Enhanced trapezoidal rule for discontinuous functions

Bengt Fornberg *
Department of Applied Mathematics, University of Colorado, Boulder, CO 80309, USA

Andrew Lawrence †
Department of Applied Mathematics, University of Colorado, Boulder, CO 80309, USA

July 15, 2023

Abstract

In many applications, data to be integrated is available only in the form of function values at predetermined equispaced points. For smooth periodic functions, the trapezoidal rule gives very accurate approximations to the integral, but is only second order accurate in more commonly arising non-periodic cases. The Gregory approach for end corrections can improve the accuracy order to 9 before the onset of negative weights, and of rapidly increasing ill-conditioning. In recent work, this has been extended to accuracy orders around 20.

This present study focuses on the more general case when one or both end points of the integration interval do not coincide with any of the equispaced grid points. We show that accuracy orders up to around 10 can also then be achieved, with all quadrature weights still remaining non-negative. This method can be utilized for example when functions to be integrated feature discontinuities at locations that can be separately determined (at or in between the equispaced grid points).

Keywords: Gregory's method, trapezoidal rule, Simpson's rule, Newton-Cotes, quadrature.
AMS classification codes: Primary: 65D30, 65D32; Secondary: 65B15.

1 Introduction

When a function that is to be integrated can be evaluated numerically at arbitrary locations, Gaussian quadrature-type methods are often used. To avoid loss of accuracy caused by the Runge phenomenon, these methods require data points to be clustered in very specific ways towards each end of the integration interval. However, in non-trivial applications, quadrature is usually a secondary task, and such specific data availability is then unrealistic. It is much more common that function values are available only at equispaced locations. Examples can include data from a time series, from grid-based calculations, pixel-based data from images, etc. Integration intervals that do not coincide with grid intervals can arise for numerous reasons. For example, when obtaining data, all future integration intervals needed for post-processing may not be known a priori. When using grid-based node layouts in more than 1-D, curved boundaries and interfaces are often serious complications. The present method applies in such cases to integrations along grid lines.

Email: fornberg@colorado.edu , ORCID 0000-0003-0014-6985
†*Email*: Andrew.Lawrence@colorado.edu

In the case of smooth functions and integration intervals starting and ending at grid points, the trapezoidal rule (TR)

$$\int_{x_0}^{x_N} f(x)dx \approx h\sum_{k=0}^{N} f(x_k) - \frac{h}{2}\left[f(x_0) + f(x_N)\right] \tag{1}$$

is second order accurate. In this formula, the nodes x_k are spaced a distance h apart. Already James Gregory[1] recognized that the dominant errors are caused by end effects, and described a method for end corrections (thus not interfering with the TR spectral accuracy throughout the interval interior).[2]

This approach by Gregory, summarized below in Section 2.2, allows the accuracy order to be increased to $O(h^9)$ before some weights become negative. A recently introduced variation, based on optimizing the solution to an under-determined system, increased this to $O(h^{20})$ [5], as summarized below in Section 2.4.[3] The present study shows that up to $O(h^{10})$ accuracy can be realized even in cases when integration interval end locations do not coincide with grid points. This is a situation which has received surprisingly little previous attention in the literature (further commented on in Chapter 5).

2 Summary of previous results: Grid aligned with interval end point

The sections in this Chapter 2 provide the background for the present extension to end points not aligned with grid points, which is then described in Chapter 3.

2.1 The Gregory method

Since the two interval ends can be considered separately, we describe at first Gregory's correction procedure at the left end of an infinite interval. We can further simplify the algebra by setting $h = 1$ since, for a general grid spacing h, one simply needs to multiply all obtained weights by h. Rather than the weights being all one (as in the leading sum in (1)), the Gregory approach gives accuracy order p when adding (to the value one) the correction set shown on line p in Table 1.[4] The $p = 2$ case corresponds to the standard trapezoidal rule.

At the right end of the integration interval, one similarly applies the correction set flipped left-right.[5] For intervals with only a few node points, the corrections from the two sides may overlap.

[1] James Gregory, 1638-1675, Scottish mathematician and astronomer. In describing this work in 1670 [7], Gregory used what later became known as Taylor expansions and generating functions well before calculus was presented by Leibniz (1684) and Newton (1687).

[2] TR for periodic functions is described in [14] (i.e., with no consideration of end effects).

[3] Building on ideas previously presented in [6].

[4] The accuracy order increases by one for each additional correction entry [11] (and not in steps of two as for the Newton-Cotes formulas and for derivative orders in the Euler-Maclaurin formulas).

[5] The Newton-Cotes approach (with Simpson's rule a special case) compensates for end errors by modifying the weights across the entire integration interval (no matter how long this is) - inconvenient as well as damaging the very high TR accuracy across interval interiors (c.f., [14]).

$p=$	Gregory corrections d_k to the weights all being one									
2	$-\frac{1}{2}$									
3	$-\frac{7}{12}$	$\frac{1}{12}$								
4	$-\frac{5}{8}$	$\frac{1}{6}$	$-\frac{1}{24}$							
5	$-\frac{469}{720}$	$\frac{59}{240}$	$-\frac{29}{240}$	$\frac{19}{720}$						
6	$-\frac{193}{288}$	$\frac{77}{240}$	$-\frac{7}{30}$	$\frac{73}{720}$	$-\frac{3}{160}$					
7	$-\frac{41393}{60480}$	$\frac{23719}{60480}$	$-\frac{11371}{30240}$	$\frac{7381}{30240}$	$-\frac{5449}{60480}$	$\frac{863}{60480}$				
8	$-\frac{12023}{17280}$	$\frac{6961}{15120}$	$-\frac{66109}{120960}$	$\frac{33}{70}$	$-\frac{31523}{120960}$	$\frac{1247}{15120}$	$-\frac{275}{24192}$			
9	$-\frac{2558783}{3628800}$	$\frac{1908311}{3628800}$	$-\frac{299587}{403200}$	$\frac{115963}{145152}$	$-\frac{426809}{725760}$	$\frac{112477}{403200}$	$-\frac{278921}{3628800}$	$\frac{33953}{3628800}$		
10	$-\frac{63887}{89600}$	$\frac{427487}{725760}$	$-\frac{3498217}{3628800}$	$\frac{500327}{403200}$	$-\frac{6467}{5670}$	$\frac{2616161}{3628800}$	$-\frac{24019}{80640}$	$\frac{263077}{3628800}$	$-\frac{8183}{1036800}$	
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\ddots$

Table 1: Corrections d_k to the weights all being one, according to Gregory's formulas, up through accuracy order $p = 10$ (shown for the left end of an interval in case of grid spacing $h = 1$). The middle entry on the $p = 10$ line shows the first instance of a correction $d_k < -1$, leading to a negative weight $w_k = 1 + d_k$.

2.2 Derivation of Gregory coefficients and correction weights

Gregory's idea was to look for an improved TR formula of the form

$$\int_0^\infty f(x)dx \sim \left(\sum_{k=0}^{\infty} f(k)\right) + \left[b_0\Delta^0 + b_1\Delta^1 + b_2\Delta^2 + \ldots\right]\, f(0)\,, \tag{2}$$

and to include only a few leading terms in the second sum. The operator Δ denotes here forward differences, i.e., $\Delta f(k) = f(k+1) - f(k)$, and thus

$$\begin{array}{ll} \Delta^0 f(0) & = f(0) \\ \Delta^1 f(0) & = f(1) - f(0) \\ \Delta^2 f(0) & = f(2) - 2f(1) + f(0) \\ \Delta^3 f(0) & = f(3) - 3f(2) + 3f(1) - f(0) \\ \vdots & \vdots \end{array} \tag{3}$$

Substituting $f(x) = e^{-zx}$ into (2)[6] gives

$$\frac{1}{z} = \frac{1}{1-e^{-z}} + \left[b_0 - b_1(1-e^{-z}) + b_2(1-e^{-z})^2 - b_3(1-e^{-z})^3 + - \ldots\right]. \tag{4}$$

With the further substitution

$$w = (1 - e^{-z})\,, \tag{5}$$

[6]Here following Gregory's pioneering use of the now-standard tool of generating functions.

i.e. $z = -\log(1-w)$, this becomes

$$\frac{1}{\log(1-w)} + \frac{1}{w} = -b_0 + b_1 w - b_2 w^2 + b_3 w^3 - + \dots \tag{6}$$

The coefficients b_k can now be calculated recursively based on the Taylor expansion of $\log(1-w)$

$$b_0 = -\frac{1}{2},\ b_1 = \frac{1}{12},\ b_2 = -\frac{1}{24},\ b_3 = \frac{19}{720},\ b_4 = -\frac{3}{160},\ b_5 = \frac{863}{60480},\ b_6 = -\frac{275}{24192},\ \dots \tag{7}$$

If one decides to use the first $n+1$ coefficients $b_0, b_1, \dots, b_n$, it follows from (2), (3) that the corrections $d_0, d_1, \dots, d_n$ to the leading $n+1$ weights are obtained by

$$\begin{bmatrix} d_0 \\ d_1 \\ \vdots \\ \\ \\ \vdots \\ d_n \end{bmatrix} = \begin{bmatrix} 1 & -1 & 1 & -1 & 1 & \cdots \\ & 1 & -2 & 3 & -4 & \cdots \\ & & 1 & -3 & 6 & \cdots \\ & & & 1 & -4 & \cdots \\ & & & & 1 & \cdots \\ & & & & & \ddots \end{bmatrix} \begin{bmatrix} b_0 \\ b_1 \\ \vdots \\ \\ \\ \vdots \\ b_n \end{bmatrix}, \tag{8}$$

or equivalently

$$\begin{bmatrix} 1 & 1 & 1 & 1 & 1 & \cdots \\ & 1 & 2 & 3 & 4 & \cdots \\ & & 1 & 3 & 6 & \cdots \\ & & & 1 & 4 & \cdots \\ & & & & 1 & \cdots \\ & & & & & \ddots \end{bmatrix} \begin{bmatrix} d_0 \\ d_1 \\ \vdots \\ \\ \\ \vdots \\ d_n \end{bmatrix} = \begin{bmatrix} b_0 \\ b_1 \\ \vdots \\ \\ \\ \vdots \\ b_n \end{bmatrix}. \tag{9}$$

2.3 Free parameter enhancement: Concept

Modifying the weights at the $p-1$ node points closest to the end of the interval (from being all one), provides the right number of free parameters to be able to satisfy the $p-1$ (linear) constraint equations that are required for ensuring the accuracy order p. Unfortunately, the Runge phenomenon causes weights obtained in this way to grow in magnitude exponentially with p. The key idea behind the enhancement in [5, 6] was to note that, if deciding on an accuracy order p but modifying the weights at $N > p-1$ nodes, the constraints for reaching accuracy order p can be satisfied, with $N-(p-1)$ parameters still remaining free. These can then be used for other purposes, such as to reduce the magnitudes of the weight corrections d_k. This idea of introducing extra free parameters, and then re-purposing these for other tasks (than maximizing the order of accuracy) has been used several times in other contexts[7].

2.4 Free parameter enhancement: Implementation

Since the b_k sequence is alternating in sign and is only slowly decreasing, the rapid growth in the Pascal triangle entries in (8) causes the central entries in the d_k sequence to grow rapidly with

[7] Examples include [8] for reducing weights in Newton-Cotes-type formulas (however still with nontrivial weights across the entire interval), and [4] for optimizing stability domains of parallel-in-time high-order ODE solvers when applied to wave type PDEs.

n (and make the Gregory method impractical for higher accuracy orders). As noted above, the approach for greatly reducing this growth, developed in [5, 6], was to choose $N > n$ and then correct $N+1$ leading weights while enforcing only the first $n+1$ order conditions, thus obtaining $N-n$ free parameters. Equation (9) can for this purpose be generalized to

$$\left[\begin{array}{cccccc|ccc} 1 & 1 & 1 & 1 & 1 & \cdots & \ldots & \ldots & \ldots \\ & 1 & 2 & 3 & 4 & \cdots & \ldots & \ldots & \ldots \\ & & 1 & 3 & 6 & \cdots & \ldots & \ldots & \ldots \\ & & & 1 & 4 & \cdots & \ldots & \ldots & \ldots \\ & & & & 1 & \cdots & \ldots & \ldots & \ldots \\ & & & & & \ddots & \ldots & \ldots & \ldots \end{array}\right] \left[\begin{array}{c} d_0 \\ d_1 \\ \vdots \\ \\ \vdots \\ d_n \\ \hline d_{n+1} \\ \vdots \\ d_N \end{array}\right] = \left[\begin{array}{c} b_0 \\ b_1 \\ \vdots \\ \\ \vdots \\ b_n \end{array}\right]. \tag{10}$$

Here, the rows of the Pascal matrix continue for another $N-n$ columns. The form of the underdetermined system (10) is well suited for providing the constraint equations when seeking optimized d_k entries (with the goal of ensuring that the d_k sequence is decreasing in magnitude while satisfying $w_k = 1 + d_k \geq 0$).

To incorporate a quadrature scheme in application codes, it can furthermore be convenient if the weights are all relatively simple rational numbers. One example, noted in [5], is the order $p = 10$ scheme with weight corrections $d_0, d_1, \ldots, d_{10}$ given by

$$\tfrac{1}{504}\ \left\{-\tfrac{22763}{64},\ \tfrac{59501}{225}, -\tfrac{64849}{180},\ \tfrac{11027}{32}, -\tfrac{40069}{225}, \tfrac{6071}{7200},\ \tfrac{45847}{800}, -\tfrac{40171}{1440}, -\tfrac{289}{2880},\ \tfrac{2917}{800}, -\tfrac{1957}{2400}\right\}. \tag{11}$$

3 The present method: Grid not aligned with the interval's end points

3.1 Gregory-type coefficients and free parameter enhancement

As in Chapter 2, we base the derivation of weights again on unit-spaced nodes at $x = 0, 1, 2, \ldots$, but with the integration interval $[\xi, \infty]$ instead of $[0, \infty]$, i.e., the integral in the left hand side of (2) is now replaced by $\int_\xi^\infty f(x)dx$, with $-1 < \xi \leq 0$. Denoting the resulting expansion coefficients $b_k(\xi)$ (with b_k above now corresponding to $b_k(0)$), the steps that led to (6) now instead give

$$\frac{(1-w)^\xi}{\log(1-w)} + \frac{1}{w} = -b_0(\xi) + b_1(\xi)w - b_2(\xi)w^2 + b_3(\xi)w^3 - + \ldots. \tag{12}$$

(reducing to (6) for $\xi = 0$). Since both $\log(1-w)$ and $(1-w)^\xi$ have simple Taylor expansions, the $b_k(\xi)$ coefficients are readily obtained numerically (c.f., the first two lines of executable code in the MATLAB function in the Appendix). Figure 1 shows $(-1)^{k+1}b_k(\xi)$ as function of $-1 \leq \xi \leq 0$ and $k = 0, 1, 2, ..., 15$.

The $b_k(\xi)$ coefficients will in the present work be utilized only numerically. Nevertheless, some

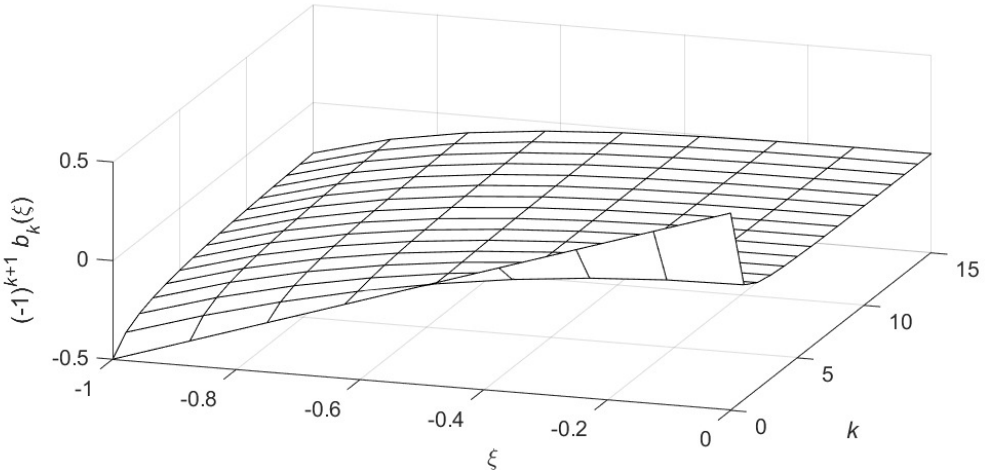


Figure 1: $(-1)^{k+1} b_k(\xi)$ as function of ξ and k.

explicit expressions may be of interest:

$$
\begin{aligned}
b_0(0) &= -\tfrac{1}{2} \quad , & b_0(\xi) &= b_0(0) + (-\xi) \\
b_1(0) &= \tfrac{1}{12} \quad , & b_1(\xi) &= b_1(0) + \left(-\tfrac{1}{2}\xi^2\right) \\
b_2(0) &= -\tfrac{1}{24} \quad , & b_2(\xi) &= b_2(0) + \left(\tfrac{1}{4}\xi^2 - \tfrac{1}{6}\xi^3\right) \\
b_3(0) &= \tfrac{19}{720} \quad , & b_3(\xi) &= b_3(0) + \left(-\tfrac{1}{6}\xi^2 + \tfrac{1}{6}\xi^3 - \tfrac{1}{24}\xi^4\right) \\
b_4(0) &= -\tfrac{3}{160} \quad , & b_4(\xi) &= b_4(0) + \left(\tfrac{1}{8}\xi^2 - \tfrac{11}{72}\xi^3 + \tfrac{1}{16}\xi^4 - \tfrac{1}{120}\xi^5\right) \\
&\cdots & &\cdots
\end{aligned}
$$

From results in [13] follows that

$$
b_k(0) = -\int_0^1 \binom{s}{k+1} ds, \quad \text{and} \quad b_k(\xi) = b_k(0) + \int_0^{-\xi} \binom{-s}{k} ds.
$$

This implies further that $b_{k+1}(0) = b_k(-1) + b_{k+1}(-1)$ and that, for large k, $b_k(0) \sim \frac{(-1)^{k+1}}{k(\log k)^2}$ and $b_k(-1) \sim \frac{(-1)^k}{\log k}$.

4 Numerical implementation

4.1 Floating point calculation of weight sets

The Appendix shows MATLAB code that computes corrections $d_k(\xi)$ for specified values of n, N, and xi (with xi the variable name for ξ). It is only for certain combinations of n and N that it finds valid corrections for all ξ in the required range $-1 < \xi \leq 0$. Such combinations include $n = 4$, $N = 8$ and $n = 8$, $N = 20$, in which cases the corrections $d_k(\xi)$ become as seen in Figures 2 and 3. At $\xi = -1$, a few of these corrections are -1, making the corresponding weights zero. The code is set to minimize $\sum_{k=0}^{N} d_k(\xi)^2 (k+1)^8$ while enforcing $d_k(\xi) \geq -1$.[8] The order of accuracy $p = 10$

[8]The power 8 in the sum is very arbitrary; low numbers do not force the correction weights to smoothly approach zero, while much higher numbers may cause difficulties for the optimization algorithm to converge.

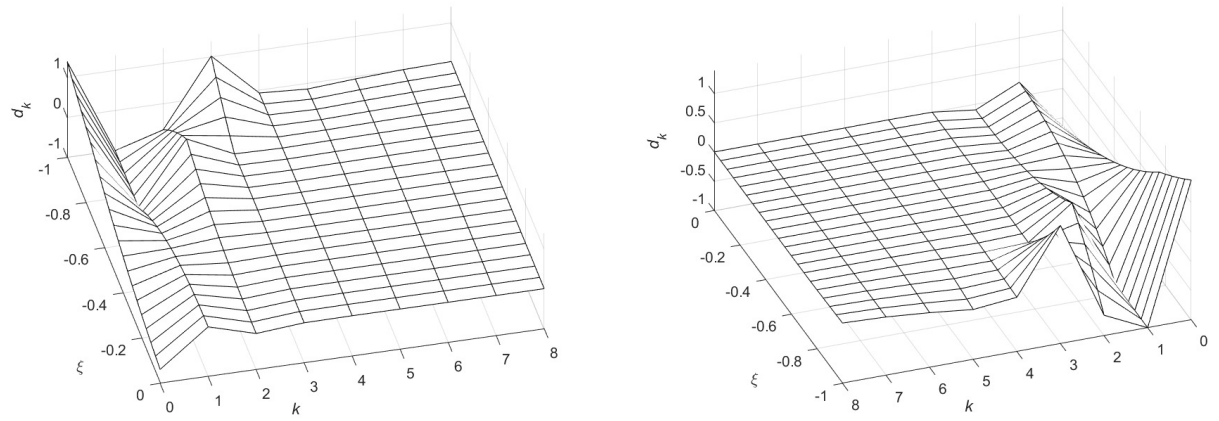


Figure 2: Corrections $d_k(\xi)$ in the case of $n = 4$, $N = 8$, as computed by the algorithm in the Appendix, seen from two perspectives in the k, ξ-plane.

appears to be as high as this procedure works across the full interval $-1 < \xi \leq 0$. In the special case of $\xi = 0$, this code provides an alternative to the least square-based code in the Appendix of [5] (see Test Problem 2 below).

4.2 Numerical tests

Two test problems are described next (with three more in Chapter 5). In four of these examples, we consider the case of two functions meeting discontinuously at some non-grid point location within the interval instead of a single smooth function with integration endpoints not coinciding with grid points. This gives rise to an identical type of numerical problem, previously considered in [2].

The rates of convergence as $h \to 0$ are jagged in all but the second test case, since the relative alignment of the discontinuity with its nearby grid points varies irregularly with h. The errors nevertheless in all cases follow their theoretically predicted rates. The second test problem repeats one from [5], serving to confirm that the present approach works just as well as the least squares-based method used previously in the standard case of interval ends aligning with grid points.

In all cases, as h decreases, the accuracy reaches and then remains at the level of machine rounding errors, i.e., around 10^{-16}. Although optimization algorithms often do not reach to this level of accuracy, we can note that the accuracy conditions and the non-negativity constraints in the present method are implemented precisely, and optimization only enters for the secondary task of ensuring the decay of the weight corrections d_k.

4.3 Test problem 1: Discontinuity within the integration interval.

Approximate

$$\int_0^1 f(x)dx \quad \text{with} \quad f(x) = \begin{cases} e^{-3x}\sin(20x) & , \quad 0 \leq x < 1/\sqrt{2} \\ -\frac{2}{5}\cos(10x) & , \quad 1/\sqrt{2} \leq x \leq 1 \end{cases} \tag{13}$$

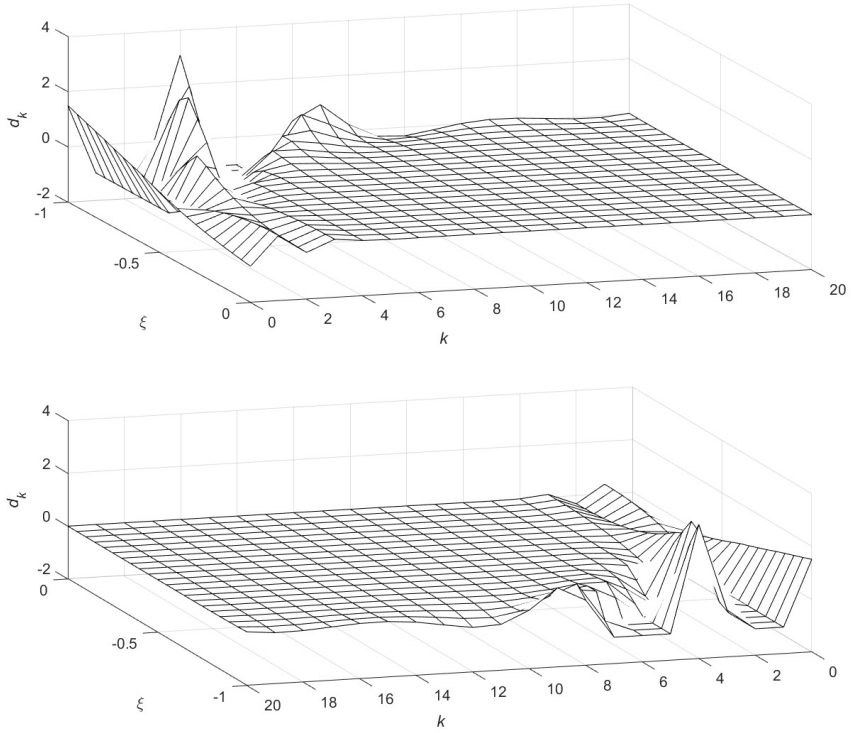


Figure 3: Corrections $d_k(\xi)$ in the case of $n = 8$, $N = 20$, as computed by the algorithm in the Appendix, seen from two perspectives in the k, ξ-plane.

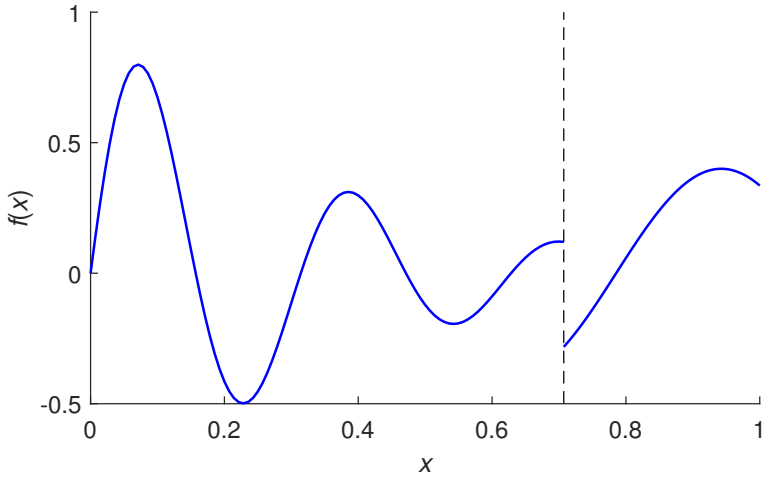


Figure 4: The test function (13), with a discontinuity at $x = 1/\sqrt{2}$.

using equispaced nodes over $[0, 1]$. Figure 4 shows this test function. Because the discontinuity location $1/\sqrt{2}$ is irrational, it will not coincide with a grid point at any level of refinement. Figure 5 shows the obtained rate of convergence using three methods:

1. Straight trapezoidal rule (TR). Instead of $O(h^2)$ convergence for smooth integrands, the discontinuity reduces the rate to $O(h^1)$,
2. The present method with $n = 4$, $N = 8$, featuring weight corrections as illustrated in Figure 2, and
3. The present method with $n = 8$, $N = 20$, featuring weight corrections as illustrated in Figure 3.

As theoretically derived, the convergence rates in the two latter cases are given by $p = n + 2$, i.e., they are $O(h^6)$ and $O(h^{10})$, respectively.

4.4 Test problem 2: Re-visit of the first test problem in [5]

The special case of $\xi = 0$ applies to equispaced quadrature nodes coinciding with both ends of the integration interval, and with no internal discontinuity. In this case, the accuracy orders can be increased well beyond $p = 10$. The test case of approximating $\int_0^1 \cos(20\sqrt{x})dx$ was considered previously in [5] for orders of accuracy up to $p = 20$. We quickly revisit this test case to show that the presently used optimization algorithm handles this special $\xi = 0$ case just as well as the previously described least squares approach.

Figure 6 shows the d_k coefficients the algorithm in the Appendix give with $N = 10$ and $N = 36$ in the cases of $p = 10$ and 20, respectively. Figure 7 shows the integrand in this test (sharply peaked at $x = 0$). Figure 8 (computed in extended precision using the Advanpix toolbox[9]) shows the how the errors go to zero in these two cases under node refinement (as well as TR results), confirming

[9] Advanpix, Multiprecision computing toolbox for MATLAB, http://www.advanpix.com/, Advanpix LLC, Yokohama, Japan.

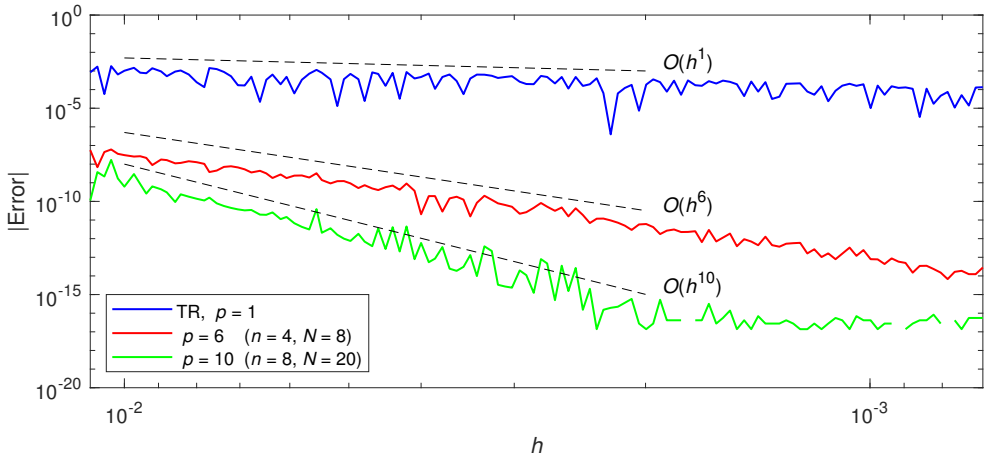


Figure 5: Error as function of h in Test problem 1. In the last case ($n = 8, N = 20$), converging as $O(h^{10})$, the error level reaches the machine precision 10^{-16} around $h = 2 \cdot 10^{-3}$ (and then remains at this level).

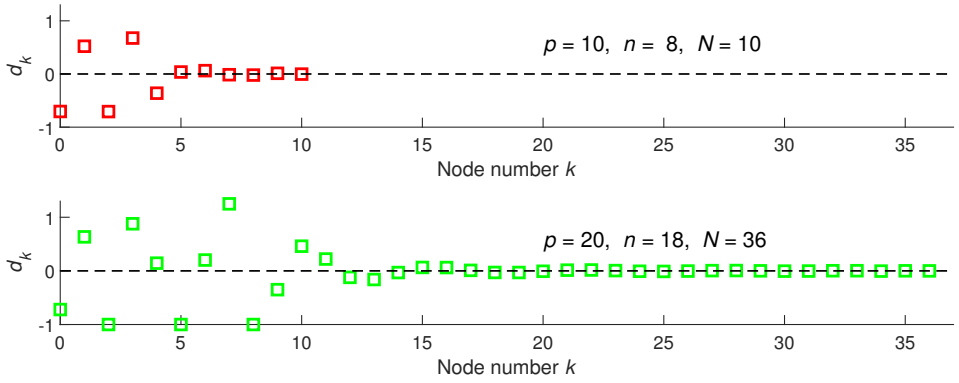


Figure 6: Weight corrections at the left interval end in the case of $\xi = 0$.

the expected convergence rates. Increasing p past 20 has not been pursued here, as the stencil sizes (N-values) then become impractically large.[10]

5 Brief comments on some other methods for obtaining quadrature weights

Since the value of an integral is a linear functional of the integrand, it is for this discussion natural to focus on linear quadrature schemes, i.e., methods expressible in the form of quadrature weights[11]. Key aspects of equispaced numerical quadrature methods are (i) how these handle interval end

[10] Very high accuracies also require the function that is integrated to be correspondingly many times differentiable (between discontinuities), which may in some applications be an unrealistic assumption.

[11] An example of a nonlinear scheme would be to integrate analytically a rational approximation of the integrand, as obtained for example with the AAA algorithm [12]. It is described for interpolation of equispaced data in [9] (although not for quadrature). Approximating the integrand with a combination of polynomials and partial fractions with known pole locations is a linear task, discussed in [15].

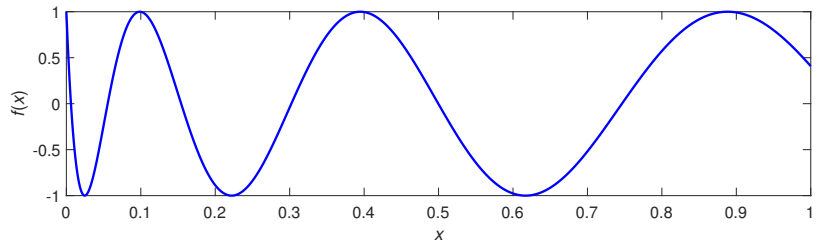


Figure 7: The integrand $\cos(20\sqrt{x})$ in Test problem 2.

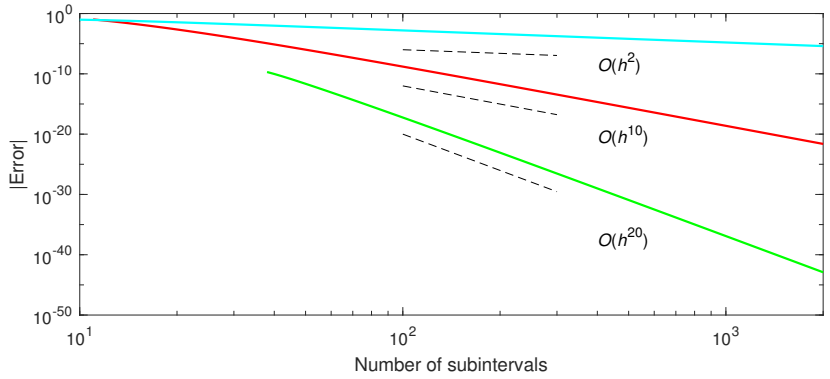


Figure 8: The curves, below the top one showing the convergence for the trapezoidal rule, use the weight sets shown in Figure 6 (with matching colors for the curves).

effects, (ii) how close their weights are to being constant throughout domain interiors[12], and (iii) all weights being non-negative.[13]

With equispaced data available, linear approaches typically amount to obtaining weights by means of integrating functional interpolants or approximations of such. Below are some possibilities:

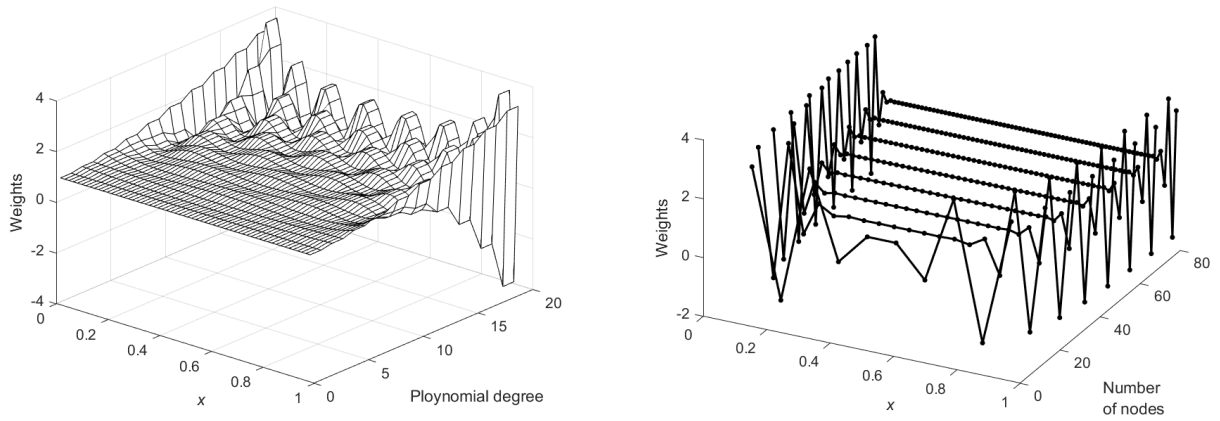
Least-square polynomial: Figure 9 (a) illustrates the case of $n = 60$ nodes over [0,1], equispaced by $h = \frac{1}{n+1}$, placed at $x_i = ih$, $i = 1, 2, \ldots, n$. The weights are seen to grow rapidly as the polynomial degree is increased. While the accuracy can be good in special cases, this is only if the integrand is well represented by a single polynomial of low degree[14] . The wave-nature seen in the quadrature weights will lead to very large errors whenever an integrand contains any similar-looking frequency component.

Barycentric interpolation: Linear barycentric rational quadrature is described in Chapter 9 of [1], and is shown to be competitive with 6^{th} order accurate Newton-Cotes (Boole's rule) in the case of interval end points coinciding with node points. Since barycentric-based quadrature has not been described without this end point assumption, nor for higher accuracy orders, we do not

[12]Realized by the trapezoidal rule (TR) and its Gregory-type generalizations; this gives spectral accuracy across interval interiors.

[13]In the (unlikely) case that analytic derivative information is available on each side of a discontinuity (assumed not to be the case in the present study), Euler-Maclaurin-based enhancements have been described [10].

[14]Fitting (by least squares) the equispaced data instead with Chebyshev polynomials and then integrating these exactly, as discussed in [3], makes no difference to the shown weights. That reference also notes that numerical stability requires $N = O(n^2)$, where N is the number of nodes (samples) and n is the polynomial degree (i.e., N typically needs to be far larger than n).



(a) Least square fit by polynomials (b) Interpolating cubic spline

Figure 9: Quadrature weights obtained by integrating approximating functions.

include it in the present illustrations of weights..

Cubic spline: Figure 9 (b) shows the weights with $n = 10, 20, \ldots, 80$ nodes over [0,1] (again with $h = \frac{1}{n+1}$, placed at $x_i = ih$, $i = 1, 2, \ldots, n$.). The excellent 'locality property' of cubic spline cardinal functions[15] leads to virtually constant weights across the domain interior. The problematic issues in this case becomes instead the low resulting order of accuracy and large magnitude (including negative) weights near the boundaries, c.f., Figure 9 (b) .

Enhanced Romberg-type quadrature: The recent study [2] describes a Romberg-type extrapolation method which includes a correction process for function disconuities. Its first three examples are here illustrated in Figures 10-12 and compared against the present method (in its $O(h^{10})$ version). Part (a) in these figures show the test functions (with analytic forms given in [2]), Part (b) shows as red dots the results tabulated in [2]. In the last two cases, two implementation versions were given (in which case, some h-values became inapplicable). Corresponding results for the present method are shown by the blue curves. In all cases, these latter curves level out close to the machine precision level of $O(10^{-16})$. Concerns about the enhanced Romberg-type method include:

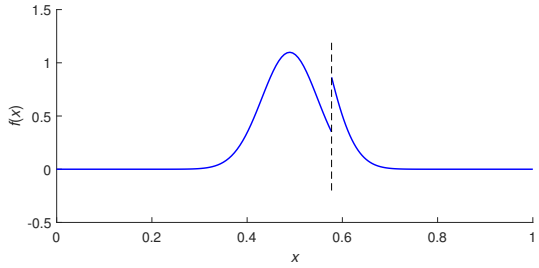
1. As is common for Romberg-type methods, only certain numbers of total node points can be used (and, with that, only certain h-values),

2. The algorithm has no counterpart to the present method's property of all weights being non-negative. It is noted on page 78 of [2] that "... sometimes the solution of the system suffers gravely from the ill-conditioning".

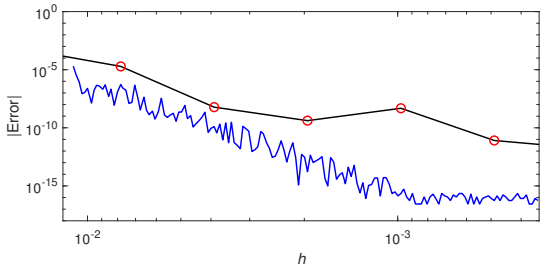
6 Discussion and conclusions

In computational contexts when high accuracies are needed, it is often cost-effective to use high order accurate numerical methods. One situation where this opportunity has not been much utilized

[15] Taking the value one at one node point and zero at all other nodes.

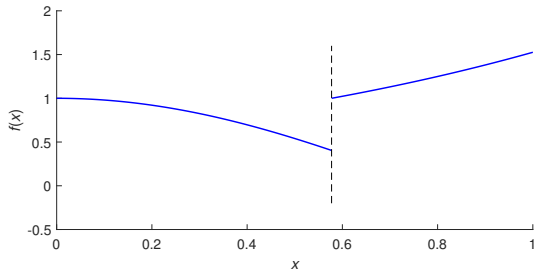


(a) Test function

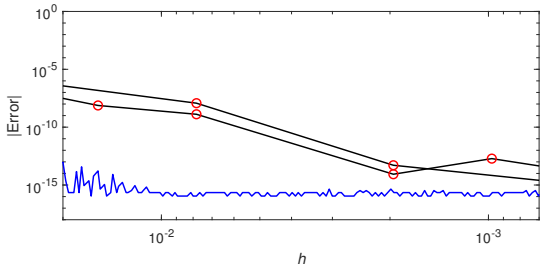


(b) Errors

Figure 10: First test case in [2]. In Part (b) of Figures 10-12, tabulated results from [2] are shown by red circles and results with the present method are shown by the blue (bottom) curve.

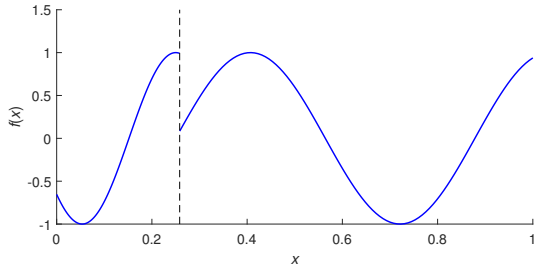


(a) Test function

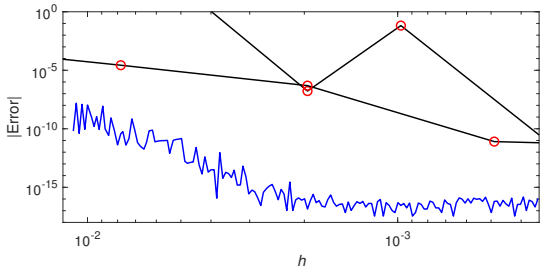


(b) Errors

Figure 11: Second test case in [2].



(a) Test function



(b) Errors

Figure 12: Third test case in [2] (re-scaled to the interval $0 < x < 1$)

is numerical integration of a smooth function based only on equispaced function values, in cases where the integration end point(s) do not align with grid points. We show here that accuracy orders up to around $O(h^{10})$ can readily be achieved by merely modifying standard trapezoidal rule weights near the interval ends (with all weights still remaining non-negative).

The present method applies also to functions sampled equidistantly and with a discontinuity at some known location (at or in-between grid points). In reverse, the present approach can also be used to accurately determine a discontinuity location from equispaced data if the integral is known (e.g., by means of some conservation law).

Funding: Andrew Lawrence acknowledges support from the US Air Force.[16]

7 Appendix: MATLAB code for finding weights using the quadprog routine

The following shows a MATLAB function for computing end corrections. We recall that the only parameter range that is relevant for ξ is $-1 < \xi \leq 0$.

```
function dk = d_k(n,N,xi)
% Input parameters
%    n    Specify accuracy order p = n+2
%    N    Modify the first N+1 weights.  It is required that N >= n.
%   xi    The \xi parameter, typical range -1 < xi <= 0.
% Output parameter
%   dk    Row vector of the weight corrections (from all = 1) for the first N+1 grid points.  It
%         will depend on n,N,xi if this function can find a solution for which all dk are <= -1.
c  = cumprod([1,((0:n)-xi)./(1:n+1)]); c = [c,zeros(1,n+1)];
bk = deconv(c,1./(1:n+2)).*cumprod(-ones(1,n+2)); bk(1) = []; bk = bk';
                                    % Solve constrained optimization problem
P  = pascal(N+1,1)'; P = P(1:n+1,:); P = P.*cumprod([1;-ones(n,1)]);
options = optimset('Algorithm','active-set','Display','off');
H  = diag((1:N+1).^8);                  % H-matrix; Optimize using quadprog
dk = quadprog(H,[],[],[],P,bk,-1*ones(N+1,1),[],zeros(N+1,1),options); dk = dk';
end
```

References

[1] J.-P. Berrut and G. Klein, *Recent advances in linear barycentric rational interpolation*, J. Comp. Appl. Math. **259** (2014), 95–107.

[2] J.-P. Berrut and M.R. Trummer, *Extrapolation quadrature from equispaced samples of functions with jumps*, Numerical Algorithms **92** (2023), 65–88.

[3] J.P. Boyd and F. Xu, *Divergence (Runge phenomenon) for least-squares polynomial approximation on an equispaced grid and Mock-Chebyshev subset interpolation*, Appl. Math. Comput. **210** (2009), 158–168.

[4] A.C. Ellison and B. Fornberg, *A parallel-in-time approach for wave-type PDEs*, Numerische Matematik **148** (2021), 79–98.

[16]The views expressed in this article are those of the authors and do not reflect the official policy or position of the Air Force, the Department of Defense or the U.S. Government.

[5] B. Fornberg, *Improving the accuracy of the trapezoidal rule*, SIAM Review **63** (2021), no. 1, 167–180.

[6] B. Fornberg and J.A. Reeger, *An improved Gregory-like method for 1-D quadrature*, Num. Math. **141** (2019), no. 1, 1–19.

[7] J. Gregory, *Letter to J. Collins, 23 November 1670.*, Oxford University Press (1841), 203–212, In Rigaud: Correspondence of Scientific Men.

[8] D. Huybrechs, *Stable high-order quadrature rules with equidistant points*, Comput. Appl. Math. **231** (2009), 933–947.

[9] D. Huybrechs and L.N. Trefethen, *AAA interpolation of equispaced data*, BIT **63** (2023), Article Nr 21.

[10] J.N. Lyness, *The calculation of Fourier coefficients by the Möbius inversion of the Poisson summation formula. Part II. Piecewise continuous functions and functions with poles near the interval [0,1]*, Math. Comput. **25** (1971), no. 113, 59–78.

[11] E. Martensen, *Optimale Fehlerschranken für die Qadraturformel von Gregory*, ZAMM **44** (1964), no. 4/5, 159–168.

[12] Y. Nakatsukasa, O. Sète, and L.N. Trefethen, *The AAA algorithm for rational approximation*, SIAM J. Sci. Comput. **40** (2018), no. 3, A1494–A1522.

[13] G.M. Phillips, *Gregory's method for numerical integration*, The Amer. Math. Monthly **79** (1972), no. 3, 270–274.

[14] L.N. Trefethen and J.A.C. Weideman, *The exponentially convergent trapezoidal rule*, SIAM Review **56** (2014), 384–458.

[15] J.A.C. Weideman and D.P. Laurie, *Quadrature rules based on partial fraction expansions*, Numerical Algorithms **24** (2000), 159–178.