That is, we can efficiently compute the coefficients of the derivative of f in another classical OP expansion via the product $D_{(\lambda)}^{(\lambda+1)}\boldsymbol{f}$.

3. Quadrature

A quadrature rule approximates an integral by a weighted sum of samples of f. The classic interpolatory quadrature rules are Newton–Cotes, Gaussian quadrature, and Clenshaw–Curtis, which all take the form

$$\int_a^b w(x)f(x)\mathrm{d}x \approx \sum_{k=1}^{N} w_k f(x_k), \tag{3.1}$$

where $w(x)$ is a nonnegative integrable weight function and $\{x_i\}$ and $\{w_i\}$ are called the quadrature nodes and weights, respectively. It is known that an N-point Gaussian quadrature exactly integrates polynomials of degree $\leq 2N-1$. When $w(x) = 1$, the corresponding Gauss quadrature rule is called Gauss–Legendre.

3.1. Weighted Clenshaw–Curtis quadrature

Clenshaw–Curtis quadrature sets the nodes $\{x_k\}$ in (3.1) to be Chebyshev points that are trivial to generate, and the weights $\{w_k\}$ are then selected so that the quadrature rule is exact for any polynomial of degree $\leq N$. There are at least three main variations on Chebyshev points:

$$\begin{aligned} &\text{Fejér's 1st rule:} && x_k = \cos\left(\frac{k\pi}{N+1}\right), && 1 \leq k \leq N,\\ &\text{Fejér's 2nd rule:} && x_k = \cos\left(\frac{(k-1/2)\pi}{N}\right), && 1 \leq k \leq N,\\ &\text{Clenshaw–Curtis:} && x_k = \cos\left(\frac{(k-1)\pi}{(N-1)}\right), && 1 \leq k \leq N. \end{aligned}$$

The differences between these choices for the quadrature nodes is minor, though the Clenshaw–Curtis rule has the advantage that the N-point and $2N$-point nodal sets are nested allowing for reuse of function evaluation when N is doubled, which can have a significant impact in speed if calculating f is expensive. Therefore, Clenshaw–Curtis has become the most popular variant.

The weights $\{w_k\}$ are selected so that any polynomial p of degree $\leq N-1$ is exactly integrated. That is,

$$\int_{-1}^{1} w(x)p(x)\mathrm{d}x = \sum_{k=1}^{N} w_k p(x_k).$$

In particular, the quadrature must exactly integrate the Chebyshev polynomials $T_0(x), \ldots, T_{N-1}(x)$. We find that $\{w_i\}$ satisfy the following linear system:

$$\boldsymbol{T}_N(\boldsymbol{x})^\top \boldsymbol{w} = \boldsymbol{b}, \quad \boldsymbol{T}_N(\boldsymbol{x}) = \begin{pmatrix} T_0(x_1) & \cdots & T_{N-1}(x_1) \\ \vdots & \ddots & \vdots \\ T_0(x_N) & \cdots & T_{N-1}(x_N) \end{pmatrix}, \tag{3.2}$$

$$\boldsymbol{b}_n = \int_{-1}^{1} w(x) T_n(x) \mathrm{d}x,$$

where $\boldsymbol{x} = (x_1, \ldots, x_N)^\top$ and $\boldsymbol{w} = (w_1, \ldots, w_N)^\top$.

The original papers by Fejér (1933) and Clenshaw and Curtis (1960) were written before the rediscovery of the *Fast Fourier Transform* (FFT) in 1965 (Cooley and Tukey 1965); however, soon after Gentleman (1972) noted that the linear system $\boldsymbol{T}_N(\boldsymbol{x})^\top \boldsymbol{w} = \boldsymbol{b}$ could be solved using the FFT (see Section 4.1). Clenshaw–Curtis quadrature is popular because the rule is accurate, and the nodes and weights can be easily computed in $\mathcal{O}(N \log N)$ operations. In the 1970s, it was one of the only accurate rules for which large quadrature rules could be computed for large N. Nowadays, with fast algorithms to compute many Gauss quadrature rules, we regard Gauss and Clenshaw–Curtis formulas as both valuable for numerical integration (Trefethen 2008).

In 2006, Waldvogel showed a factor of 2 computational saving in computing $\{w_i\}$ when $w(x) = 1$ by exploiting the fact that $b_k = 0$ for odd k in (3.2) (Waldvogel 2003), which allows one to compute the weights using an FFT of half the size. Waldvogel's speedup can be extended to symmetric weights of the form $w(x) = (1 - x^2)^\alpha$ for $\alpha > -1$ (Sommariva 2013).

3.2. The original deviation of Gauss–Legendre quadrature

In contrast to Clenshaw–Curtis quadrature, an N-point Gauss–Legendre quadrature rule exactly integrates polynomials of degree $\leq 2N - 1$ by carefully selecting *both* the nodes and weights. Most textbooks in numerical analysis present the topic in its modern formulation using orthogonal polynomials, as derived by Jacobi (1826). However, the original derivation of the Gauss–Legendre quadrature rule by Gauss (1815) is a beautiful calculation involving continued fractions (Gautschi 1981*a*).

Let p be any polynomial of degree $\leq 2N - 1$ and Γ be the circle of radius $1 + \delta$ (for arbitrary small $\delta > 0$) centered at 0 and parameterized in a counterclockwise orientation. Then, from Cauchy's integral formula we can write

$$\int_{-1}^{1} p(x) \mathrm{d}x = \int_{-1}^{1} \frac{1}{2\pi \mathrm{i}} \int_\Gamma \frac{p(z)}{z - x} dz \mathrm{d}x = \frac{1}{2\pi \mathrm{i}} \int_\Gamma p(z) \phi(z) dz, \tag{3.3}$$

where $\phi(z) = \log((z+1)/(z-1))$ is the analytic function in $\mathbb{C} \cup \{\infty\} \setminus [-1,1]$ given by the standard branch of log, so that ϕ is positive for $z \in (1,\infty)$. Since we want the quadrature rule to be exact, we need

$$\int_{-1}^{1} p(x)\mathrm{d}x = \sum_{k=0}^{N} w_k p(x_k) = \frac{1}{2\pi i}\int_\Gamma p(z) r_N(z)\mathrm{d}z, \qquad r_N(z) = \sum_{k=1}^{N} \frac{w_k}{z - x_k}, \tag{3.4}$$

where $r_N(z)$ is a $(N, N-1)$ rational function with poles at the quadrature nodes $\{x_k\}$. By combining (3.3) and (3.4) and noting that p could be any polynomial, we require that

$$\frac{1}{2\pi i}\int_\Gamma z^j(\phi(z) - r_N(z))dz = 0, \qquad 0 \leq j \leq 2N-1.$$

Due to the orthogonality of the monomials on the unit circle, we need $\phi(z) - r_N(z) = \mathcal{O}(z^{2N})$. Gauss would have immediately known how to solve this using continued fractions, which were a popular topic in the early 1800s. In particular, $\phi(z)$ can be expressed as the following continued fraction:

$$\phi(z) = \log\left(\frac{z+1}{z-1}\right) = \frac{a_1}{z + \frac{a_2}{z + \frac{a_3}{z + \frac{a_4}{z + \cdots}}}}, \qquad a_k = -\frac{k^2}{4k^2-1}.$$

From this expression one can find the N-convergent, which is an $(N, N-1)$ rational function $r_N(z) = A_N(z)/B_N(z)$ such that $\phi(z) - r_N(z) = \mathcal{O}(z^{2N})$. Moreover, it is known that

$$B_{k+1}(z) = zB_k(z) - \frac{k^2}{4k^2-1}B_{k-1}(z), \quad k \geq 0, \quad B_0(z) = 1, \quad B_{-1}(z) = 0. \tag{3.5}$$

Gauss used this three-term recurrence to derive $B_N(z)$ and then found its zeros to calculate the quadrature nodes. For example, $B_3(z) = z^3 - (3/5)z$ so the 3-point Gauss quadrature rule has nodes $\{-\sqrt{3/5}, 0, \sqrt{3/5}\}$. One can then find the corresponding Gauss–Legendre weights by solving the linear system

$$\left(\boldsymbol{x}^0, \boldsymbol{x}^1, \cdots, \boldsymbol{x}^{N-1}\right)^\top \boldsymbol{w} = \boldsymbol{b}, \qquad \boldsymbol{b}_k = \int_{-1}^{1} x^k \mathrm{d}x = \frac{1-(-1)^{k+1}}{k+1}.$$

For the 3-point quadrature, we find that the weights are $\{5/9, 8/9, 5/9\}$. Nowadays, tables of Gauss–Legendre quadrature nodes and weights can be found throughout the internet and textbooks, though it is more practical to calculate them numerically.

3.3. The Golub–Welsch algorithm

Only a few years after Gauss' derivation, Jacobi (1826) realized the connection between Gauss quadrature and OPs. Jacobi first observed that the recurrence relationship in (3.5) is satisfied by the Legendre polynomials when scaled to be monic. That means that the N-point Gauss–Legendre quadrature nodes satisfy $P_N(x_k) = 0$ for $1 \leq k \leq N$. At the time, Jacobi's observation did not help one compute Gauss quadrature nodes and weights; however, it did allow Jacobi to generalize Gauss quadrature to other integrable weight functions. In particular, the Gauss quadrature nodes associated to (3.1) are given by the zeros of $p_N(x)$, where $p_0, p_1, \ldots,$ is a family of orthogonal polynomials associated to the weight function $w(x)$. In other words, the Gauss quadrature nodes are the zeros of an OP.

The most well-known algorithm for computing Gauss quadrature rules is the Golub–Welsch algorithm (Golub and Welsch 1969). It is based on the fact that

$$J_{0:N-1,0:N}\begin{pmatrix} p_0(\lambda) \\ \vdots \\ p_N(\lambda) \end{pmatrix} = \lambda \begin{pmatrix} p_0(\lambda) \\ \vdots \\ p_{N-1}(\lambda) \end{pmatrix}$$

and hence, by rearranging, we find

$$(J_N - \lambda I)\begin{pmatrix} p_0(\lambda) \\ \vdots \\ p_{N-2}(\lambda) \\ p_{N-1}(\lambda) \end{pmatrix} = \begin{pmatrix} 0 \\ \vdots \\ 0 \\ -b_{N-1}p_N(\lambda) \end{pmatrix}. \tag{3.6}$$

Here, J_N is the $N \times N$ truncation of the Jacobi operator associated with the OPs. The right-hand side of (3.6) is the zero vector if and only if λ is a root $p_N(\lambda)$. This means that the quadrature nodes can be computed in $\mathcal{O}(N^3)$ operations as an eigenvalue problem involving a tridiagonal matrix. Moreover, if the OPs are orthonormal (which does not change the zeros), then J_N is a symmetric tridiagonal matrix reducing the cost to only $\mathcal{O}(N^2)$ operations.

From (3.6), the eigenvector of J_N corresponding to eigenvalue x_j is

$$\begin{pmatrix} p_0(x_j) \\ \vdots \\ p_{N-1}(x_j) \end{pmatrix}.$$

This means that the quadrature weights can be obtained from the eigenvectors of J_N. Since an N-point Gauss quadrature is exact for polynomials up to degree $\leq 2N-1$, we know that the quadrature weights $\boldsymbol{w} = (w_1, \ldots, w_N)^\top$ satisfy

$$\boldsymbol{S}_N(\boldsymbol{x})D_{\boldsymbol{w}}\boldsymbol{S}_N^\top(\boldsymbol{x}) = D_{\boldsymbol{b}}, \qquad b_k = \int_{-1}^{1} w(x)p_k(x)dx,$$

where $(\boldsymbol{S}_N(\boldsymbol{x}))_{jk} = p_{k-1}(x_j)$ for $1 \leq j,k \leq N$. Therefore, $D_{\boldsymbol{w}}^{-1} = \boldsymbol{S}_N^\top(\boldsymbol{x})\boldsymbol{S}_N(\boldsymbol{x})$, i.e.,

$$\frac{1}{w_j} = \sum_{i=0}^{N-1} p_i(x_j)^2, \qquad 1 \le j \le N.$$

In other words, the reciprocal of w_j is equal to the sum of the squares of the entries of the eigenvector of J_N corresponding to eigenvalue x_j, where the first entry of the eigenvector is scaled to have the value of 1.

3.4. Computing Gauss quadrature rules associated with classical orthogonal polynomials

The most important examples of Gauss quadrature rules are those associated with classical OPs. By now, there are hundreds of publications on the subject of computing classical Gauss quadrature rules and the most efficient algorithms are tailor-made for each quadrature rule. In Table 3.2 we have surveyed the literature on computing Gauss–Legendre quadrature to observe the continuing progression to larger and larger quadrature rules (Townsend 2015). Also, see Figure 3.3. The surprise is that the Golub–Welsch algorithm is not, and was never, the fastest or most accurate algorithm for computing Gauss–Legendre quadrature rules. Though, its generality and ease of implementation makes it one of the most used algorithms for computing Gauss–Legendre quadrature still to this day.

There are now spectacularly fast algorithms for computing Gauss quadrature rules associated with classical OPs. The fastest methods are based on high-order asymptotic formulas for the quadrature nodes (the zeros of the classical OPs) and weights. For example, Bogaert derived asymptotic formulas for the n-point Gauss–Legendre nodes, which are accurate to about 16-digits of precision for any $N \ge 20$ (Bogaert 2014). Let $x_{N,k} = \cos(\theta_{N,k})$ be the kth node in the N-point Gauss quadrature rule. Then,

$$\theta_{N,k} = \alpha_{N,k} + \sum_{m=1}^{5} F_m\left(\alpha_{N,k}, \frac{\cos(\alpha_{N,k})}{\sin(\alpha_{N,k})}\right) v_N^{2m} + \mathcal{O}(v_N^{12}), \qquad v_N = \frac{1}{N+\frac{1}{2}}, \tag{3.7}$$

where $\alpha_{N,k} = v_N j_{0,k}$ and $j_{0,k}$ is the kth positive zero of the Bessel function $J_0(x)$. Here, the functions $F_1, \ldots, F_5$ are known explicitly. For example,

$$F_1(x,u) = \frac{1}{8}\frac{ux-1}{x}, \quad F_2(x,u) = \frac{1}{384}\frac{6x^2(1+u^2)+25-u(31u^2+33)x^3}{x^3},$$

where the remaining formulas for F_3, F_4, and F_5 along with an asymptotic formula for the quadrature weights can be found in (Bogaert 2014). For all practical purposes, (3.7) are essentially explicit formulas for the nodes and weights as they can be used for any $N \ge 20$ (in double precision) and the cost to evaluate them involves basic arithmetic and trigonometric evaluations. If

N	Algorithm	Reference
7	By hand	(Gauss 1815)
8	By hand	(Tallqvist 1905)
10	By hand	(Moors 1905)
7	By hand	(Nyström 1930)
12	By hand	(de F. Bayly 1938)
16	By hand	(Lowan, Davids and Levenson 1942)
16	Newton–Raphson	(Davis and Rabinowitz 1956)
64	Newton–Raphson	(Gawlik 1958)
96	Newton–Raphson	(Davis and Rabinowitz 1958)
10	QD algorithm	(Rutishauer 1962)
200	Newton–Raphson	(Love 1966)
512	Newton–Raphson	(Stroud and Secrest 1966)
50	Golub–Welsch	(Golub and Welsch 1969)
256	Newton–Raphson	(Lether 1978)
48	Asymptotics for nodes	(Gatteschi 1979)
640	Golub–Welsch	(Gautschi 1982)
32	Asymptotics	(Förster and Petras 1990)
32	Asymptotics for nodes	(Förster and Petras 1993)
8000	Higher order Newton–Raphson	(Yakimiw 1996)
8000	Optimized Golub–Welsch	(Yakimiw 1996)
200	Newton–Raphson	(Petras 1999)
2048	Golub–Welsch	(Swarztrauber 2003)
16384	Variant of Newton–Raphson	(Swarztrauber 2003)
12288	Newton-Raphson	(Bailey, Jeyabalan and Li 2004)
10^6	GLR algorithm	(Glaser, Liu and Rokhlin 2007)
10^8	Newton-Raphson with asymptotics	(Bogaert, Michiels and Fostier 2012)
10^6	Newton-Raphson with asymptotics	(Hale and Townsend 2013)
10^8	Asymptotics formulas for the nodes	(Bogaert 2014)

Table 3.2. A brief survey of computing Gauss–Legendre quadrature. Originally, small Gauss–Legendre quadrature rules were calculated by brute-force hand calculations; nowadays, a billion quadrature nodes and weights can be computed in a fraction of second on a standard laptop. The value N is the largest reported Gauss–Legendre quadrature rule in the publication.

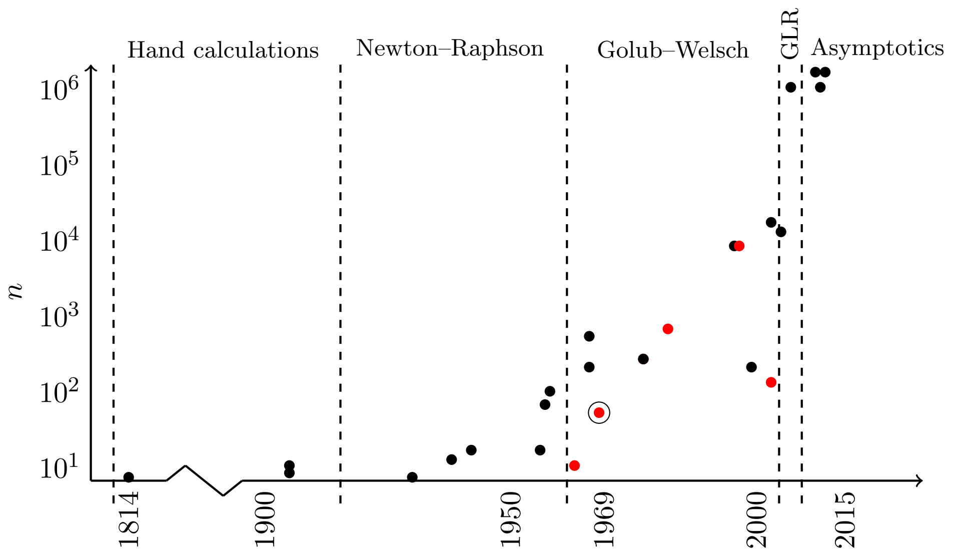

Figure 3.3. The history of computing high-order Gauss–Legendre quadrature. A dot represents published work, located at the publication year and the largest Gauss–Legendre rule reported therein. A red dot is a paper based on variants of the Golub–Welsch algorithm, and the circled red dot is the original paper by Golub and Welsch. GLR represents the Glaser, Liu and Rokhlin (2007) algorithm.

one wants to compute the nodes to more than 16 digits of precision, then one recommends using $\theta_{N,k}$ in (3.7) as an initial guess for Newton's method on the rootfinding problem $P_N(\cos\theta) = 0$ (Bogaert et al. 2012, Hale and Townsend 2013).

Other asymptotic formulas have been derived for Gauss–Jacobi quadrature (Gil, Segura and Temme 2019*b*) as well as Gauss–Laguerre (Gil, Segura and Temme 2019*a*) and Gauss–Hermite (Gil, Segura and Temme 2018). Again, for high-precision calculations, one can take these asymptotic formulas as initial guesses in Newton's method (Hale and Townsend 2013, Townsend, Trogdon and Olver 2016).

3.5. Computing non-classical Gauss quadrature rules

The previous section shows that having control over the asymptotic behavior of the zeros of classical OPs can lead to fast iterative-free algorithms for computing Gauss quadrature rules associated with classical weights. Going beyond classical weights, one can build on the recent advances in deriving the asymptotics of general OPs using the Riemann–Hilbert approach (Deift 1999) to determine the asymptotic behavior of the zeros for general weights on the real line. One can alternatively use the numerical solution of the underlying Riemann–Hilbert problem (Olver 2012) to calculate numerical asymptotic approximations to OPs and thereby calculate associated Gauss quadrature nodes for relatively general weights (Townsend et al. 2016).

4. Orthogonal polynomial transforms

Orthogonal polynomial transforms involve computing expansion coefficients in one OP basis either from samples or from another OP basis. There are two main types:

1 *Synthesis* and *analysis*. The infinite-dimensional synthesis operator is the map that takes OP expansion coefficients to the corresponding function by summing up the expansion. The natural finite-dimensional analogue of the synthesis operator is the map from expansion coefficients to values of the expansion at a specified grid. The infinite-dimensional analysis operator takes a function to its expansion coefficients, and the finite-dimensional counterpart takes function samples to expansion coefficients.

2 *Connection problems*. It is possible to represent a polynomial in different OP bases. The connection problem is the change-of-basis transform from one OP basis to another.

Our general strategy for synthesis, i.e., sampling a polynomial represented in a given OP expansion, is to first represent the polynomial in an OP expansion in which a fast evaluation scheme can be derived based on the FFT. Therefore, most of the synthesis and analysis operators are decomposed into two steps: (1) Convert one OP basis into another and (2) Evaluate the polynomial in its new OP basis using the FFT. Our strategy for analysis is, therefore, to reverse the process: (1) Compute the coefficients in a new OP basis using the FFT and (2) Convert from the new basis into the desired basis.

4.1. Synthesis and analysis based on the fast Fourier transform

The synthesis and analysis transforms that are closely connected to FFTs are the ones related to the four types of Chebyshev expansions and the associated four types of Gauss quadrature nodes (Mason 1993). That is, consider the four kinds of Chebyshev nodes,

$$\begin{aligned} x_k^{\mathrm{I}} &= -\cos\left(\frac{k-\frac{1}{2}}{N}\pi\right), & x_k^{\mathrm{II}} &= -\cos\left(\frac{k}{N+1}\pi\right), & 1 \le k \le N, \\ x_k^{\mathrm{III}} &= \cos\left(\frac{N-k+\frac{1}{2}}{N+\frac{1}{2}}\pi\right), & x_k^{\mathrm{IV}} &= \cos\left(\frac{k+\frac{1}{2}}{N+\frac{1}{2}}\pi\right), & 1 \le k \le N, \end{aligned} \tag{4.1}$$