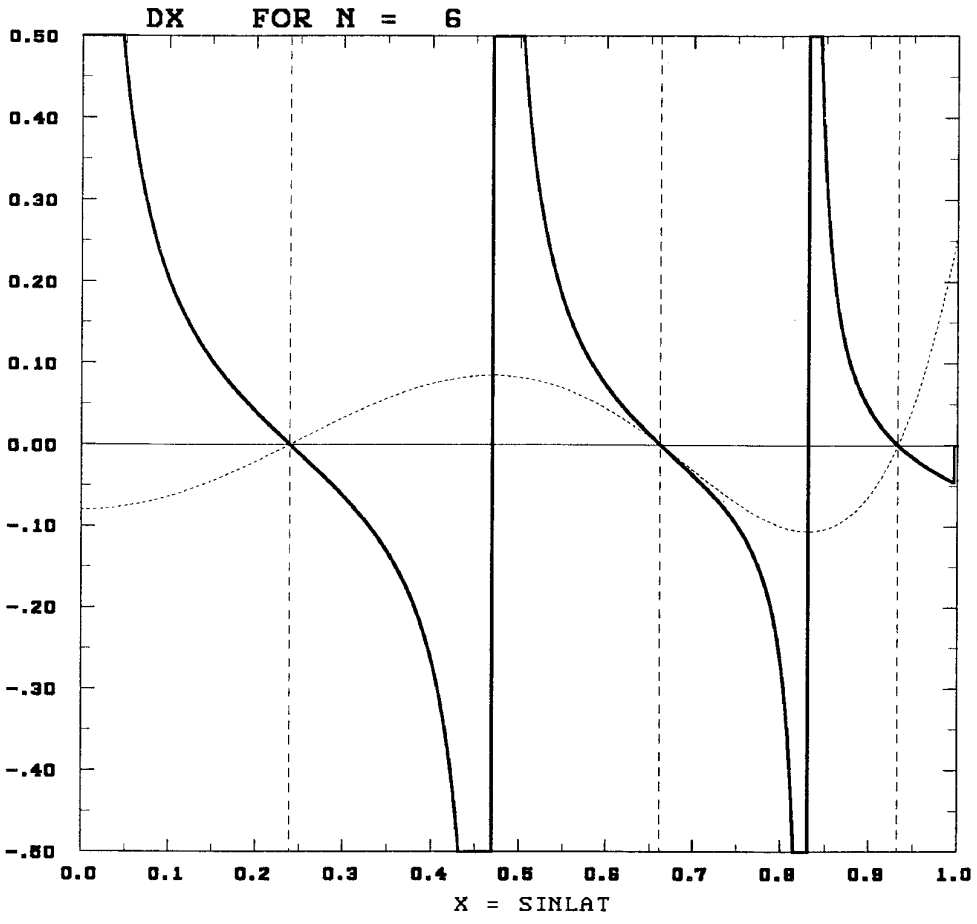

FIG. 4. Gauss–Legendre quadrature nodes for $n = 6$. The heavy line represents the function $\{Dx\}_2$ given by (31) (Newton–Raphson rule) for the Gauss–Legendre quadrature rule of order $n = 6$. The long dash vertical lines indicate the x-position of the quadrature nodes (the zeros of the ordinary Legendre polynomials of degree $n = 6$ whose behavior is represented by the dotted line). The nodes are symmetric with respect to $x = 0$ in the interval $[-1, 1]$. Notice the behavior of Dx in the neighborhood of the nodes. Singularities and nodes of $\{Dx\}_2$ interlace. The nodes are the attractors within the singularities. Any initial guess of the nodes within two singularities will eventually converge towards the node located between these singularities.

terms in the series. In Fig. 4, we plot the second-order iteration function,

$$\{Dx\}_2 = -f_0 = -\frac{P_n(x)}{P_n'(x)} \tag{31}$$

(Newton–Raphson rule). Using (2), we have

$$\frac{d\{Dx\}_2}{dx} = -(1 - f_2 f_0)$$

and $d(\{Dx\}_2)/dx|_\alpha = -1$. Thus, (31) satisfies (12) exactly at the node α only.

However, if we include the first five terms in (24), i.e.,

$$\{Dx\}_6 = -c^2 \sum_{j=1}^{5} \frac{\delta_j}{j!} \left(\frac{f_0}{c^2}\right)^j, \tag{32}$$

we get the graph shown in Fig. 5. From this graph, one can see that the slope of (32) is practically constant and equal to (-1) for a significant large interval around the nodes. Hence, such a function leads to a superlinear convergence scheme in this interval. A similar behavior can be shown to hold for expression (28).

A robust analytic expression for the Gauss–Legendre weights satisfying the condition (19) can be derived from (18). In order to simplify the mathematics, write (18) as

$$w(x) = 2 \Big/ \left[cf'(x) \sum_{j=0}^{\infty} \frac{\delta_j}{j!} \left(\frac{f_0}{c^2}\right)^j \right]^2, \tag{33}$$

where $\delta_0 = 1$. According to (20), this implies that

$$\delta_{j+1} = c^2 \delta_j' - x\delta_j - j(j-2)\overline{n}c^2\delta_{j-1}, \quad j = 0, 1, 2, \ldots \tag{34}$$

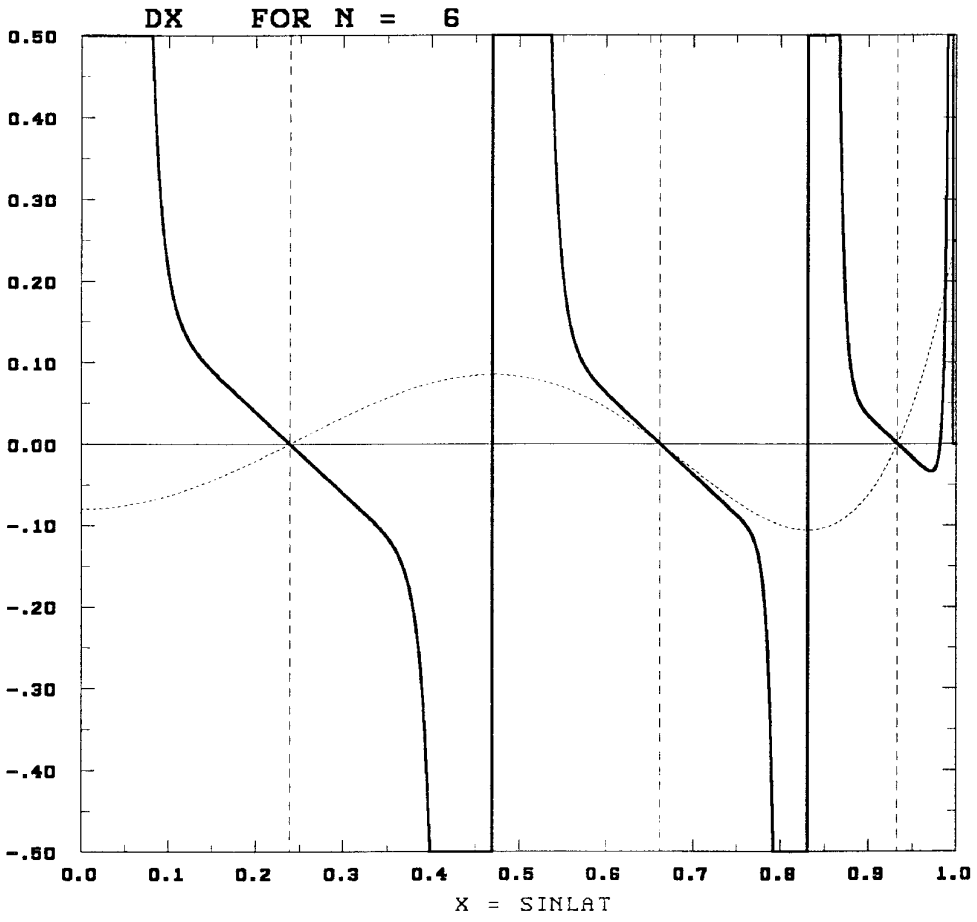

FIG. 5. Gauss–Legendre quadrature nodes for $n = 6$. Same as in Fig. 4, except that the function $\{Dx\}_6$ given by (32) includes the first five terms. Notice the linear behavior of Dx (with slope equal to -1) for a relatively large neighborhood around the nodes, leading to a superlinear converging iterative scheme in that neighborhood.

(where $c^2 = 1 - x^2$) and $\overline{n} = n(n+1)$). In particular, we find that

$$\begin{aligned}
\delta_0 &= 1,\\
\delta_1 &= -x,\\
\delta_2 &= \overline{n}c^2 + 1,\\
\delta_3 &= x(\overline{n}c^2 - 1),\\
\delta_4 &= 1 - 2\overline{n}c^2 - \overline{n}c^4(3\overline{n} + 2),\\
\delta_5 &= -x[1 - 6\overline{n}c^2 + \overline{n}c^4(17\overline{n} + 6)], \quad \text{etc.},
\end{aligned} \tag{35}$$

or explicitly,

$$w(x) = 2c^2 \Big/ \left\{ c^2 P_n' - xP_n + \frac{1}{2}\left[n(n+1) + \frac{1}{c^2}\right]\frac{P_n^2}{P_n'} + \cdots \right\}^2. \tag{36}$$

If we use the function F defined in (27), instead of f_0, we can derive yet another valid and more robust expression for the weight function:

$$w(x) = 2 \Big/ \left\{ F + \frac{1}{2}\left[n(n+1) + \frac{1}{c^2}\right]\frac{P_n^2}{F} - \frac{x}{3c}\left[n(n+1) + \frac{2}{c^2}\right]\frac{P_n^3}{F^2} + \cdots \right\}^2. \tag{37}$$

To visualize the effect of adding these terms in the weight formula, we plot the function (33) for $n = 6$. In Fig. 6, only the first two terms δ_0 and δ_1 in (33) are retained. Figure 7 plots (36), which includes the δ_2 term.

In these figures, the heavy line represents $w(x)$ and the vertical lines indicate the position of the nodes. These results should be compared with those of Fig. 1. It is clear from Fig. 7 that relatively large deviations or errors in the position of the nodes will not affect the value of the

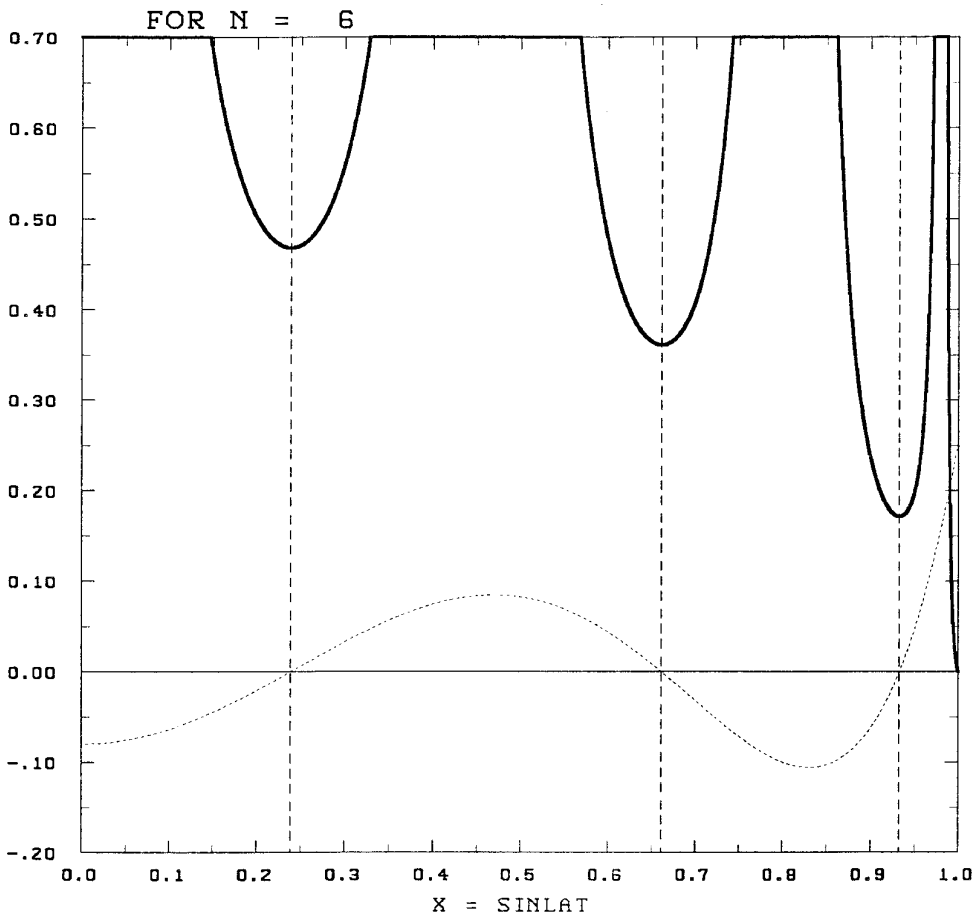

FIG. 6. Gauss–Legendre quadrature weight function for $n = 6$. Same as in Fig. 1, except that the heavy line represents the function $w(x)$ given by (33) for the Gauss–Legendre quadrature rule of order $n = 6$ when the first two terms are retained in the series (δ_0 and δ_1). Notice all the nodes coincide with the minima of $w(x)$, where its variation near these nodes is not large. Similar results were reported in Yakimiw [11].

Gaussian weights in the neighborhood of these nodes. We applied the same method to other quadrature rules and the results are summarized in the Appendix. Notice also that second-order correction terms to the analytic expressions of the weight functions in classical Gauss–Christoffel quadrature rules have been briefly examined in Yakimiw [22].

In the following section, we apply these formulas to actually compute the nodes and the weights in the Gauss–Legendre quadrature rule and we compare the results with the ones obtained using the eigenvalue method of Golub and Welsh.

3.2. *Initial Guess and Actual Computation*

In order to apply the root-finding technique described in the preceding paragraph, an initial guess of the nodes is needed. There exist quite accurate and efficient formulas for computing approximate values of the zeros of the ordinary Legendre polynomials. Let x_k be the kth zero of the ordinary Legendre polynomials $P_n(x)$. Then, a good approximation of these zeros is given in Lether [9]

$$x_k = \cos\left\{\frac{j_k}{\left(n^2+n+\frac{1}{3}\right)^{1/2}}\left[1-\frac{j_k^2-2}{360\left(n^2+n+\frac{1}{3}\right)^2}\right]\right\} + \mathcal{O}(n^{-7}) \tag{38}$$

and

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

where

$$\theta_k = \left(\frac{4k-1}{4n+2}\right)\pi$$

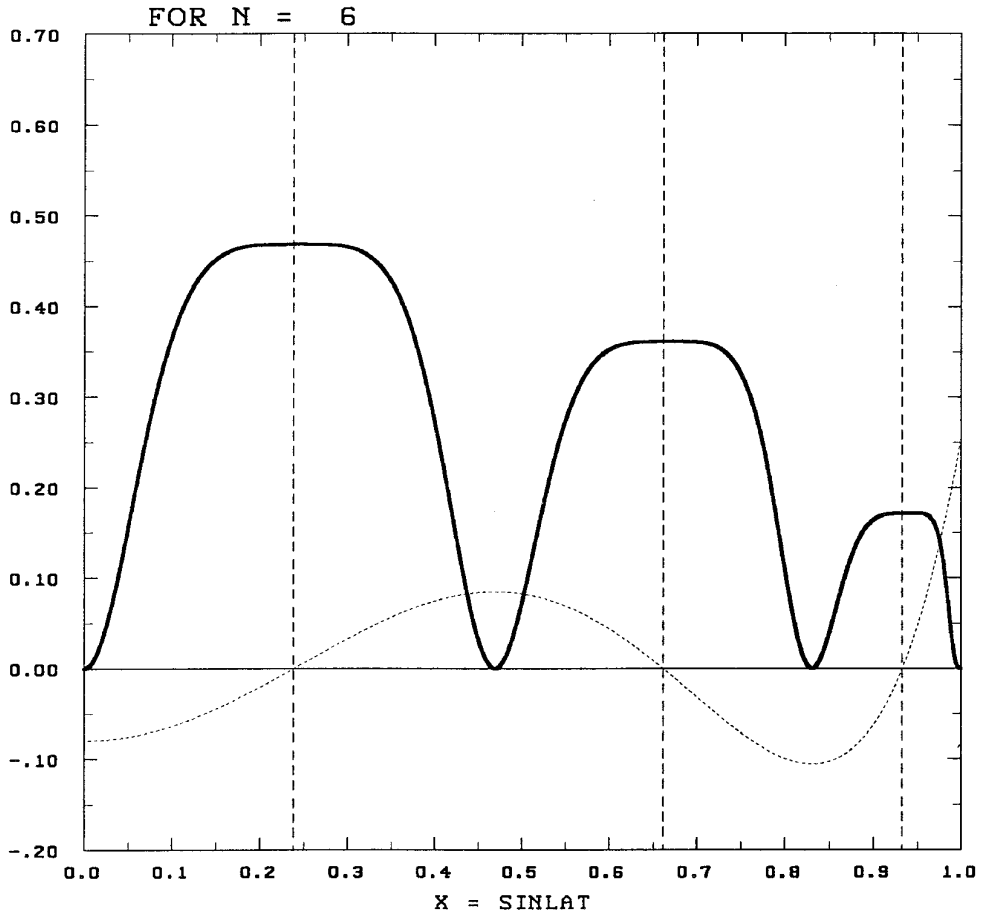

FIG. 7. Gauss–Legendre quadrature weight function for $n = 6$. Same as in Fig. 1, except that the weight function $w(x)$ is given by (36) for the Gauss–Legendre quadrature rule of order $n = 6$ when the first three terms (δ_0, δ_1, and δ_2) are retained in the series (33). Notice that linear (flat) behavior of $w(x)$ (with slope equal to 0) for a relatively large interval in the neighborhood of the nodes.

and j_k is the kth positive zero of the Bessel function of the first kind $J_0(x)$. A highly accurate approximation of j_k, valid for all k, can be found in Branders *et al.* [20],

$$j_k \approx \beta + \frac{a_0 + a_1\beta^2 + a_2\beta^4 + a_3\beta^6}{\beta + b_1\beta^3 + b_2\beta^5 + b_3\beta^7}$$

with $\beta = (4k - 1)\pi/4$ and

$$a_0 = 0.68289\,48973\,49453\ \text{E} - 01$$
$$a_1 = 0.13142\,08074\,70708\ \text{E} + 00$$
$$a_2 = 0.24598\,82418\,03681\ \text{E} - 01$$
$$a_3 = 0.81300\,57215\,43268\ \text{E} - 03$$
$$b_1 = 0.11683\,72425\,70470\ \text{E} + 01$$
$$b_2 = 0.20099\,11221\,97811\ \text{E} + 00$$
$$b_3 = 0.65040\,45772\,61471\ \text{E} - 02$$

It can be shown that expression (38) gives a better approximation of the zeros closest to the end-point interval $[-1, 1]$, whereas expression (39) is superior in the middle of the interval. An empirical rule has been designed for selecting the formula which leads to the best approximation. The rule goes as follows: Use formula (38) whenever k is such that

$$k \le \text{NINT}\left\{\frac{0.062}{n}(n + 33)(n - 1.5)\right\}; \qquad (40)$$

otherwise, take (39), where NINT means the nearest integer. For $n \ge 5$, for instance, formula (38) gives at least 14 significant correct digits for the first few nodes closest to the end-points. In general for other values of k and for high values of n, the above rule guarantees at least eight accurate significant digits for the nodes. We do not recommend using (39) for the nodes closest to the endpoints since it gives a relatively poor initial guess for these nodes. Lower order formulas analogous to (39) (which are used in many applications) provide a poor initial guess for the nodes. Lether [9] suggested using (38) only for the first node nearest the end-point of the interval. Given the initial

guesses of the nodes, first the ordinary Legendre polynomials $P_n(x)$ are evaluated using the recurrence relation and its first derivative according to (22). This computation is relatively the most expensive operation of the entire scheme. With these initial approximations of the nodes, we use a fifth-order iterative scheme based on (24) and (26) to determine the correction $\{Dx_k\}_5$ to x_k. Given the initial guess suggested above and for quadrature order sufficiently large, the nodes $\alpha_k = x_k + \{Dx_k\}_5$ are then computed to machine precision (16 significant digits in double-precision arithmetic). For very low quadrature orders (for n less than 5), an iteration may be required. In all the applications, this one-iteration scheme was used resulting in practically no significant loss in efficiency. Since the Legendre polynomial and its first derivative need to be evaluated at the new values, $x_k + Dx$, the following procedure was adopted. In the iteration step, rather than using the recurrence relation for the calculation of the Legendre polynomial and its first derivatives, it is more efficient to compute the required quantities by means of a Taylor series in terms of Dx. According to the convention adopted in Section 2.2 and using (23), evaluate $f_0(x_k + Dx) = P_n(x_k + Dx)/(P'_n(x_k + Dx)$, where

$$\frac{P_n(x_k + Dx)}{P'_n(x_k)} = \sum_{j=0}^{5} \frac{f_j}{j!} Dx^j,$$
$$\frac{P'_n(x_k + Dx)}{P'_n(x_k)} = \sum_{j=0}^{4} \frac{f_{j+1}}{j!} Dx^j. \tag{41}$$

Following this procedure ensures full precision for the nodes for any quadrature order. Once the nodes have been computed, apply (33), together with (35), to calculate the Gaussian weights. Observe that the accuracy to which the weights may be found is now limited only by the accuracy with which the ordinary Legendre polynomial and its derivative may be evaluated. For this reason, we suggest using a highly accurate formula for these computations near the end-points of the interval of integration (θ is the colatitude)

$$P_n(\theta) = \frac{1}{2^{2n}} \sum_{j=0}^{\lfloor n/2 \rfloor} \varepsilon_{n-2j} \frac{(2j)!(2n-2j)!}{[j!(n-j)!]^2} \cos[(n-2j)\theta] \tag{42}$$

and

$$P'_n(\theta) = -\frac{1}{2^{2n}} \sum_{j=0}^{\lfloor n/2 \rfloor} \varepsilon_{n-2j} \frac{(n-2j)!(2j)!(2n-2j)!}{[j!(n-j)!]^2} \sin[(n-2j)\theta], \tag{43}$$

where $\lfloor n/2 \rfloor$ means the integer value of $n/2$ and ε_n is the Neumann number ($\varepsilon_0 = 1$ and $\varepsilon_n = 2$ for $n > 0$). At nodes away from the end-points, for $k >$ INT$(3.3 \log_e(n))$ in the $(0, 1)$ interval region, we use the recurrence relation (22) which gives superior results and is computationally less expensive than using the previous expressions (42) and (43). Following this procedure, extensive numerical tests were carried out in the Gauss–Legendre quadrature rule. Then, the results were compared with those obtained with the standard eigenvalue method of Golub and Welsh. Hereafter follows a brief discussion of the findings.

3.3. *Results and Comparisons*

Numerical tests were performed in single and in double precision arithmetic for Gauss–Legendre quadrature rules of order ranging from 2 to 4000 using both techniques, the eigenvalue method implemented in most software libraries and the root-finding method developed in this paper. Two criteria were examined: numerical accuracy and computational efficiency. For the eigenvalue method, single and double precision versions of the code were obtained from the *Netlib* software library. This code is based on the original article of Golub and Welsh [13]. The diagonalization of the Jacobian matrix makes use of the highly efficient implicitly shifted QL-algorithm described in Wilkinson and Reinsch [16]. We were able to optimize the code by a factor of 27% mainly by taking into account the symmetry properties obeyed by the Gauss–Legendre quadrature nodes and weights. It is this optimized code that was used in the tests. In the root-finding method, a first guess of the Gauss–Legendre quadrature nodes was generated with (38) and (39), according to the technique described in Section 3.2. A fifth-order one-iteration scheme derived from (24) and (26) produced the desired nodes. The ordinary Legendre quadrature polynomials were initially evaluated by recurrence (22), and subsequently by a Taylor series expansion which included the first five terms, as indicated in (41). The weight computation proceeded, using next an analytic expression such as (33) including up to six terms with the Legendre polynomials calculated according to the method described in Section 3.2. Simpler, lower order schemes were also tested and showed no significant gain in efficiency. The tables published by Stroud and Secrest [21] which report the nodes and the weights to 30 significant digits served as control values.

Table I and Table II give the results for the node computation of the Gauss–Legendre quadrature rules of order $n = 92$ and $n = 384$, respectively. Table III and Table IV include the corresponding Gauss–Legendre quadrature weights. These are typical results and we have reported in these tables only the nodes and the weights in the interval where their accuracy is most likely to differ in the two methods of computation. In addition, these cases were chosen because of their actual interest to climate and atmospheric forecast modellers.

With regard to quadrature accuracy, it was found that, whatever iterative scheme is used, the root-finding tech-

TABLE I

Gauss–Legendre Quadrature Rule with $n = 92$ Nodes

Root #	Control values for the nodes	Root-Finding method	Eigenvalue method
1	0.9996620713 565245 54	0.9996620713 565245 2	0.9996620713 565250 7
2	0.9982199021 954501 53	0.9982199021 954502 0	0.9982199021 954507 6
3	0.9956270752 692151 69	0.9956270752 692151 2	0.9956270752 692157 9
4	0.9918860044 372534 78	0.9918860044 372535 0	0.9918860044 372533 9
5	0.9870009339 240120 40	0.9870009339 240120 0	0.9870009339 240126 7
6	0.9809774812 101540 27	0.9809774812 101540 5	0.9809774812 101543 0
7	0.9738225880 269419 34	0.9738225880 269418 8	0.9738225880 269419 9
8	0.9655445044 215959 01	0.9655445044 215958 9	0.9655445044 215965 6
9	0.9561527771 340069 46	0.9561527771 340069 3	0.9561527771 340068 1
10	0.9456582378 843629 56	0.9456582378 843629 8	0.9456582378 843628 7
11	0.9340729906 070954 98	0.9340729906 070954 8	0.9340729906 070954 8
12	0.9214103973 710760 54	0.9214103973 710761 1	0.9214103973 710758 8
13	0.9076850629 109250 43	0.9076850629 109249 9	0.9076850629 109252 1
14	0.8929128177 527762 04	0.8929128177 527762 0	0.8929128177 527768 7
15	0.8771106999 395696 69	0.8771106999 395696 9	0.8771106999 395696 9
⋮			
⋮			
30	0.5315106888 865978 78	0.5315106888 865979 2	0.5315106888 865966 9
31	0.5024417347 304565 13	0.5024417347 304565 0	0.5024417347 304559 5
32	0.4727932715 079918 35	0.4727932715 079918 3	0.4727932715 079909 4
33	0.4425994953 295942 31	0.4425994953 295942 5	0.4425994953 295936 4
34	0.4118952312 618817 08	0.4118952312 618817 1	0.4118952312 618815 4
35	0.3807158931 609505 69	0.3807158931 609505 6	0.3807158931 609501 1
36	0.3490974428 265281 83	0.3490974428 265281 6	0.3490974428 265279 9
37	0.3170763485 241381 07	0.3170763485 241381 3	0.3170763485 241384 1
38	0.2846895429 231169 43	0.2846895429 231169 3	0.2846895429 231171 0
39	0.2519743804 989961 80	0.2519743804 989961 8	0.2519743804 989961 8
40	0.2189685944 493801 52	0.2189685944 493801 6	0.2189685944 493803 5
41	0.1857102531 730124 50	0.1857102531 730124 6	0.1857102531 730127 1
42	0.1522377163 622269 96	0.1522377163 622269 9	0.1522377163 622271 9
43	0.1185895907 594259 29	0.1185895907 594259 3	0.1185895907 594260 4
44	0.8480468562 861401 26E-01	0.8480468562 861401 3E-01	0.8480468562 861470 7E-01
45	0.5092196799 334792 34E-01	0.5092196799 334792 3E-01	0.5092196799 334820 1E-01
46	0.1698051769 272823 43E-01	0.1698051769 272823 3E-01	0.1698051769 272897 5E-01

Note. Results for the Gauss–Legendre quadrature nodes computation in double precision floating point IEEE arithmetic. Only nodes close to the end-points and in the middle of the interval of integration are reported as they are where accuracy is most likely to differ in the two methods of computation. The control values in column 2 are from Stroud and Secrest [21] and are given to 18 significant digits. We report the computed values with 17 significant digits. In double precision arithmetic these values should be correct to at most 16 significant digits. The root-finding method produces nodes which are accurate to 16 significant digits, whereas the eigenvalue method gives in general 15 correct decimals and loses accuracy for nodes that are very small.

nique ultimately produces nodes which are accurate to machine precision up to the last digits (e.g., 16 significant digits in double precision arithmetic), as shown in column 3 of Table I and Table II (In fact, we report the results to 17 digits). Column 2 of these tables contains the control values of Stroud and Secrest [21] given here to 18 significant digits. The accuracy of the nodes in the eigenvalue method which are obtained from the diagonalization of a tridiagonal Jacobi matrix depends on the accuracy of the initial matrix elements of that matrix. Fortunately, these matrix elements are known exactly (to machine precision) since they are generated from a known three-term recur-

TABLE II

Gauss–Legendre Quadrature Rule with $n = 384$ Nodes

Root #	Control values for the nodes	Root-Finding method	Eigenvalue method
1	0.9999804411 726473 54	0.9999804411 726473 9	0.9999804411 726473 9
2	0.9998969471 378595 99	0.9998969471 378595 9	0.9998969471 378601 5
3	0.9997467408 113522 75	0.9997467408 113522 9	0.9997467408 113522 9
4	0.9995297988 558858 11	0.9995297988 558858 9	0.9995297988 558852 2
5	0.9992461316 671844 03	0.9992461316 671844 6	0.9992461316 671845 7
6	0.9988957572 063257 57	0.9988957572 063257 3	0.9988957572 063257 3
7	0.9984786985 384589 56	0.9984786985 384589 4	0.9984786985 384590 5
8	0.9979949833 727938 97	0.9979949833 727938 0	0.9979949833 727934 6
9	0.9974446439 389107 95	0.9974446439 389107 5	0.9974446439 389110 8
10	0.9968277169 440913 10	0.9968277169 440913 4	0.9968277169 440912 2
11	0.9961442435 551086 37	0.9961442435 551086 7	0.9961442435 551083 3
12	0.9953942693 885953 36	0.9953942693 885953 2	0.9953942693 885953 2
13	0.9945778445 047067 58	0.9945778445 047067 7	0.9945778445 047066 5
14	0.9936950234 020882 23	0.9936950234 020882 6	0.9936950234 020880 4
15	0.9927458650 133153 93	0.9927458650 133153 0	0.9927458650 133147 4
16	0.9917304327 004320 07	0.9917304327 004320 4	0.9917304327 004317 1
17	0.9906487942 504060 64	0.9906487942 504060 8	0.9906487942 504064 2
18	0.9895010218 704086 78	0.9895010218 704086 7	0.9895010218 704095 6
19	0.9882871921 828698 24	0.9882871921 828698 7	0.9882871921 828705 3
20	0.9870073862 202814 98	0.9870073862 202815 0	0.9870073862 202819 5
⋮			
176	0.1344066496 809674 68	0.1344066496 809674 7	0.1344066496 809668 6
177	0.1263058061 156662 99	0.1263058061 156663 0	0.1263058061 156657 8
178	0.1181965306 016578 36	0.1181965306 016578 4	0.1181965306 016575 7
179	0.1100793644 996070 42	0.1100793644 996070 5	0.1100793644 996069 5
180	0.1019548496 969403 59	0.1019548496 969403 5	0.1019548496 969408 0
181	0.9382352857 167028 51E-01	0.9382352857 167028 4E-01	0.9382352857 167065 9E-01
182	0.8568594395 618718 40E-01	0.8568594395 618718 6E-01	0.8568594395 618715 8E-01
183	0.7754263910 102077 58E-01	0.7754263910 102077 2E-01	0.7754263910 102013 3E-01
184	0.6939415763 857370 51E-01	0.6939415763 857370 6E-01	0.6939415763 857310 9E-01
185	0.6124104354 682961 88E-01	0.6124104354 682961 1E-01	0.6124104354 682893 8E-01
186	0.5308384111 303817 30E-01	0.5308384111 303817 6E-01	0.5308384111 303796 1E-01
187	0.4492309489 737939 46E-01	0.4492309489 737939 7E-01	0.4492309489 737930 7E-01
188	0.3675934969 660982 12E-01	0.3675934969 660981 5E-01	0.3675934969 660987 8E-01
189	0.2859315050 769284 91E-01	0.2859315050 769284 7E-01	0.2859315050 769311 1E-01
190	0.2042504249 141571 44E-01	0.2042504249 141571 1E-01	0.2042504249 141602 7E-01
191	0.1225557093 599553 90E-01	0.1225557093 599553 8E-01	0.1225557093 599569 4E-01
192	0.4085281220 676868 08E-02	0.4085281220 676867 9E-02	0.4085281220 677178 4E-02

Note. Same as in Table I but for Gauss–Legendre quadrature rule with 384 nodes. The increased loss of accuracy for the nodes computed with the eigenvalue method shown in column 4 is consistent for the smallest nodes. No such loss is observed for the nodes computed with the root-finding method, as shown in column 3.

rence relation. Consequently, let us say in double precision arithmetic, the matrix elements are correct to at least 15 significant digits. However, it was found that the computed eigenvalues of the matrix are correct to 15 decimal places but not necessarily to 15 significant digits. This results in a loss of accuracy for very small nodes as can be seen to happen in column 4 of Table I and Table II. For the Gauss–Legendre quadrature, the nodes in the middle of

TABLE III

Results for the Gauss–Legendre Quadrature Weight for $n = 92$

Root #	Control values for the weights	method using Eqn.(15)	New method described in Section 3	Eigenvalue method
1	0.8671851787 671421 35E-03	0.8671851787 705189 E-03	0.8671851787 671409 8E-03	0.8671851787 668861 9E-03
2	0.2017671366 262838 59E-02	0.2017671366 258245 0E-02	0.2017671366 262835 1E-02	0.2017671366 262536 3E-02
3	0.3167535943 396097 87E-02	0.3167535943 399320 4E-02	0.3167535943 396090 4E-02	0.3167535943 395855 8E-02
4	0.4313895331 861700 47E-02	0.4313895331 860717 6E-02	0.4313895331 861701 2E-02	0.4313895331 862189 5E-02
5	0.5455308908 000870 98E-02	0.5455308908 002439 6E-02	0.5455308908 000862 7E-02	0.5455308908 000444 7E-02
6	0.6590439334 214895 22E-02	0.6590439334 214118 2E-02	0.6590439334 214883 2E-02	0.6590439334 215167 7E-02
7	0.7717971837 373568 50E-02	0.7717971837 374776 8E-02	0.7717971837 373541 7E-02	0.7717971837 373972 8E-02
8	0.8836604056 467877 37E-02	0.8836604056 467956 7E-02	0.8836604056 467868 2E-02	0.8836604056 468229 0E-02
9	0.9945045019 726082 04E-02	0.9945045019 726657 4E-02	0.9945045019 726079 7E-02	0.9945045019 725968 7E-02
10	0.1104201592 263594 22E-01	0.1104201592 263481 1E-01	0.1104201592 263538 9E-01	0.1104201592 263474 5E-01
11	0.1212625136 263771 05E-01	0.1212625136 263807 5E-01	0.1212625136 263771 6E-01	0.1212625136 263809 1E-01
12	0.1319650070 571113 80E-01	0.1319650070 571033 2E-01	0.1319650070 571117 5E-01	0.1319650070 571141 1E-01
13	0.1425152948 895392 52E-01	0.1425152948 895467 8E-01	0.1425152948 895389 9E-01	0.1425152948 895304 5E-01
14	0.1529012082 579650 50E-01	0.1529012082 579665 8E-01	0.1529012082 579654 1E-01	0.1529012082 579579 5E-01
15	0.1631107680 025595 80E-01	0.1631107680 025573 7E-01	0.1631107680 025594 9E-01	0.1631107680 025767 7E-01
⋮				
⋮				
30	0.2876796607 210717 58E-01	0.2876796607 210695 8E-01	0.2876796607 210712 1E-01	0.2876796607 210666 7E-01
31	0.2936435364 342281 26E-01	0.2936435364 342284 6E-01	0.2936435364 342281 8E-01	0.2936435364 342238 1E-01
32	0.2992687279 231107 33E-01	0.2992687279 231102 9E-01	0.2992687279 231102 9E-01	0.2992687279 231183 4E-01
33	0.3045487471 715832 09E-01	0.3045487471 715829 4E-01	0.3045487471 715834 6E-01	0.3045487471 715776 0E-01
34	0.3094775042 804103 16E-01	0.3094775042 804101 2E-01	0.3094775042 804101 5E-01	0.3094775042 804073 4E-01
35	0.3140493144 912217 79E-01	0.3140493144 912218 1E-01	0.3140493144 912215 3E-01	0.3140493144 912159 8E-01
36	0.3182589047 432008 58E-01	0.3182589047 432008 6E-01	0.3182589047 432003 7E-01	0.3182589047 432039 1E-01
37	0.3221014197 549332 95E-01	0.3221014197 549325 3E-01	0.3221014197 549328 1E-01	0.3221014197 549351 7E-01
38	0.3255724276 244004 52E-01	0.3255724276 244006 3E-01	0.3255724276 244004 2E-01	0.3255724276 243967 4E-01
39	0.3286679249 406566 03E-01	0.3286679249 406567 6E-01	0.3286679249 406566 2E-01	0.3286679249 406555 8E-01
40	0.3313843414 012938 18E-01	0.3313843414 012936 1E-01	0.3313843414 012936 8E-01	0.3313843414 012911 1E-01
41	0.3337185439 303681 03E-01	0.3337185439 303679 7E-01	0.3337185439 303681 1E-01	0.3337185439 303695 0E-01
42	0.3356678402 920367 63E-01	0.3356678402 920371 1E-01	0.3356678402 920371 1E-01	0.3356678402 920363 5E-01
43	0.3372299821 957387 16E-01	0.3372299821 957389 1E-01	0.3372299821 957389 8E-01	0.3372299821 957349 5E-01
44	0.3384031678 893360 18E-01	0.3384031678 893358 7E-01	0.3384031678 893358 7E-01	0.3384031678 893254 7E-01
45	0.3391860442 372254 94E-01	0.3391860442 372261 1E-01	0.3391860442 372261 1E-01	0.3391860442 372330 4E-01
46	0.3395777082 810234 79E-01	0.3395777082 810229 3E-01	0.3395777082 810229 3E-01	0.3395777082 810132 9E-01

Note. Results for the Gauss–Legendre quadrature weight computation in double precision floating point IEEE arithmetic. The second column contains the control values from Stroud and Secrest [21]. Only the weights which correspond to the nodes reported in Table I are given. The first few weights are the ones which show the largest difference between various methods used for their computation. With nodes accurate to 16 significant digits such as shown in Table I, the weight computation results using (15) are shown in column 3. The eigenvalue results for the weights are given in column 5. The weights computed with the new expression described in Section 3 are reported in column 4. Provided enough correction terms are kept in the expression, these weights are practically independent of large magnitude errors of the order of $1/n^4$ in the nodes, where n is the number of quadrature nodes.

the interval $[-1, 1]$ are significantly smaller than 1, whereas the values of the nodes nearest the end-points approach 1. For relatively large quadrature order, the difference may reach a few orders of magnitude resulting in a significant loss of accuracy for the nodes in the middle of the interval region. Such a loss does not occur in the root-finding method.

Concerning the weight accuracy, the eigenvalue method again guarantees weights which are accurate to 15 decimal places (in double precision arithmetic) but not necessarily

TABLE IV

Results for the Gauss–Legendre Quadrature Weight for $n = 384$

Root #	Control values for the weights	method using Eqn.(15)	New method described in Section 3	Eigenvalue method
1	0.5019410348 692173 75E-04	0.5019410345 2655955E-04	0.5019410348 692176 5E-04	0.5019410348 723959 2E-04
2	0.1168390665 730188 62E-03	0.1168390665 7736422E-03	0.1168390665 730193 0E-03	0.1168390665 724639 6E-03
3	0.1835749193 551260 44E-03	0.1835749193 5198770E-03	0.1835749193 551263 6E-03	0.1835749193 557834 4E-03
4	0.2503070890 844147 24E-03	0.2503070890 8916873E-03	0.2503070890 844152 1E-03	0.2503070890 847744 6E-03
5	0.3170242698 112706 30E-03	0.3170242698 1897338E-03	0.3170242698 112708 8E-03	0.3170242698 107591 9E-03
6	0.3837208020 912924 38E-03	0.3837208020 9531496E-03	0.3837208020 912910 0E-03	0.3837208020 912185 2E-03
7	0.4503919137 716877 61E-03	0.4503919137 7272786E-03	0.4503919137 716872 4E-03	0.4503919137 705995 2E-03
8	0.5170330453 491546 38E-03	0.5170330453 4946840E-03	0.5170330453 491544 1E-03	0.5170330453 503842 2E-03
9	0.5836397042 629966 77E-03	0.5836397042 6729027E-03	0.5836397042 629978 0E-03	0.5836397042 625606 5E-03
10	0.6502074240 969915 51E-03	0.6502074240 9489622E-03	0.6502074240 969918 8E-03	0.6502074240 973900 0E-03
11	0.7167317509 947397 54E-03	0.7167317509 9278257E-03	0.7167317509 947394 5E-03	0.7167317509 944481 3E-03
12	0.7832082385 905052 31E-03	0.7832082385 9153669E-03	0.7832082385 905056 2E-03	0.7832082385 904226 7E-03
13	0.8496324460 039079 10E-03	0.8496324460 0321777E-03	0.8496324460 039092 7E-03	0.8496324460 033415 9E-03
14	0.9159999370 632673 68E-03	0.9159999370 6118388E-03	0.9159999370 632667 4E-03	0.9159999370 633531 5E-03
15	0.9823062800 663751 21E-03	0.9823062800 6604854E-03	0.9823062800 663738 0E-03	0.9823062800 665730 8E-03
16	0.1048547047 793687 52E-02	0.1048547047 7921493E-02	0.1048547047 793686 0E-02	0.1048547047 793774 5E-02
17	0.1114717817 647312 65E-02	0.1114717817 6463886E-02	0.1114717817 647309 3E-02	0.1114717817 647248 0E-02
18	0.1180814171 855886 46E-02	0.1180814171 8561737E-02	0.1180814171 855892 2E-02	0.1180814171 855312 8E-02
19	0.1246831697 715436 93E-02	0.1246831697 7137758E-02	0.1246831697 715438 5E-02	0.1246831697 715960 2E-02
20	0.1312765987 850601 06E-02	0.1312765987 8504835E-02	0.1312765987 850603 6E-02	0.1312765987 850774 5E-02
⋮				
⋮				
176	0.8096447276 312187 23E-02	0.8096447276 3121933E-02	0.8096447276 312191 5E-02	0.8096447276 311681 5E-02
177	0.8105149720 727896 70E-02	0.8105149720 7278965E-02	0.8105149720 727896 5E-02	0.8105149720 727190 5E-02
178	0.8113311079 909191 51E-02	0.8113311079 9091954E-02	0.8113311079 909195 4E-02	0.8113311079 909695 0E-02
179	0.8120930809 018394 51E-02	0.8120930809 0183785E-02	0.8120930809 018383 7E-02	0.8120930809 018002 1E-02
180	0.8128008399 376079 12E-02	0.8128008399 3760920E-02	0.8128008399 376088 6E-02	0.8128008399 376123 3E-02
181	0.8134543378 495027 88E-02	0.8134543378 4950488E-02	0.8134543378 495048 8E-02	0.8134543378 495170 3E-02
182	0.8140535310 111774 80E-02	0.8140535310 1117583E-02	0.8140535310 111758 3E-02	0.8140535310 112523 3E-02
183	0.8145983794 215729 60E-02	0.8145983794 2157495E-02	0.8145983794 215746 0E-02	0.8145983794 216115 5E-02
184	0.8150888467 075881 62E-02	0.8150888467 0758470E-02	0.8150888467 075848 7E-02	0.8150888467 076473 2E-02
185	0.8155249001 265081 92E-02	0.8155249001 2650938E-02	0.8155249001 265090 3E-02	0.8155249001 264909 9E-02
186	0.8159065105 681901 67E-02	0.8159065105 6818991E-02	0.8159065105 681900 8E-02	0.8159065105 681706 5E-02
187	0.8162336525 570065 60E-02	0.8162336525 5700901E-02	0.8162336525 570091 9E-02	0.8162336525 570038 1E-02
188	0.8165063042 535459 03E-02	0.8165063042 5354788E-02	0.8165063042 535478 8E-02	0.8165063042 535799 7E-02
189	0.8167244474 560707 52E-02	0.8167244474 5606865E-02	0.8167244474 560689 9E-02	0.8167244474 561583 3E-02
190	0.8168880676 017327 98E-02	0.8168880676 0173288E-02	0.8168880676 017328 8E-02	0.8168880676 016775 4E-02
191	0.8169971537 675450 57E-02	0.8169971537 6754595E-02	0.8169971537 675459 5E-02	0.8169971537 676169 0E-02
192	0.8170516986 711110 73E-02	0.8170516986 7111165E-02	0.8170516986 711114 8E-02	0.8170516986 710861 5E-02

Note. Same as in Table III but for Gauss–Legendre quadrature rule with $n = 384$ nodes. In addition to giving superior results for the weights, the proposed method is more efficient than the eigenvalue method by a factor of at least 5.

to 15 significant digits. This is a real disadvantage because, as the quadrature order gets larger, the Gauss–Legendre quadrature weights get smaller resulting in a loss of a larger number of significant digits. This result defeats, in a sense, the whole purpose of using quadrature rules of very high order to compute integrals more accurately. The problem in addition is more severe for weights nearest the endpoints of the interval. This peculiar loss of accuracy reflected in the loss of significant digits for the Gauss–Legendre weights can be seen in column 4 of Table III

and Table IV. One may observe that the loss of accuracy is larger for the first few weights nearest the end-points. In the Gauss–Legendre quadrature rule, if x is the sine of the latitude, then one can show that the weights are proportional to the cosine of that latitude. Hence, the peculiar behavior of the nodes and the weights in the regions mentioned above. Using an analytical expression such as the one given in (15) for computing the Gauss–Legendre weights leads to an even more serious problem of accuracy, especially for small weights nearest the end-points. The reason for such a problem is depicted in Fig. 1 for a quadrature rule of very low order. Column 3 in Table III and in Table IV reproduces the weights computed in double precision arithmetic using (15) when the nodes are accurate to 16 significant digits such as shown in column 3 of Table I and of Table II. It can be seen that the accuracy of the weights nearest the end-points is unacceptably low. In fact, we have tested the consequences of using weights calculated from (15) in a high resolution atmospheric spectral forecast model similar to the one described in Ritchie *et al.* [12] and found a measurable deterioration in the five-day forecast results.

Using nodes accurate to 16 significant digits (column 3 of Table I and Table II) and retaining only the first two terms in (33) is sufficient to produce weights with the accuracy shown in column 4 of Table III and of Table IV. These results are general and clearly demonstrate the superiority of the proposed technique compared to the eigenvalue method whose results appear in column 5 of Table III and Table IV. For $n = 92$ in Table III, the first weight nearest the end-point has 14 correct significant digits when computed with (33), as shown in column 4, compared to 12 in the eigenvalue method (column 5). Whereas expression (15) guarantees only 11 significant digits for the weights (column 3). For a Gauss–Legendre quadrature rule of order $n = 384$ given in Table IV the differences in the weight accuracy get even larger. The expression (33) gives 15 significant digits for the first weight nearest the end-point, whereas the eigenvalue method produces only 11 correct digits. The weight computed with (15) further deteriorates and barely gives nine correct significant digits.

There is another advantage of using analytic expressions such as (33) or other analogous expressions such as (36). It is that these expressions provide accurate weights even though the nodes may carry a relatively large error. When five or six terms are included in the analytic expression (33) (i.e., terms proportional to δ_0 and up to δ_5), then the weights are not sensitive to perturbations in the nodes of the order of $1/n^4$, where n is the number of quadrature nodes. This robustness of the weight expression in the neighborhood of the nodes is due to the peculiar behavior of the weight function such as the one depicted in Fig. 7. Even a very small error in the nodes has a devastating result on the accuracy of the weights given by (15). Similarly, a small error in the initial elements of the Jacobi matrix has a direct impact on the accuracy of the weights generated in the eigenvalue method. Similar tests were also performed in single precision arithmetic and small perturbations in the nodes had an analogous impact on the weight precision.

As far as the efficiency of the nodes and of the weight computation is concerned, one must admit that the methods examined in this paper are all very efficient. However, it was found that the root-finding method implemented was at least five times more efficient than the optimized version of the *Netlib* eigenvalue method, as described previously. This is clear, especially for very high quadrature order. The results are shown in Table V for various Gauss–Legendre quadrature rules of order up to 8000. It is seen that, for very high order quadrature rules, the efficiency of the eigenvalue method deteriorates markedly from the root-finding method. The timings reported in column 2 of Table V are for the comptutation of the nodes and of the weights. The iterative scheme for computing the nodes included up to five terms in (24) with one iteration where the new values for the Legendre polynomials was computed with a Taylor series expansion with five terms. For the weight computation, we used an expression such as (33) with six terms included in the series. The computing cost for adding these higher order terms in the computation of the nodes and of the weights is minimal compared to the cost of evaluating the ordinary Legendre polynomials by recursion (22) or by using expressions such as (42) and (43). For particular applications, one can easily tailor an iteration scheme and an analytic weight function that com-

TABLE V

CPU Time (in seconds) for Gauss–Legendre Quadrature

Total number of nodes	CPU time (in seconds)	
	Root-finding method	Eigenvalue method
50	0.01	0.01
100	0.01	0.03
200	0.03	0.10
500	0.12	0.58
1000	0.42	2.26
2000	1.59	8.94
4000	6.19	34.68
6000	13.80	76.65
8000	24.41	133.62

Note. Comparison of the computing cost (in seconds) for the evaluation of Gauss–Legendre quadrature nodes and weights of order n following the proposed method in this paper and the Golub–Welsh eigenvalue method. The computation was carried in double precision floating point IEEE arithmetic using a SGI Power Challenge computer. The timing for the eigenvalue method takes into account a code optimization mentioned in Section 3.3.

bine all the qualities of accuracy and efficiency, as well as simplicity.

The method developed in this paper was also applied and tested for the classical Gauss–Christoffel quadrature rules described in the Appendix. Very similar conclusions were reached concerning the accuracy and the efficiency of this method. Some results were also reported in Yakimiw [11].

4. CONCLUSION

A simple, highly accurate and efficient method for computing the nodes and the weights in the classical Gauss–Christoffel quadrature rules has been presented. The computation relies on using a root-finding technique for the nodes and analytic expressions for the weights. The root-finding technique itself is based on iterative schemes exhibiting superlinear convergence rates and involving solely the quadrature polynomials and their first derivatives. Contrary to previous analytic expressions widely used for computing the quadrature weights, the new expressions are more robust, resulting in a significant improvement in accuracy for the weights, even though the nodes may carry a relatively large error. These new expressions need only the accurate evaluation of the quadrature polynomials and their first derivatives.

Extensive tests were performed in single and double precision IEEE arithmetic in order to assess the method in terms of accuracy and efficiency. The results were compared with the eigenvalue method proposed in 1969 by Golub and Welsh [13] and implemented in most software libraries. We obtained both single and double versions of the Golub–Welsh code from the *Netlib* software library. The code was further optimized by a factor of 27% in the case of the Gauss–Legendre quadrature rule mainly by taking into account the symmetry properties obeyed by the nodes and the weights.

In terms of accuracy for the nodes and, most important, for the weights, the tests indicate that the method developed in this paper gives consistently superior results, compared to the eigenvalue method. This is especially true for very high order quadrature rules. This can be seen, for instance, in Table I and Table II for the nodes and in Table III and Table IV for the weights. In the eigenvalue method with double precision arithmetic, the resulting eigenvalues, and thus the nodes, are given with 15 correct decimal places but not necessarily with 15 accurate significant digits. This is true, provided the initial elements of the Jacobi matrix are known to at least 15 significant digits. The same rule applies for the quadrature weights. This loss of accuracy in the nodes and in the weights increases with the quadrature order since the nodes and the weights become increasingly small in certain regions of the interval of integration. This is a real disadvantage since it defeats the whole purpose of using higher order quadrature rules to compute integrals more accurately.

In addition to giving superior results, the proposed method turns out to be more efficient than the eigenvalue method by a factor of at least 5. This can be seen in Table V for the Gauss–Legendre quadrature rule, especially for very high order quadrature rules. For these reasons, for its accuracy, its efficiency and its simplicity, the method developed for the Gauss–Legendre quadrature rule has been implemented in the Canadian climate spectral model and the operational high resolution atmospheric spectral forecast model. A version of the Gauss–Legendre quadrature code is available on request from the author at eyakimiw@cmc.doe.ca.

APPENDIX A: EFFICIENT AND ACCURATE FORMULAS FOR COMPUTING THE NODES AND THE WEIGHTS IN CLASSICAL GAUSS–CHRISTOFFEL QUADRATURE RULES

The general method described in this paper is applied to various Gauss–Christoffel quadrature rules, where the quadrature polynomials are known to obey a second-order differential equation. Superlinear converging iterative schemes for computing the nodes are given as well as high order analytic expressions for the Gaussian weights which are practically independent of the rounding-error in the nodes and involve solely the quadrature polynomials and their first derivatives. The following rules are examined, Lobatto, Radau, Hermite, and Laguerre. Similar formulas for other related Gauss–Christoffel quadrature rules can easily be derived.

A.1. *Gauss–Lobatto Quadrature Rule*

The Gauss–Lobatto quadrature rule which includes the two end-points of the interval of integration is given as

$$\int_{-1}^{1} F(x)\,dx = \frac{2}{n(n-1)}[F(-1) + F(1)] + \sum_{i=2}^{n-1} w(x_i)F(x_i),$$

where the quadrature nodes x_i are the zeros of the first derivative $P'_{n-1}(x)$ of the ordinary Legendre polynomials of degree $n-1$. For the Lobatto quadrature polynomials, one can use either $P'_{n-1}(x)$ or $(1-x^2)P'_{n-1}(x)$ whose zeros include the end-points. In practice, the weights at the end-points are known exactly and need not to be computed through analytic expressions. However, both choices are examined as possible candidates for deriving high order iterative schemes and robust analytical expressions for the weights.

A.1.1. *Using $P'_{n-1}(x)$ in Gauss–Lobatto Quadrature Rule.* Let $f(x) = P'_{n-1}$ so that $f'(x) = P^{(2)}_{n-1}(x)$. Then, ac-

cording to the convention adopted in Section 2.2,

$$f_0 = \frac{P'_{n-1}}{P^{(2)}_{n-1}},$$
$$f_2 = \frac{4x}{c^2} + \frac{2-\overline{n}}{c^2} f_0 \equiv a_2 + b_2 f_0, \tag{A.1.1}$$

where $c^2 = 1 - x^2$ and $\overline{n} = n(n-1)$. Higher order derivatives of $f(x)$ obey the following relations:

$$f_j = \frac{2jx}{c^2} f_{j-1} + \frac{j(j-1)-\overline{n}}{c^2} f_{j-2}, \quad j = 2, 3, ... \tag{A.1.2}$$

Let us rewrite (13) as

$$Dx = -c^2 \sum_{j=1}^{\infty} \frac{\delta_j}{j!} \left(\frac{f_0}{c^2}\right)^j. \tag{A.1.3}$$

Since

$$a_2 = \frac{4x}{c^2}, \quad b_2 = \frac{2-\overline{n}}{c^2},$$

and

$$d_j = \frac{\delta_j}{c^{2(j-1)}}, \quad d'_j = \frac{\delta'_j}{c^{2(j-1)}} + \frac{2(j-1)x}{c^{2j}} \delta_j,$$

the substitution of d_j and d'_j in (14) leads to the following recurrence relation for the coefficients δ_j in (A.1.3):

$$\delta_1 = 1$$
$$\delta_{j+1} = -c^2 \delta'_j + 2(j+1)x\delta_j + j(j-1)(2-\overline{n})c^2 \delta_{j-1},$$
$$j = 1, 2, ... \tag{A.1.4}$$

In particular, we have

$$\delta_1 = 1$$
$$\delta_2 = 4x$$
$$\delta_3 = 2(12x^2 - \overline{n}c^2)$$
$$\delta_4 = 4x(48x^2 - 11\overline{n}c^2)$$
$$\delta_5 = 4[480x^4 - \overline{n}c^2(203x^2 + 1 - 6\overline{n}c^2)], \tag{A.1.5}$$

where $c^2 = 1 - x^2$ and $\overline{n} = n(n-1)$.

Similarly, the general Gaussian weight formula (18) is written as

$$w(x) = 2\overline{n} \Big/ \left[c^2 f' \sum_{j=0}^{\infty} \frac{\delta_j}{j!} \left(\frac{f_0}{c^2}\right)^j \right]^2. \tag{A.1.6}$$

From (20), the recurrence relation holds for δ_j in (A.1.6):

$$\delta_0 = 1$$
$$\delta_{j+1} = -c^2 \delta'_j + 2(j-1)x\delta_j + j(j-2)(2-\overline{n})c^2 \delta_{j-1},$$
$$j = 0, 1, 2, ... \tag{A.1.7}$$

The first terms in (A.1.6) are

$$\delta_0 = 1$$
$$\delta_1 = -2x$$
$$\delta_2 = \overline{n}c^2$$
$$\delta_3 = 4\overline{n}xc^2$$
$$\delta_4 = \overline{n}c^2[24 - (22 + 3\overline{n})c^2]$$
$$\delta_5 = 2\overline{n}xc^2[96 - (78 + 31\overline{n})c^2], \tag{A.1.8}$$

where again we have defined $c^2 = 1 - x^2$ and $\overline{n} = n(n-1)$.

A.1.2. *Using $c^2 P'_{n-1}(x)$ in Gauss–Lobatto Quadrature Rule.* If we choose

$$f(x) = c^2 P'_{n-1}(x) \tag{A.1.9}$$

as the Gauss–Lobatto quadrature polynomials, where $c^2 = 1 - x^2$, then

$$f'(x) = -\overline{n} P_{n-1}(x), \tag{A.1.10}$$

where $\overline{n} = n(n-1)$ and

$$f_2 = -\frac{\overline{n}}{c^2} f_0. \tag{A.1.11}$$

Hence, $a_2 = 0$ and $b_2 = -\overline{n}/c^2$. In general, the higher derivatives in this case are

$$f_j = \frac{2(j-2)x}{c^2} f_{j-1} + \frac{(j-2)(j-3)-\overline{n}}{c^2} f_{j-2},$$
$$j = 2, 3, ... \tag{A.1.12}$$

From (13), recall that

$$Dx = -\sum_{j-1}^{\infty} \frac{d_j}{j!} f_0^j. \tag{A.1.13}$$

Then (14) becomes

$$d_1 = 1$$
$$d_{j+1} = -d'_j - j(j-1)\frac{\overline{n}}{c^2} d_{j-1}, \quad j = 1, 2, ... \tag{A.1.14}$$

In particular, the first five d_j's in (A.1.13) are

$$\begin{aligned} d_1 &= 1 \\ d_2 &= 0 \\ d_3 &= -\frac{2\bar{n}}{c^2} \\ d_4 &= \frac{4\bar{n}x}{c^4} \\ d_5 &= -\frac{4\bar{n}}{c^6}(1+3x^2-6\bar{n}c^2), \end{aligned} \tag{A.1.15}$$

where $c^2 = 1 - x^2$ and $\bar{n} = n(n-1)$. One can see that the resulting expression for Dx in (A.1.13) with the coefficients given by (A.1.15) is somehow simpler than those in (A.1.5) and, thus, is easier to compute. Indeed, after some simplifications, the expression leads to a rapidly converging series:

$$\begin{aligned} Dx = \frac{c^2F}{\bar{n}}\Bigg\{1 - \frac{c^2F^2}{3\bar{n}}\Bigg[1 + \frac{x}{2\bar{n}}F \\ + \frac{(1+3x^2-6\bar{n}c^2)}{10\bar{n}^2}F^2 + \cdots\Bigg]\Bigg\}, \end{aligned} \tag{A.1.16}$$

where $F \equiv P'_{n-1}/P_{n-1}$, $c^2 = 1 - x^2$, and $\bar{n} = n(n-1)$. Using the same polynomials defined in (A.1.9) in the weight formula (18), we have

$$w(x) = 2\bar{n}\Bigg/\Bigg[-\bar{n}P_{n-1}\sum_{j=0}^{\infty}\frac{d_j}{j!}f_0^j\Bigg]^2, \tag{A.1.17}$$

where

$$\begin{aligned} d_0 &= 1 \\ d_{j+1} &= -d'_j - j(j-2)\frac{\bar{n}}{c^2}d_{j-1}, \quad j = 0, 1, 2, ..., \end{aligned} \tag{A.1.18}$$

so that

$$\begin{aligned} d_0 &= 1 \\ d_1 &= 0 \\ d_2 &= \frac{\bar{n}}{c^2} \\ d_3 &= -\frac{2\bar{n}x}{c^4} \\ d_4 &= \frac{2\bar{n}}{c^6}\left(1 + 3x^2 - \frac{3}{2}\bar{n}c^2\right) \\ d_5 &= -\frac{24\bar{n}x}{c^8}\left(1 + x^2 - \frac{7}{6}\bar{n}c^2\right), \end{aligned} \tag{A.1.19}$$

where $c^2 = 1 - x^2$ and $\bar{n} = n(n-1)$, as before. Specifically, we obtain the expression for the Gauss–Lobatto weight function,

$$\begin{aligned} w(x) = 2\bar{n}\Bigg/\Bigg\{\bar{n}P_{n-1}\Bigg[1 + \frac{c^2F^2}{\bar{n}}\Bigg[\frac{1}{2} + \frac{x}{3\bar{n}}F \\ + \frac{(1+3x^2-\frac{3}{2}\bar{n}c^2)}{12\bar{n}^2}F^2 \\ + \frac{x(1+x^2-\frac{7}{6}\bar{n}c^2}{5\bar{n}^3}F^3 + \cdots\Bigg]\Bigg]\Bigg\}^2, \end{aligned} \tag{A.1.20}$$

where $F = P'_{n-1}/P_{n-1}$, $c^2 = 1 - x^2$, and $\bar{n} = n(n-1)$.

A.2. *Gauss–Radau Quadrature Rule*

For the Gauss–Radau quadrature rule, we have

$$\int_{-1}^{1} F(x)\,dx = \frac{2}{n^2}F(-1) + \sum_{i=1}^{n-1} w(x_i)F(x_i),$$

where the Radau quadrature polynomials are defined as

$$f = \frac{P_{n-1} + P_n}{1+x}. \tag{A.2.1}$$

Here, P_n and P_{n-1} are the ordinary Legendre polynomials defined in (22). One can show that

$$c^2f' = nP_{n-1} - nP_n + (x-1)f \tag{A.2.2}$$

and that the higher derivatives of $f(x)$ obey the relation

$$\begin{aligned} f_j &= \frac{(2j-1)x}{c^2}f_{j-1} + \frac{(j-1)^2 - n^2}{c^2}f_{j-2}, \\ j &= 2, 3, ..., \end{aligned} \tag{A.2.3}$$

according to the convention defined in Section 2.2. In particular, we have

$$f_2 = \frac{3x-1}{c^2} + \frac{1-n^2}{c^2}f_0 = a_2 + b_2f_0. \tag{A.2.4}$$

Given the expression (A.2.3) for the higher derivatives, it is relatively simple to use (4) and (6) to derive superlinear converging iterative schemes for Dx. Instead, use (13) and write

$$Dx = -c^2\sum_{j=1}^{\infty}\frac{\delta_j}{j!}\left(\frac{f_0}{c^2}\right)^j, \tag{A.2.5}$$

where $c^2 = 1 - x^2$. From (14), we obtain the following relations for the δ_j's in (A.2.5).

$$\begin{aligned}\delta_1 &= 1\\ \delta_{j+1} &= -c^2\delta_j' + [(j+2)x - j]\delta_j\\ &\quad + j(j-1)(1-n^2)c^2\delta_{j-1},\\ j &= 1, 2, \ldots.\end{aligned} \tag{A.2.6}$$

In particular,

$$\begin{aligned}\delta_1 &= 1\\ \delta_2 &= (3x-1)\\ \delta_3 &= (13x^2 - 10x + 1) - 2n^2c^2\\ \delta_4 &= (73x^3 + 93x^2 + 27x + 1) - 4n^2c^2(8x-3)\\ \delta_5 &= (501x^4 + 572x^3 - 258x^2 - 408x - 19)\\ &\quad -4n^2c^2[(21x-13)(5x-1) - 6n^2c^2].\end{aligned} \tag{A.2.7}$$

For the Radau weight formula, we write (18) as

$$w(x) = 4\Big/\left[\frac{c^2}{\sqrt{1-x}}f'\sum_{j=0}^{\infty}\frac{\delta_j}{j!}\left(\frac{f_0}{c^2}\right)^j\right]^2, \tag{A.2.8}$$

where

$$\begin{aligned}\delta_0 &= 1\\ \delta_{j+1} &= -c^2\delta_j' + \tfrac{1}{2}[(2j-3)x - (2j-1)]\delta_j\\ &\quad + j(j-2)(1-n^2)c^2\delta_{j-1},\\ j &= 0, 1, 2, \ldots.\end{aligned} \tag{A.2.9}$$

In particular, the first five terms in $w(x)$ of (A.2.8) are

$$\begin{aligned}\delta_0 &= 1\\ \delta_1 &= \frac{1-3x}{2}\\ \delta_2 &= \tfrac{1}{4}(x+1)^2 + n^2c^2\\ \delta_3 &= \frac{(x+1)^2}{8}(5x-7) + \frac{n^2c^2}{2}(5x-3)\\ \delta_4 &= \frac{(x+1)^2}{16}(11x-13)(3x-5)\\ &\quad + \frac{n^2c^2}{2}[(5x-7)(3x-1) - 6n^2c^2]\\ \delta_5 &= -c^2\delta_4' + \tfrac{1}{2}(5x-7)\delta_4 + 8(1-n^2)c^2\delta_3,\end{aligned} \tag{A.2.10}$$

where

$$\delta_4' = \frac{3(x+1)}{4}(11x^2 - 18x + 3) - n^2(30x^3 - 39x^2 - 8x + 13 - 12n^2c^2x)$$

and $c^2 = 1 - x^2$. Using the properties of the ordinary Legendre polynomials, the first three terms in $w(x)$ give the expression

$$\begin{aligned}w(x) = 16(1-x)\Big/\Bigg\{&(2n-1)P_{n-1} - (2n+1)P_n\\ &+\left[\frac{(2n+1)(2n-1)}{4}\right.\\ &\left.+\frac{1}{2(1-x)}\right]\frac{f^2}{f'} + \cdots\Bigg\}^2,\end{aligned} \tag{A.2.11}$$

where f and f' are given according to (A.2.1) and (A.2.2).

A.3. *Gauss–Hermite Quadrature Rule*

The Gauss–Hermite quadrature rule is defined as

$$\int_{-\infty}^{\infty} e^{-x^2}F(x)\,dx = \sum_{i=1}^{n} w(x_i)F(x_i).$$

The quadrature polynomials are the Hermite polynomials $H_n(x)$ which obey the relations:

$$\begin{aligned}H_0(x) &= 1, \quad H_1(x) = 2x\\ H_{n+1}(x) &= 2xH_n(x) - 2nH_{n-1}(x)\\ H_n'(x) &= 2nH_{n-1}(x)\\ H_n^{(2)}(x) &= 2xH_n'(x) - 2nH_n(x).\end{aligned} \tag{A.3.1}$$

Using the convention adopted in Section 2.2, we have

$$f = H_n \tag{A.3.2}$$

$$\begin{aligned}f_2 &= 2x - 2nf_0\\ &= a_2 + b_2f_0\end{aligned} \tag{A.3.3}$$

and, in general,

$$f_j = 2xf_{j-1} - 2(n-j+2)f_{j-2}, \quad j = 2, 3, \ldots. \tag{A.3.4}$$

Using (A.3.3), the recurrence relation (14) becomes

$$\begin{aligned}d_1 &= 1\\ d_{j+1} &= -d_j' + 2jxd_j - 2j(j-1)nd_{j-1},\\ j &= 1, 2, \ldots,\end{aligned} \tag{A.3.5}$$

and the first five terms of (13), for the Gauss–Hermite quadrature rule, are

$$\begin{aligned} d_1 &= 1 \\ d_2 &= 2x \\ d_3 &= 2[4x^2 - (2n+1)] \\ d_4 &= 4x(12x^2 - 12n - 7) \\ d_5 &= 4[96x^4 - 4x^2(23 + 36n) + 24n(n+1) + 7]. \end{aligned} \tag{A.3.6}$$

Similarly, for the Hermite weight formula, we get from (20)

$$\begin{aligned} d_0 &= 1 \\ d_{j+1} &= -d_j' + 2(j-1)xd_j - 2j(j-2)nd_{j-1}, \\ j &= 0, 1, 2, ..., \end{aligned} \tag{A.3.7}$$

where the d_j's are the coefficients in the series (18). In particular, the first five terms for the Hermite weight function (18) are

$$\begin{aligned} d_0 &= 1 \\ d_1 &= -2x \\ d_2 &= 2(n+1) \\ d_3 &= 4x(n+1) = -d_1 d_2 \\ d_4 &= 4(n+1)(4x^2 - 3n - 1) \\ &= -2d_2(1+3n) + 4xd_3 \\ d_5 &= 8x(n+1)(12x^2 - 17n - 7) \\ &= 2[x(3d_4 - 4d_2) - 2d_3(1+4n)] \end{aligned} \tag{A.3.8}$$

and the constant k is equal to $2^{n+1}n!\sqrt{\pi}$. More specifically, the Gauss–Hermite formula for the weights has the form

$$\begin{aligned} w(x) = 2^{n+1}n!\sqrt{\pi} \Bigg/ \Bigg\{ & H_{n+1} - \frac{n+1}{2n}\frac{H_n^2}{H_{n-1}} \\ & - \frac{(n+1)x}{6n^2}\frac{H_n^3}{H_{n-1}^2} \\ & - \frac{(n+1)(4x^2 - 3n - 1)}{48n^3}\frac{H_n^4}{H_{n-1}^3} \\ & + \cdots \Bigg\}^2. \end{aligned} \tag{A.3.9}$$

A.4. *Gauss–Laguerre Quadrature Rule*

The Gauss–Laguerre quadrature rule is defined as

$$\int_0^\infty e^{-x}F(x)\,dx = \sum_{i=1}^n w(x_i)F(x_i),$$

where the quadrature nodes x_i are the zeros of the Laguerre polynomials $L_n(x)$ satisfying the relations

$$\begin{aligned} L_0(x) &= 1, \quad L_1(x) = x \\ (n+1)L_{n+1}(x) &= (2n+1-x)L_n(x) - nL_{n-1}(x) \\ xL_n'(x) &= nL_n(x) - nL_{n-1}(x) \\ xL_n^{(2)}(x) &= (x-1)L_n'(x) - nL_n(x). \end{aligned} \tag{A.4.1}$$

Using the convention adopted in Section 2.2, we have

$$f = L_n \tag{A.4.2}$$

$$\begin{aligned} f_2 &= \frac{x-1}{x} - \frac{n}{x}f_0 \\ &= a_2 + b_2 f_0, \end{aligned} \tag{A.4.3}$$

and, in general,

$$\begin{aligned} f_j &= \frac{x+1-j}{x}f_{j-1} - \frac{n+2-j}{x}f_{j-2}, \\ j &= 2, 3, ... \end{aligned} \tag{A.4.4}$$

We also write (13) in the form

$$Dx = -\sum_{j=1}^\infty x\frac{\delta_j}{j!}\left[\frac{f_0}{x}\right]^j. \tag{A.4.5}$$

Using (A.4.3), the recurrence relation (14) translates into

$$\begin{aligned} \delta_1 &= 1 \\ \delta_{j+1} &= -x\delta_j' + (jx-1)\delta_j - j(j-1)nx\delta_{j-1}, \\ j &= 1, 2, ... \end{aligned} \tag{A.4.6}$$

In particular, the first five coefficients in the series (A.4.5) are

$$\begin{aligned} \delta_1 &= 1 \\ \delta_2 &= x - 1 \\ \delta_3 &= 2x^2 - 2x(n+2) + 1 \\ \delta_4 &= 6x^3 - 6x^2(2n+3) + x(10n+11) - 1 \\ \delta_5 &= 24x^4 - 24x^3(3n+4) \\ &\quad + 2x^2(12n^2 + 62n + 49) \\ &\quad - 2x(16n+13) + 1. \end{aligned} \tag{A.4.7}$$

The Laguerre weight formula is given by

$$w(x) = 1 \Bigg/ \left[f' \sum_{j=0}^\infty \frac{\delta_j}{j!}\frac{f_0^j}{\sqrt{x}}\right]^2 \tag{A.4.8}$$

and, using (20), where $d_j = \delta_j \sqrt{x}$ and (A.4.3), we obtain the general relation for the δ_j in (A.4.8),

$$\begin{aligned}\delta_0 &= x \\ \delta_{j+1} &= -\delta_j' + \left[(j-1) - \frac{2j-3}{2x}\right]\delta_j \\ &\quad - j(j-2)\frac{n}{x}\delta_{j-1}, \quad j = 0, 1, 2, ...\end{aligned} \tag{A.4.9}$$

In particular,

$$\begin{aligned}\delta_0 &= x \\ \delta_1 &= \frac{1}{2}(1-2x) \\ \delta_2 &= \frac{1}{2}\left(\frac{1}{2x} + 2n + 1\right) \\ \delta_3 &= \frac{1}{2}\left(\frac{1}{4x^2} - \frac{n}{x} + 2n + 1\right] \\ \delta_4 &= \frac{1}{4}\left[\frac{1}{4x^3} + \frac{(1-2n)}{x^2} - \frac{[3+4n(3n+4)]}{x}\right. \\ &\quad \left. + 4(2n+1)\right. \\ \delta_5 &= \frac{1}{4}\left[\frac{1}{8x^4} + \frac{(1-12n)}{4x^3} + \frac{(68n^2+36n+15)}{2x^2}\right. \\ &\quad \left. - \frac{(68n^2+84n+19)}{x} + 12(2n+1)\right].\end{aligned} \tag{A.4.10}$$

Including the first three terms in $w(x)$ gives the Gauss–Laguerre weight formula

$$w(x) = x \Big/ \left\{ xL_n' + \frac{1}{2}(1-2x)L_n + \frac{1}{4}\left(2n+1+\frac{1}{2x}\right)\frac{L_n^2}{L_n'} + \cdots \right\}^2, \tag{A.4.11}$$

ACKNOWLEDGMENTS

It is an honor for the author to thank the Academician Sergei K. Godunov from the Sobolev Institute of Mathematics in Novosibirsk for his interest in this work, his comments, and his encouragement during his short visit at RPN in the Fall of 1995. The author gratefully acknowledges the assistance of Steve Thomas and Vivian Lee in the preparation of the manuscript.

REFERENCES

1. F. B. Hildebrand, *Introduction to Numerical Analysis,* 2nd ed. (McGraw–Hill, New York, 1974).
2. W. Gautschi, "A Survey of Gauss–Christoffel Quadrature Formulae," in *E. B. Christoffel. The Influence of His Work in Mathematics and the Physical Sciences,* edited by P. L. Butzer and F. Feher (Birkhaüser, Basel, 1981), p. 72.
3. P. Davis and Rabinowitz P, *Methods of Numerical Integration* 2nd ed. (Academic Press, San Diego, 1984).
4. W. Gautschi, *Math. Comput.* **57,** 309 (1991).
5. G. V. Milovanovic, "Summation of Series and Gaussian Quadrature," in *International Series of Numerical Mathematics,* Vol. 119, edited by R. V. M. Zahar (Birkhaüser, Basel, 1994), p. 459.
6. G. Mastroianni and G. Monegato, "Error Estimates for Gauss–Laguerre and Gauss–Hermite Quadrature Formulas," in *E. B. Christoffel. The Influence of His Work in Mathematics and the Physical Sciences,* edited by P. L. Butzer and F. Feher (Birkhaüser, Basel, 1981), p. 421.
7. W. Gautschi and S. Li, *J. Comput. Appl. Math.* **34,** 343 (1991).
8. W. Gautschi, *Math. Comput.* **22,** 251 (1968).
9. F. G. Lether, *J. Comput. Appl. Math.* **4,** 47 (1978).
10. T. Nehrkorn, *Mon. Weather Rev.* **118,** 2248 (1990).
11. E. Yakimiw, "On the Accurate Computation of the Weights in Gaussian Quadrature," in *Proceedings, Third Annual Conference of the CFD Society of Canada, 25–27 June 1995, Banff, Canada,* Vol. 1, p. 121 edited by P. A. Thibault and D. M. Bergeron (CFD Soc., Canada, 1995).
12. H. Ritchie, C. Temperton, A. Simmons, M. Hortal, T. Davies, D. Dent, and M. Hamrud, *Mon. Weather Rev.* **123,** 489 (1995).
13. G. H. Golub and J. H. Welsh, *Math. Comput.* **23,** 221 (1969).
14. A. M. Ostrowski, *Solution of Equations in Euclidean and Banach Spaces,* 3rd ed. (Academic Press, New York, 1973).
15. J. F. Traub, *Iterative Methods for the Solution of Equations* (Prentice–Hall, Englewood Cliffs, NJ, 1964).
16. J. H. Wilkinson and C. Reinsch, *Linear Algebra in Handbook for Automatic Computation,* Vol. 2 (Springer-Verlag, New York, 1971).
17. J. H. Wilkinson, *The Algebraic Eigenvalue Problem* (Clarendon Press, Oxford, 1969).
18. L. Fox and D. F. Mayers, *Computing Methods for Scientists and Engineers* (Clarendon Press, Oxford, 1968).
19. J. Gerlach, *SIAM Rev.* **36,** 272 (1994).
20. M. Branders, R. Piessens, and M. De Meue, *J. Comput. Phys.* **42,** 403 (1981).
21. A. H. Stroud and D. Secrest, *Gaussian Quadrature Formulas* (Prentice-Hall, Englewood Cliffs, NJ, 1966).
22. E. Yakimiw, "Second-Order Correction to the Weight Formulas in Gaussian Quadrature," in *Proceedings of the Sixth International Symposium on Computational Fluid Dynamics, Sept. 4–8, 1995, Lake Tahoe, NV,* Vol. 3, p. 1399, edited by M. Hafez (University of California, Davis, 1995).