On Fast and Stable Implementation of Clenshaw-Curtis and Fejér-Type Quadrature Rules

Shuhuang Xiang, Guo He, and Haiyong Wang

School of Mathematics and Statistics, Central South University, Changsha, Hunan 410083, China

School of Mathematics and Statistics, Huazhong University of Science and Technology, Wuhan, Hubei 430074, China

Correspondence should be addressed to Guo He; beibingyang009@163.com

Received 27 June 2014; Accepted 5 August 2014; Published 27 August 2014

Research Article

1. Introduction

The interpolation quadrature of the Clenshaw-Curtis rule as well as Fejér-type formulas for

\[ I[f] = \int_{-1}^{1} f(x)w(x)\,dx \approx I_N[f] = \sum_{k=0}^{N} w_k f(x_k) \quad (1) \]

has been extensively studied since Fejér [1, 2] in 1933 and Clenshaw and Curtis [3] in 1960, where the nodes \( x_k \) are of Chebyshev types while the weights \( \{w_k\} \) are computed by sums of trigonometric functions. When \( x_k = \cos((2k+1)\pi/(2N + 2)) \) (\( k = 0, 1, \ldots, N \)), this quadrature is called Fejér’s first-type rule. This kind of points is called the first kind of Chebyshev points, while Fejér’s second-type rule is corresponding to the Filippi points \( x_k = \cos((k+1)\pi/(N+2)) \) (\( k = 0, 1, \ldots, N \)) and the Clenshaw-Curtis-type quadrature to the Clenshaw-Curtis points (the second kind of Chebyshev points) \( x_k = \cos(k\pi/N) \) (\( k = 0, 1, \ldots, N \)). For more details, see Davis and Robinowitz [4], Sloan and Smith [5, 6], Sommariva [7], Trefethen [8], Waldvogel [9], and so forth.

In the case \( w(x) \equiv 1 \), a connection between the Fejér, Clenshaw-Curtis quadrature rules, and discrete Fourier transforms (DFTs) was given by Gentleman [10, 11], where the Clenshaw-Curtis rule is implemented with \( (N+1) \) nodes by means of a discrete cosine transformation (DCT).

An independent approach along the same lines, unified algorithms based on DFTs of order \( N \) for generating the weights of the two Fejér rules and of the Clenshaw-Curtis rule, was presented in Waldvogel [9]. A streamlined MATLAB code is given as well in [9]. In addition, Clenshaw and Curtis [3], O’Hara and Smith [12], Trefethen [8, 13], Xiang and Bornemann in [14], Xiang et al. [15–17], and so forth showed that the Gauss, Clenshaw-Curtis, and Fejér quadrature rules are about equally accurate.

In this paper, we focus the attention on the weight functions \( w(x) = (1-x)^\alpha(1+x)^\beta \) and \( w(x) = \ln((1+x)/2)(1-x)^\alpha(1+x)^\beta \). For these two weight functions, the Clenshaw-Curtis-type quadrature has been extensively studied in a series of papers of Piessens [18, 19] and Piessens and Branders [20–23], by using Chebyshev interpolant \( Q_N[f](x) = \sum_{n=0}^{N} a_n T_n(x) \) of \( f(x) \) at the \( (N+1) \) Clenshaw-Curtis points together with the modified moments \( M_n = \int_{-1}^{1} w(x)T_n(x)\,dx \) [24]:

\[ I[f] = \int_{-1}^{1} f(x)w(x)\,dx \approx I_N^{C-C}[f] = \sum_{n=0}^{N} a_n \int_{-1}^{1} T_n(x)w(x)\,dx \]
Table 1: Computation of \( M_n(\alpha, \beta) = \int_{-1}^{1} (1-x)^\alpha (1+x)^\beta T_n(x) dx \) with different \( n \) and \((\alpha, \beta)\) by the forward recursion of (4).

<table>
<thead>
<tr>
<th>( n )</th>
<th>5</th>
<th>10</th>
<th>100</th>
</tr>
</thead>
<tbody>
<tr>
<td>Exact value for ((20, -0.5))</td>
<td>-1.734810854604316e+05</td>
<td>4.04903666168904e+03</td>
<td>-3.083991348593134e-11</td>
</tr>
<tr>
<td>(4) for ((20, -0.5))</td>
<td>-1.734810854604308e+05</td>
<td>4.04903666169083e+03</td>
<td>1.787242305340324e-11</td>
</tr>
<tr>
<td>Exact value for ((100, -0.5))</td>
<td>-2.47129504946878e+29</td>
<td>1.174275526131223e+29</td>
<td>2.805165440968788e-29</td>
</tr>
<tr>
<td>(4) for ((100, -0.5))</td>
<td>-2.471295049468764e+29</td>
<td>1.174275526131312e+29</td>
<td>-1.380038973213404e+13</td>
</tr>
</tbody>
</table>

where \( T_n(x) \) is the Chebyshev polynomial of degree \( n \) and \( a_n \) can be efficiently computed by FFT [8, 10, 11] which is widely used for the approximation of highly oscillatory integrals such as [22, 25–30]. The modified moments \( M_n = \int_{-1}^{1} w(x) T_n(x) dx \) satisfy the following recurrence formulas for Jacobi weights or Jacobi weights multiplied by \( \ln((x+1)/2) \) [20].

(i) For \( w(x) = (1-x)^\alpha (1+x)^\beta \), by using Fasenmyer’s technique, the recurrence formula for the evaluation of the modified moments

\[
Q_n(\alpha, \beta) = \int_{-1}^{1} (1-x)^\alpha (1+x)^\beta T_n(x) dx
\]

is

\[
(\beta + \alpha + k + 2) Q_{n+1}(\alpha, \beta) + 2(\alpha - \beta) Q_n(\alpha, \beta) + (\beta + \alpha - n + 2) Q_{n-1}(\alpha, \beta) = 0
\]

with

\[
Q_0(\alpha, \beta) = 2^\beta (\alpha + 1) \Gamma(\alpha + 1) \Gamma(\beta + 1) / \Gamma(\beta + \alpha + 2),
\]

\[
Q_1(\alpha, \beta) = 2^\beta \Gamma(\alpha + 1) \Gamma(\beta + 1) / \Gamma(\beta + \alpha + 2) - 2(\beta - \alpha).
\]

The forward recursion is numerically stable [20], except in two cases:

\[
\alpha > \beta, \quad \beta = -\frac{1}{2}, \frac{3}{2}, \frac{5}{2}, \ldots
\]

\[
\beta > \alpha, \quad \alpha = -\frac{1}{2}, \frac{3}{2}, \frac{5}{2}, \ldots
\]

(ii) For \( w(x) = \ln((x+1)/2) (1-x)^\alpha (1+x)^\beta \), for

\[
G_n(\alpha, \beta) = \int_{-1}^{1} \ln \left( \frac{x+1}{2} \right) (1-x)^\alpha (1+x)^\beta T_n(x) dx,
\]

the recurrence formula [20] is

\[
(\beta + \alpha + k + 2) G_{n+1}(\alpha, \beta) + 2(\alpha - \beta) G_n(\alpha, \beta) + (\beta + \alpha - n + 2) G_{n-1}(\alpha, \beta) = 2Q_n(\alpha, \beta) - Q_{n-1}(\alpha, \beta) - Q_{n+1}(\alpha, \beta)
\]

with

\[
G_0(\alpha, \beta) = -2^\beta \Phi(\alpha, \beta + 1),
\]

\[
G_1(\alpha, \beta) = -2^\beta [2 \Phi(\alpha, \beta + 2) - \Phi(\alpha, \beta + 1)],
\]

where

\[
\Phi(\alpha, \beta) = B(\alpha + 1, \beta) \Psi(\alpha + \beta + 1) - \Psi(\beta)
\]

and

\[
B(x, y) \text{ is the Beta function and } \Psi(x) \text{ is the Psi function (see Abramowitz and Stegun [31]). The forward recursion is as numerically stable as (4) except for (6) or (7) [20].}
\]

Thus, the modified moments can be fast computed by the forward recursions (4) and (9) except the two cases (6) and (7), and the total costs for \( I_n[f] \) are \( O(N \log N) \) operations.

However, in case (6) or (7), the accuracy of the forward recursion is catastrophic particularly when \( \alpha - \beta \gg 1 \) or \( \alpha - \alpha \gg 1 \) and \( \alpha \gg 1 \) (see Table 1). In case (6) the relative errors \( \epsilon_n \) of the computed values \( Q_n(\alpha, \beta) \) obtained by the forward recursion behave approximately as

\[
\epsilon_n \sim n^{2(\alpha - \beta)}, \quad n \to \infty
\]

and in case (7) as

\[
\epsilon_n \sim n^{2(\beta - \alpha)}, \quad n \to \infty.
\]

In this paper, we will consider interpolation approaches for Clenshaw-Curtis rules as well as Fejér’s first- and second-type formulas for

\[
I[f] = \int_{-1}^{1} f(x) w(x) dx \approx \sum_{k=0}^{N} a_n \int_{-1}^{1} T_n(x) w(x) dx
\]

with \( w(x) = (1-x)^\alpha (1+x)^\beta \) or \( w(x) = (1-x)^\alpha (1+x)^\beta \ln((1+x)/2) \), which can be efficiently calculated in \( O(N \log N) \) operations. Computing the modified moments \( M_n \) in cases (6) and (7) by Oliver’s algorithm [32] or Lozier’s algorithm [33] with one starting value and one end value, as well as offering the very short codes for the evaluation of the coefficients by FFT, is the topic of this paper.

This paper is organized as follows. In Section 2.1, we studied the asymptotic expansions of the modified moments \( M_n \). Based on the results of the asymptotic expansions, we
Table 2: Computation of $Q_n(\alpha, \beta) = \int_{-1}^{1} (1-x)^\alpha (1+x)^\beta T_n(x) \, dx$ with $(\alpha, \beta) = (100, -0.5)$ and different $n$ by Oliver’s algorithm.

<table>
<thead>
<tr>
<th>$n$</th>
<th>100</th>
<th>500</th>
<th>1000</th>
</tr>
</thead>
<tbody>
<tr>
<td>Exact value for $(100, -0.5)$</td>
<td>$-2.83851909785347e-29$</td>
<td>$-2.283851909785405e-198$</td>
<td>$-1.24789046118544e-259$</td>
</tr>
<tr>
<td>Oliver’s method for $(100, -0.5)$</td>
<td>$2.805165440968788e-29$</td>
<td>$-2.283851909785405e-198$</td>
<td>$-1.24789046118544e-259$</td>
</tr>
<tr>
<td>(9) for $(−0.5, 100)$</td>
<td>$2.805165440968788e-29$</td>
<td>$-2.283851909785405e-198$</td>
<td>$-1.24789046118544e-259$</td>
</tr>
</tbody>
</table>

Table 3: Computation of $G_n(\alpha, \beta) = \int_{-1}^{1} (1-x)^\alpha (1+x)^\beta \ln((1+x)/2) T_n(x) \, dx$ with $(\alpha, \beta) = (−0.5, 100)$ and different $n$ by Oliver’s algorithm compared with that computed by the forward recursion (9).

<table>
<thead>
<tr>
<th>$n$</th>
<th>100</th>
<th>500</th>
<th>1000</th>
</tr>
</thead>
<tbody>
<tr>
<td>Exact value for $(−0.5, 100)$</td>
<td>$1.089944378602585e-28$</td>
<td>$7.222157005510106e-198$</td>
<td>$5.715301877322031e-259$</td>
</tr>
<tr>
<td>Oliver’s method for $(−0.5, 100)$</td>
<td>$1.089944378602206e-28$</td>
<td>$7.222157005510654e-198$</td>
<td>$5.715301877322483e-259$</td>
</tr>
</tbody>
</table>

2. Computation of the Modified Moments and the Coefficients of the Interpolation Polynomials

From (4) and (9), we see that if $Q_n(\alpha, \beta)$ can be efficiently and stably computed then $G_n(\alpha, \beta)$ can be so. Note that the asymptotic behaviour of two linearly independent solutions of (4) satisfies

$$y_{n,1} \sim n^{-2\alpha-2} \left[ 1 + O \left( n^{-2} \right) \right],$$
$$y_{n,2} \sim n^{-2\beta-2} \left[ 1 + O \left( n^{-2} \right) \right],$$

as $n \to \infty$ (see Denef and Piessens [34] and Piessens and Branders [20]). Consequently, the forward recursions for (4) and (9) are perfectly stable except cases (6) and (7). In these two cases, both the forward recursions and backward recursions for (4) and (9) are numerically unstable. In the following, we will consider Oliver’s algorithms to evaluate modified moments based on their asymptotic formulas of the moments with one starting value and one end value for these two cases.

2.1. Asymptotic Expansions of the Modified Moments

**Lemma 1** (Erdélyi [35]). If $0 < \lambda, \mu \leq 1$, and $\phi(t)$ is $m$ times continuously differentiable for $\alpha \leq t \leq \beta$, then

$$\int_{a}^{b} e^{ix(t-\alpha)^{-\lambda-1}(\beta-t)^{-\mu-1}} \phi(t) \, dt = B_m(x) - A_m(x) + O(x^{-m}),$$

where

$$A_m(x) = \sum_{n=0}^{m-1} \frac{\Gamma(n+\lambda)}{n!} e^{i\pi(n+\lambda-2)/2} x^{-n-\lambda} \times \frac{d^n}{dt^n} \left\{ (\beta-t)^{-\mu-1} \phi(t) \right\} \bigg|_{t=a},$$
$$B_m(x) = \sum_{n=0}^{m-1} \frac{\Gamma(n+\mu)}{n!} e^{i\pi(n-\mu)/2} x^{-n-\mu} \times \frac{d^n}{dt^n} \left\{ (t-a)^{-\lambda-1} \phi(t) \right\} \bigg|_{t=b}.$$  

**Proof.** The proof can be directly derived from the proof of Lemma 1 and the proof of the Theorem 5 in Erdélyi [36].
Table 4: Computation of $Q_n(\alpha, \beta) = \int_1^{-1} (1-x)^\alpha (1+x)^\beta T_n(x) dx$ with different $n$ and $(\alpha, \beta)$ by Oliver’s algorithm.

<table>
<thead>
<tr>
<th>$n$</th>
<th>2000</th>
<th>4000</th>
<th>8000</th>
</tr>
</thead>
<tbody>
<tr>
<td>Exact value for $(0.6, -0.5)$</td>
<td>9.551684021848334e-12</td>
<td>1.039402748103725e-12</td>
<td>1.13106574449745e-13</td>
</tr>
<tr>
<td>Oliver’s method for $(0.6, -0.5)$</td>
<td>9.551684021848322e-12</td>
<td>1.039402748103918e-12</td>
<td>1.131065744497332e-13</td>
</tr>
<tr>
<td>Exact value for $(10, -0.5)$</td>
<td>-8.412345942129556e-57</td>
<td>-2.005493070382270e-63</td>
<td>-4.781368848995069e-70</td>
</tr>
<tr>
<td>Oliver’s method for $(10, -0.5)$</td>
<td>-8.412345942129623e-57</td>
<td>-2.005493070383202e-63</td>
<td>-4.781368848995179e-70</td>
</tr>
</tbody>
</table>

Theorem 3. If $-1 < \alpha, -1 < \beta$, then

$$Q_n(\alpha, \beta) = 2^{\beta-\alpha} \sum_{k=0}^{m-1} a_k (\alpha, \beta) h(\alpha + k) + (-1)^n 2^{\alpha-\beta} \sum_{k=0}^{m-1} a_k (\beta, \alpha) h(\beta + k) + O(n^{-2 \min(\alpha, \beta) - 2m}),$$

(20)

where

$$h(\alpha) = \cos(\pi(\alpha + 1)) \Gamma(2\alpha + 2) n^{-2\alpha-2},$$

$$a_0(\alpha, \beta) = 1, \quad a_1(\alpha, \beta) = -\frac{\alpha}{12} - \frac{\beta}{4} - \frac{1}{6},$$

$$a_2(\alpha, \beta) = \frac{1}{120} + \frac{19\alpha}{1440} + \frac{\alpha^2}{288} + \frac{\alpha \beta}{48} + \frac{\beta}{32} + \frac{\beta^2}{32},$$

$$a_3(\alpha, \beta) = \frac{1}{5040} - \frac{1}{960} + \frac{107\alpha}{181440} - \frac{\alpha^2}{384} + \frac{\alpha \beta}{1920} - \frac{\beta^3}{384} + \frac{\alpha^3}{10368} - \frac{7\alpha \beta}{2880} + \frac{\alpha \beta^2}{1152} - \frac{\beta^3}{384},$$

(21)

Proof. For $-1 < \alpha, \beta < 1$, taking $x = \cos(\theta)$ in (3), we have

$$Q_n(\alpha, \beta) = \int_0^\pi \left(1 - \cos(\theta)\right)^{\alpha+1/2} \times \left(1 + \cos(\theta)\right)^{\beta+1/2} \cos(n\theta) d\theta$$

$$= \int_0^\pi \varphi(\theta) \theta^{2\alpha+1}\left(\pi - \theta\right)^{2\beta+1} \cos(n\theta) d\theta,$$

(22)

where

$$\varphi(\theta) = \left(\frac{1 - \cos(\theta)}{\theta^2}\right)^{\alpha+1/2} \left(\frac{1 + \cos(\theta)}{(\pi - \theta)^2}\right)^{\beta+1/2}.$$ 

Consequently, the desired result can be derived by applying (16) to (22).

The above outcome can be extended to the case of $-1/2 < \alpha, -1/2 < \beta$, since $Q_n(\alpha, \beta)$ can be written as

$$Q_n(\alpha, \beta) = \int_0^\pi \varphi(\theta) \theta^{2\alpha+1}\left[\pi - \theta\right]^{2\beta+1+1} \times \theta^{2\alpha+1}\left[\pi - \theta\right]^{2\beta+1-1} \times \cos(n\theta) d\theta,$$

(24)

where $[z]$ denotes the largest integer less than or equal to $z$.

Theorem 4. If $-1 < \alpha, -1 < \beta$, then

$$G_n(\alpha, \beta) = 2^{\beta-\alpha} \sum_{k=0}^{m-1} b_k(\alpha, \beta) h(\alpha + k) + (-1)^n 2^{\alpha-\beta} \sum_{k=0}^{m-1} b_k(\beta, \alpha) h(\beta + k) + O(n^{-2 \min(\alpha, \beta) - 2m}),$$

(25)

$$= \int_0^\pi \varphi(\theta) \theta^{2\alpha+1}\left(\pi - \theta\right)^{2\beta+1} \times \left(\frac{1 + \cos(\theta)}{2(\pi - \theta)^2}\right) \times \left(\frac{1 + \cos(\theta)}{(\pi - \theta)^2}\right)^{\beta+1/2} \cos(n\theta) d\theta + 2 \int_0^\pi \varphi(\theta) \ln(\pi - \theta) \theta^{2\alpha+1} \times \left(\pi - \theta\right)^{2\beta+1} \cos(n\theta) d\theta,$$

(27)

where

$$\phi(\beta) = \Psi(2\beta + 2) - \ln(2n) - \frac{\pi}{2} \tan(\pi\beta),$$

$$b_0 = 0, \quad b_1 = \frac{1}{12}, \quad b_2 = \frac{19}{1440}, \quad b_3 = \frac{9}{192}, \quad b_4 = \frac{7}{384}, \quad b_5 = \frac{25}{1920}, \quad b_6 = \frac{17}{1536}, \quad b_7 = \frac{1}{1024},$$

(26)

Proof. Letting $x = \cos(\theta)$ in (8) and using $\ln(1 + \cos(\theta))/2 = \ln((1 + \cos(\theta))/2(\pi - \theta)^2) + 2\ln(\pi - \theta)$, we have

$$G_n(\alpha, \beta) = \int_0^\pi \left[1 + \cos(\theta)\right] \left(\frac{1}{2(\pi - \theta)^2}\right) \times \theta^{\alpha+1} \left(\pi - \theta\right)^{\beta+1} \cos(n\theta) d\theta$$

$$+ 2 \int_0^\pi \varphi(\theta) \ln(\pi - \theta) \theta^{\alpha+1} \left(\pi - \theta\right)^{\beta+1} \cos(n\theta) d\theta,$$

(27)

Applying Lemmas 1 and 2 to the two integrals on the right hand side of (27), respectively, leads to the desired result.

Remark 5. Piessens and Branders gave the first term of asymptotic expansion for $Q_n(\alpha, \beta)$ and $G_n(\alpha, \beta)$ in [20] using the asymptotic theory of Fourier coefficients (Lighthill [37]). Furthermore, Piessens [18] presented the asymptotic expansion with explicit formulas for the first three terms for...
Table 5: Computation of $G_n(\alpha, \beta) = \int_{-1}^{1} (1-x)^\alpha (1+x)^\beta \ln((1+x)/2) T_n(x) dx$ with different $n$ and $(\alpha, \beta)$ by Oliver's algorithm.

<table>
<thead>
<tr>
<th>$n$</th>
<th>10</th>
<th>100</th>
<th>500</th>
</tr>
</thead>
<tbody>
<tr>
<td>Exact value for $(-0.4999, -0.5)$</td>
<td>-0.314181354550401</td>
<td>-0.031418104511487</td>
<td>-0.006283620842004</td>
</tr>
<tr>
<td>Oliver's method for $(-0.4999, -0.5)$</td>
<td>-0.314181354550428</td>
<td>-0.031418104511490</td>
<td>-0.006283620842004</td>
</tr>
<tr>
<td>Exact value for $(0.9999, -0.5)$</td>
<td>-0.895286620533541</td>
<td>-0.088858164406923</td>
<td>-0.017770353734330</td>
</tr>
<tr>
<td>Oliver's method for $(0.9999, -0.5)$</td>
<td>-0.895286620533558</td>
<td>-0.088858164406925</td>
<td>-0.017770353734330</td>
</tr>
</tbody>
</table>

Table 6: The CPU time for calculation of the first $(N+1)$ modified moments by the Oliver’s methods for $\alpha = -0.5$ and $\beta = 100$.

<table>
<thead>
<tr>
<th>Modified moments</th>
<th>$N = 10^3$</th>
<th>$N = 10^5$</th>
<th>$N = 10^7$</th>
<th>$N = 10^9$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$Q_n(\alpha, \beta)$</td>
<td>0.003954 s</td>
<td>0.011553 s</td>
<td>0.121553 s</td>
<td>1.200070 s</td>
</tr>
<tr>
<td>$G_n(\alpha, \beta)$</td>
<td>0.006087 s</td>
<td>0.028416 s</td>
<td>0.296032 s</td>
<td>2.933928 s</td>
</tr>
</tbody>
</table>

Algorithm 1

Function $a = \text{clenshaw.curtis}(f, N)$ % $(N+1)$-coefficients for C-C quadrature

$x = \cos(\pi*(0:N)' / N)$; % C-C points

$fx = \text{feval}(f,x)/(2*N)$; % $f$ evaluated at these points

$g = \text{fft}(fx([1:N+1:N:-1:2]));$ % FFT

$a = [g(1); g(2:N)+g(2*N:-1:N+2); g(N+1)];$ % Chebyshev coefficients

Algorithm 1

$Q_n(\alpha, \beta)$, which is equivalent to (20) except a minor clerical error. In [18], $a_i(\alpha, \beta) = -((\alpha + 2\beta + 2)/3)2^{\beta-\alpha-2}$ should be modified by $a_i(\alpha, \beta) = -((\alpha + 3\beta + 2)/3)2^{\beta-\alpha-2}$. The first term of (25) is different from the first asymptotic term for $G_n(\alpha, \beta)$ proposed in [18]; that is, $2^{\beta-\alpha} \sum_{k=0}^{N+1} \phi_k h(\alpha + k) + (-1)^{N+1} \sum_{k=0}^{N} h(\beta + k)[2 \phi_k(\beta, \alpha) + b_k]$. In order to ensure the accuracy of the modified moments evaluated by Oliver’s method, the end value of modified moments must be computed very precisely. In this paper, we use the first four-term truncation.

2.2. Oliver’s Algorithm. Let

$$A_N := \begin{pmatrix}
2(\alpha - \beta) & \alpha + \beta + 2 + 1 \\
\alpha + \beta + 2 - 2 & 2(\alpha - \beta) & \alpha + \beta + 2 + 2 \\
& \ddots & \ddots & \ddots \\
& & \alpha + \beta + 2 - (N-1) & 2(\alpha - \beta) & \alpha + \beta + 2 + (N-1) \\
& & & \alpha + \beta + 2 - N & 2(\alpha - \beta)
\end{pmatrix}, \quad (28)
$$

$$b_N := (- (\beta + \alpha + 1) Q_0(\alpha, \beta) \ 0 \cdots 0 \ - (\alpha + \beta + 2 + N) Q_{N+1})^T, \quad (29)$$

where $^T$ denotes the transpose; then the modified moments can be solved by

$$A_N M = b_N, \quad (30)$$

where $Q_{N+1}(\alpha, \beta)$ is computed by hypergeometric function [20] when $N \leq 2000$:

$$Q_{N+1}(\alpha, \beta) = 2^{\alpha+\beta+1} \frac{\Gamma(\alpha+1) \Gamma(\beta+1)}{\Gamma(\alpha+\beta+2)} \times_3 F_2 \left( \left[ N+1, -N-1, \alpha+1 \right], \left[ \frac{1}{2}, \alpha+\beta+2 \right], 1 \right). \quad (31)$$
function a = fejer1(f,N)
    x = cos(pi*(2*(0:N)+1)/(2*N+2)); % coefficients for Fejér's first rule
    fx = feval(f,x)/(N+1); % first kind of Chebyshev points
    g = fft(fx([1:N+1 N+1:-1:1])); % f evaluated at these points
    hx = real(exp(2i*pi*(0:2*n+1)/(4*n+4)).*g); % FFT
    a = hx(1:n+1);a(1)=0.5*a(1); % Chebyshev coefficients

Algorithm 2

function a = fejer2(f,N) % coefficients for Fejér's second rule
    x = cos(pi*(0:N+2)/(N+2)); % The first kind of Chebyshev points
    fx = feval(f,x)/(2*N+4); % f evaluated at these points
    g = fft(fx([1:N+3 N+2:-1:2])); % FFT
    b = [g(1); g(2:N+4:-1:N+4); g(N+3)];
    b(N+1:-2:1) = b(N+1:-2:1) - 2*b(N+3);
    b(N:-2:1) = b(N:-2:1) - b(N+2);
    b(1) = b(1) + mod(N+1,2)*b(N+3) + mod(N,2)*b(N+2)/2;
    a = b(1:N+1); % Chebyshev coefficients

Algorithm 3

function a = fejer1dct(f,N) % coefficients for Fejér's second rule
    x = cos(pi*(2*(0:N)+1)/(2*N+2)); % coefficients for Fejér's first rule
    fx = feval(f,x); % f evaluated at these points
    a = dct(fx)*sqrt(2/(N+1));a(1)=a(1)/sqrt(2); % Chebyshev coefficients

Algorithm 4

function a = fejer2idst(f,N) % coefficients for Fejér's second rule
    x = cos(pi*(1:N+1)/(N+2)); % Filippi points
    fx = feval(f,x).*sin(pi*(1:N+1)/(N+2)); % f evaluated at these points
    a = idst(fx); % Chebyshev coefficients

Algorithm 5

If \( N > 2000 \), \( Q_{N+1}(\alpha, \beta) \) is computed by using the asymptotic expression (20) with the first four-term truncation. Particularly, Oliver's algorithm can be fast implemented by applying LU factorization (chasing method) with \( O(N) \) operations.

In addition, for the weight \( w(x) = \ln((x+1)/2)(1-x)\beta \) in case of (6) or (7), the computation of the modified moments can be fixed up by Oliver's algorithm similar to (29) with one starting \( G_0(\alpha, \beta) = -2^{\alpha+\beta+1}\Phi(\alpha, \beta + 1) \) and one end \( G_{N+1}(\alpha, \beta) \). Consequently, the modified moments can be solved by

\[
A_N G = C_N, \quad G = (G_1(\alpha, \beta), G_2(\alpha, \beta), \ldots, G_N(\alpha, \beta))^T, \tag{32}
\]

where

\[
C_N = (c_1 \ c_2 \ \cdots \ c_{N-1} \ c_N)^T, \\
c_1 = - (\beta + \alpha + 1) G_0(\alpha, \beta) + 2Q_1(\alpha, \beta) - Q_0(\alpha, \beta) - Q_2(\alpha, \beta),
\]

\[
c_k = 2Q_k(\alpha, \beta) - Q_{k-1}(\alpha, \beta) - Q_{k+1}(\alpha, \beta), \quad k = 2, \ldots, N - 1,
\]

\[
c_N = - (\beta + \alpha + N + 2) G_{N+1}(\alpha, \beta) + 2Q_N(\alpha, \beta) - Q_{N-1}(\alpha, \beta) - Q_{N+1}(\alpha, \beta).
\]

(33)

The end value \( G_{N+1}(\alpha, \beta) \) can be calculated by its asymptotic formula (25) with the first four-term truncation.

Tables 2 and 3 show the accuracy of Oliver's algorithm for \( \alpha = 100 \) and \( \beta = -0.5, \alpha = -0.5 \) and \( \beta = 100 \), respectively. Table 4 displays the accuracy of Oliver's algorithm for \( \alpha = 0.6 \) and \( \beta = -0.5, \alpha = 10 \) and \( \beta = -0.5 \) for \( Q_{\alpha}(\alpha, \beta) \). Table 5 displays the accuracy of Oliver's algorithm for \( \alpha = -0.4999 \) and \( \beta = -0.5 \) and \( \beta = 0.9999 \) and \( \beta = -0.5 \) for \( G_{\alpha}(\alpha, \beta) \). Table 6 shows the CPU time for implementation of the two Oliver's algorithms. From these tables, we see that Oliver's algorithms have high efficiency and are precise. The MATLAB codes on Oliver's algorithms and all the MATLAB codes in this paper can be downloaded from [38].
Figure 1: The relative errors compared with Gauss quadrature for \( \int_{-1}^{1} (1-x)^{\alpha}(1+x)^{-\beta} f(x) dx \) evaluated by the Clenshaw-Curtis, Fejér's first- and second-type rules with \( N \) nodes, respectively: \( f(x) = \tan|x| \) or \( 1/(1+16x^2) \) and \( N = 10:1000 \).

2.3. Fast Computation of the Coefficients by FFT. The coefficients \( a_n \) for the interpolation polynomial \( Q_N[f](x) = \sum_{n=0}^{N} a_n T_n(x) \) at these three kinds of points can be efficiently computed by FFT. For the Clenshaw-Curtis points, the elegant MATLAB code on the \( a_n \) is from [8] (see Algorithm 1).

For the other two classes of points, by the sums of trigonometric functions at these two point-set, it is not difficult to get the FFT implementation. Here, we will not give details but just offer the following MATLAB functions.

For the first kind of Chebyshev points, we presented a MATLAB code for computing the coefficients \( a_n \) by FFT in Algorithm 2.

For the Filippi points, we presented a MATLAB code for computing the coefficients \( a_n \) by FFT in Algorithm 3.

In addition, the coefficients of the interpolation polynomials \( Q_N[f](x) \) at these Chebyshev-type points may be also efficiently evaluated by DCT and inverse discrete sine transform (IDST), respectively. The discrete cosine transform denoted by \( Y = \text{dct}(X) \) is closely related to the discrete Fourier transform but using purely real numbers and takes \( O(N \log N) \) operations for

\[
Y(k) = \sigma(k) \sum_{s=1}^{N} X(s) \cos \left( \frac{(k-1)\pi(2s-1)}{2N} \right)
\]
\[ I(f) = \int_{-1}^{1} (1-x)^{0.6} (1+x)^{-0.5} \sin((x-0.5)0.6)dx \]

The discrete sine transform denoted by \( Y = \text{dst}(X) \) and its inverse by \( X = \text{idst}(Y) \) both take \( O(N \log N) \) operations for

\[ Y(k) = \sum_{s=1}^{N} X(s) \sin \left( \frac{k \pi s}{N+1} \right). \]  

Notice that the coefficients \( a_j \) for the interpolation polynomial \( Q_N[f](x) = \sum_{s=1}^{N+1} a_{n-1} T_{n-1}(x) \) at \( \{\cos((2k-1)\pi/(2N+2))\}_{k=1}^{N+1} \) are represented by

\[ a_{n-1} = \frac{2}{N+1} \sum_{s=1}^{N+1} f \left( \cos \left( \frac{(2s-1)\pi}{2N+2} \right) \right) \times \cos \left( \frac{(2s-1)(n-1)\pi}{2N+2} \right), \]  

for \( n = 1, 2, \ldots, N+1, \)
and \( \tilde{a}_n \) for the interpolation polynomial \( Q_N[f](x) = \sum_{n=1}^{N+1} \tilde{a}_{n-1} U_{n-1}(x) \) at \( \{\cos(k\pi/(N + 2))\}_{k=1}^{N+1} \) satisfies

\[
\begin{align*}
&f(\cos(k\pi/(N + 2))) \sin(k\pi/(N + 2)) \\
&= \sum_{n=1}^{N+1} \tilde{a}_{n-1} \sin(k\pi/(N + 2)), \quad k = 1, 2, \ldots, N + 1.
\end{align*}
\]

Then both can be efficiently calculated by DCT and IDST, respectively. The MATLAB codes on the DCT and IDST are very short and just need three rows. Now, we give the codes in Algorithm 4 and Algorithm 5.

For the first kind of Chebyshev points, a MATLAB code for computing the coefficients \( a_n \) by DCT is presented in Algorithm 4.

As far as the Filippi points, a MATLAB code for computing the coefficients \( \tilde{a}_n \) by IDST is presented in Algorithm 5.

Remark 6. For the modified moments on the second kind Chebyshev polynomial \( \tilde{M}_n := \int_{-1}^{1} w(x) U_n(x) dx \), we have

\[
\tilde{M}_0 = M_0, \quad \tilde{M}_1 = 2M_1,
\]

\[
\tilde{M}_n = 2M_n + \tilde{M}_{n-2}, \quad n = 2, 3, \ldots;
\]

here we use the simple equation \( U_{n+2} = 2T_{n+2} + U_n \) (see [31, pp. 778]).

3. Numerical Examples

In this section, we illustrate the accuracy and efficiency of the Clenshaw-Curtis, Fejér’s first- and second-type rules for the functions \( \tan x \), \( 1/(1 + 16x^2) \) and \( |x - 0.5|^{1/6} \) by the algorithms presented in this paper, which are compared with the Gauss-Jacobi quadrature used with Chebfun v4.2 [39] (see Figure 1). The first column computed by Gauss-Jacobi quadrature in Figure 1 takes 85.7386 seconds and the others totally take 2.9211 seconds on a Lenovo computer with Intel Core 3.20 GHz and 3.47 GB RAM. Figure 2 shows the convergence errors by the three quadratures, which takes 7.336958 seconds.

Conflict of Interests

The authors declare that there is no conflict of interests regarding the publication of this paper.

Acknowledgment

This paper is supported by the National Natural Science Foundation of China, no. 11371376.

References


Submit your manuscripts at http://www.hindawi.com