Improving the accuracy of the trapezoidal rule

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

July 10, 2019

Abstract

The trapezoidal rule uses function values at equi-spaced nodes. It is very accurate for integrals over periodic intervals, but is usually quite inaccurate in non-periodic cases. Commonly used improvements, such as Simpson's rule and the Newton-Cotes formulas, are not much (if at all) better than the even more classical quadrature formulas described by James Gregory in 1670. For increasing orders of accuracy, these methods all suffer from the Runge phenomenon (the fact that polynomial interpolants on equi-spaced grids become violently oscillatory as their degree increases). In the context of quadrature methods on equi-spaced nodes, and for orders of accuracy around 10 or higher, this leads to weights of oscillating signs and large magnitudes. This article develops further a recently discovered approach for avoiding these adverse effects.

Keywords: Gregory's method, trapezoidal rule, Simpson's rule, Newton-Cotes, Euler-Maclaurin, quadrature, radial basis functions, RBF, RBF-FD.
AMS classification codes: Primary: 65D30, 65D32; Secondary: 65B15.

1 Introduction

The trapezoidal rule (TR)

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

can be traced back to Babylonian astronomers before 50 BC [11]. In this formula, the nodes x_k are spaced a distance h apart.

Much later, around 1740, Leonhard Euler and Colin Maclaurin (independently) discovered an

* *Email*: fornberg@colorado.edu

infinite sequence of further correction terms[1] to (1):

$$\int_{x_0}^{x_N} f(x)dx \sim h\sum_{k=0}^{N} f(x_k) - \frac{h}{2}\left[f(x_0)+f(x_N)\right] + \\ + \frac{h^2}{12}\left[f^{(1)}(x_0) - f^{(1)}(x_N)\right] - \frac{h^4}{720}\left[f^{(3)}(x_0) - f^{(3)}(x_N)\right] + \\ + \frac{h^6}{30240}\left[f^{(5)}(x_0) - f^{(5)}(x_N)\right] - \frac{h^8}{1209600}\left[f^{(7)}(x_0) - f^{(7)}(x_N)\right] + - \dots , \tag{2}$$

with coefficients obtained from the generating function

$$\frac{1}{1-e^{-z}} - \frac{1}{z} = \frac{1}{2} + \frac{1}{12}z - \frac{1}{720}z^3 + \frac{1}{30240}z^5 - \frac{1}{1209600}z^7 + - \dots \tag{3}$$

From (2), we can make two key observations:

1. *The leading errors in the TR come from the edges.* The usual 2^{nd} order accuracy (error $O(h^2)$) becomes 4^{th}order if $f'(x_0) = f'(x_N)$, 6^{th}order if also $f'''(x_0) = f'''(x_N)$, etc. For periodic problems, the accuracy increases beyond any algebraic order (assuming $f(x)$ is infinitely differentiable). This case is surveyed in [17, 18].

2. *It is at the edges one should adjust quadrature weights if one want to increase the accuracy of a TR-type scheme.*

Although James Gregory (1638-1675) lived too early to have the Euler-Maclaurin formula available, he nevertheless made these observations, and produced an alternative series of correction terms. Instead of requiring derivatives, they altered some trapezoidal weights near each end.

In the next Section 2, we describe further Gregory's work, display the quadrature weights his approach leads to, and contrast these weights with the (nowadays better known) Newton-Cotes weights. Section 3 gives some background to why the Gregory method has recently been re-visited, and summarizes a new approach that makes equi-spaced quadrature formulas practical also at high orders of accuracy. Section 4 describes how such weight sets can be calculated, both in floating point and as rational numbers, and gives also some test results. Section 5, containing some concluding remarks, is followed by References and an Appendix with a simple MATLAB code for calculating quadrature weights.

Some of the ideas behind this present article can be found in [9]. We here summarize, simplify, and expand on these earlier observations.

2 The pioneering work by James Gregory

An even cruder approximation than TR would be to use

$$\int_{x_0}^{x_N} f(x)dx \approx h\sum_{k=0}^{N} f(x_k). \tag{4}$$

[1] Asymptotically correct to all orders

In Gregory's methods, one adjusts $p-1$ weights at each end, and obtains then schemes of accuracy order p (with TR representing the $p=2$ case). Since the two ends of an interval are equivalent in terms of making these corrections, it suffices to consider just one boundary (for ex. the semi-infinite interval $x \in [0,\infty]$) , and furthermore set $h=1$. The quadrature weights w_k can then trivially be adjusted to step size h by multiplying each by h. We focus on approximations for which all the weights w_k are the same from some k and onward (since this provides optimal accuracy away from boundaries). A finite interval is treated by adding the weights corrections inwards from each end. The corrections from the two sides may overlap.

2.1 Derivation

With the notation $\Delta f(k) = f(k+1) - f(k)$, it follows that

$$\begin{array}{ll} \Delta^0 f(0) & = f(0) \\ \Delta^1 f(0) & = f(1) - f(0) \\ \Delta^2 f(0) & = f(2) - 2f(1) + f(0) \\ \Delta^3 f(0) & = f(3) - 3f(2) + 3f(1) - f(0) \\ \vdots & \quad \vdots \end{array} \tag{5}$$

with coefficients from Pascal's triangle. Gregory's idea was to look for an improved TR formula of the form

$$\int_0^\infty f(x)dx \sim \left(\sum_{k=0}^{\infty} f(k)\right) + \left[b_0\Delta^0 + b_1\Delta^1 + b_2\Delta^2 + \ldots\right]\, f(0)\,, \tag{6}$$

using only a finite number of these correction terms.

Similarly to how the Fourier transform expresses a real-valued functions $f(x)$ as a superposition of exponentials $e^{-z\,x}$ with z purely imaginary, it is natural for the one-sided interval $[0,\infty]$ to also include z-values from the right half-plane (corresponding to decaying modes). Substituting $f(x) = e^{-zx}$ into (6)[2] gives

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

With the substitution

$$w = (1-e^{-z})\,, \tag{7}$$

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

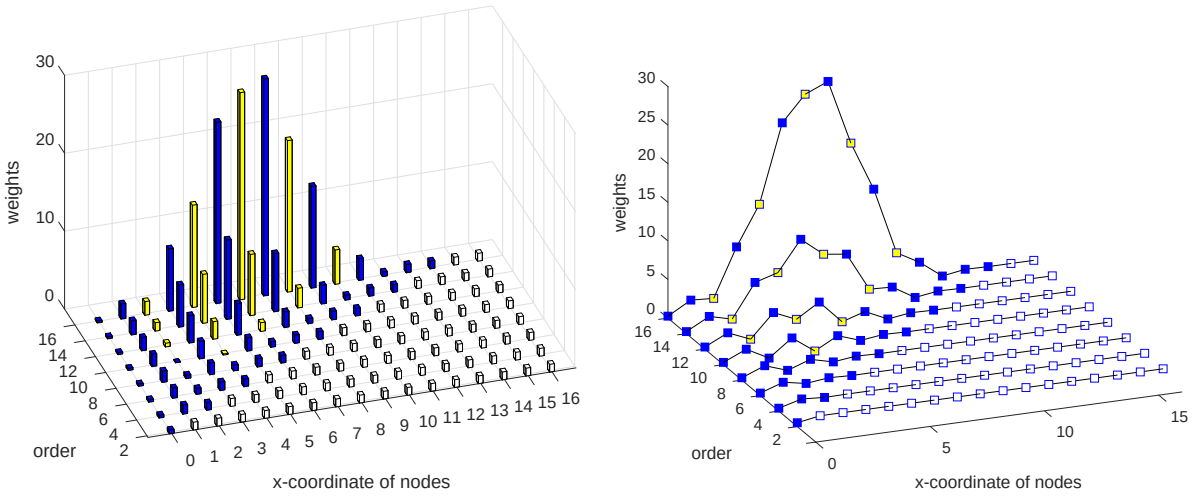
$$\frac{1}{\log(1-w)} + \frac{1}{w} = -b_0 + b_1 w - b_2 w^2 + b_3 w^3 - + \ldots \tag{8}$$

The coefficients b_k can now be calculated recursively based on the Taylor expansion of $\log(1-w)$ (as seen in the MATLAB code in the Appendix):[3]

$$b_0 = -\frac{1}{2},\ b_1 = \frac{1}{12},\ b_2 = -\frac{1}{24},\ b_3 = \frac{19}{720},\ b_4 = -\frac{3}{160},\ b_5 = \frac{863}{60480},\ b_6 = -\frac{275}{24192}, \ldots \tag{9}$$

[2]This same substitution in $\int_0^\infty f(x)dx \sim \sum_{k=0}^\infty f(k) + \sum_{n=0}^\infty \alpha_n f^{(n)}(0)$ gives (3) and the Euler-Maclaurin coefficients.

[3]As an alternative to calculating Taylor coefficients (such as in (3) and (8)) recursively, they can also be obtained numerically by applying the trapezoidal rule to Cauchy's integral formula - effectively carried out by means of FFTs. An automated procedure for choosing the radius of the integration path is given in [5]. However, this approach incurs truncation errors, and offers no advantages in the present context (for which simple recursions are available).



(a) (b)

Figure 1: Magnitudes of the Gregory weights w_k for orders $p = 2, 4, 6, \ldots, 16$, displayed in two slightly different ways. Weights equal to one shown in white, else blue for positive and yellow for negative weights.

These coefficients oscillate in sign and decay to zero for increasing k: $b_k \sim \frac{(-1)^{k+1}}{k\,(\log\,k)^2}$. As noted above, using only $b_0 = -\frac{1}{2}$ turns (6) into the TR. For each further term, the accuracy order increases by one.

Figure 1 shows the Gregory weights w_k for accuracy orders $p = 2,\ 4,\ 6, \ldots,\ 16$. Heights correspond to magnitude, with blue meaning a positive weight, and yellow a negative weight. Uncolored markers indicate weights that are exactly equal to one, beginning at location $p - 1$. It is visually clear that the Gregory formulas are severely affected by the Runge phenomenon [6, 16], manifesting itself in large oscillations of the weights. We see negative weights for orders $p = 10$ and above.

Table 1 gives the Gregory *corrections* $d_k = w_k - 1$, to be added to weights all starting off as one, up through $p = 10$. In case of a finite interval, we add this d_k-sequence from each end (in reversed order at the right end) to obtain actual weights to use. These two sequences can overlap; each one just has to fit into the interval.

The passage shown in Figure 2 appears in a letter that James Gregory wrote in 1670. We recognize here exactly the same content as in (5), (6), and (9) (however, with a typo in one of the denominators; 164 instead of 160) [4]. This letter by Gregory well precedes the first publications on calculus, by Leibniz (1684) and Newton (1687), respectively, as well as Brook Taylor's description in 1715 of what has become known as Taylor expansions. The early history of calculus may well have developed differently, had it not been for Gregory's premature death in 1675 (of stroke, at age 36).

[4] Only extracts of the letter are preserved. Although these do not include Gregory's derivation, it seems plausible from other preserved letter extracts that he used a generating function concept. The typo most likely occurred when the handwritten letter extract was typeset in 1870 [10].

$p=$	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	$-\frac{63887}{89600}$	$\frac{427487}{725760}$	$-\frac{3498217}{3628800}$	$\frac{500327}{403200}$	$-\frac{6467}{5670}$	$\frac{2616161}{3628800}$	$-\frac{24019}{80640}$	$\frac{263077}{3628800}$	$-\frac{8183}{1036800}$	
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋱

Table 1: Corrections d_k to the weights all being one, according to Gregory's formulas, up through order $p = 10$. The middle entry on the $p = 10$ line shows the first instance of a correction < -1, leading to a negative weight $w_k = 1 + d_k$.

ponendo AP = PO = c
PB = d

primam ex differentiis { primis = f ; secundis = h ; tertiis = i ; quartis = k ; quintis = l }

et omnes differentias affici signo +, erit ABP =

$$\frac{dc}{2} - \frac{fc}{12} + \frac{hc}{24} - \frac{19ic}{720} + \frac{3kc}{160} - \frac{863lc}{60480} + \text{&c. in infinitum.}$$

Figure 2: Brief extract from the bottom of page 208 and the top of page 209 in [10].

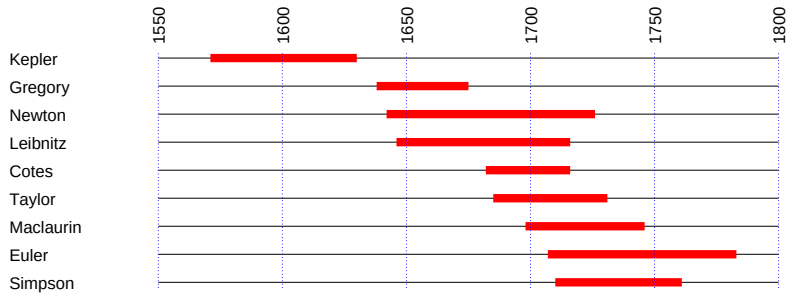


Figure 3: The life spans of the pioneering mathematicians in the early history of numerical quadrature.

2.2 Some comments on the Newton-Cotes (NC) family of methods

The TR can be thought of as integrating exactly a piece-wise linear interpolant to $f(x)$. Simpson's rule is obtained if one divides the interval in node groups $\{x_0, x_1, x_2\}$, $\{x_2, x_3, x_4\}, \ldots,$ $\{x_{N-2}, x_{N-1}, x_N\}$ and, separately for each, fits a parabola. In place of the TR weight sequence $h\{\frac{1}{2}, 1, 1, \ldots, 1, \frac{1}{2}\}$, one then obtains $\frac{h}{3}\{1, 4, 2, 4, \ldots, 4, 2, 4, 1\}$. Already Kepler used this approach[5], and Newton and Cotes carried this concept to increasingly high orders (as known from from their notes, edited and published by Simpson). This NC approach is conceptually questionable for several reasons:

- It attempts to correct for end errors by changing weights across the entire interval,
- For periodic problems (or integrals over very long intervals), TR with its weights equal, is optimal. Under refinement, the Simpson weights lose approximately half of the digits of accuracy that TR gives (and worse still for higher order NC versions).
- The NC weights diverge for increasing orders even faster than the Gregory weights (and do so across the entire interval).

Surprisingly, modern numerical analysis text books rarely mention the Gregory approach while often discussing the NC approach at great length.

History is often unjust in how methods become named after different individuals. As a reference to the names mentioned so far, Figure 3 shows timelines of their lives. We can note how Gregory's brief life preceded the active time spans of many who later became credited with results he had understood (and had mentioned in his letters).

3 The present method

At clear indication that the Runge problem can be eliminated (or at least greatly reduced) for increasing order Gregory-type methods emerged from [12]. The problem considered there was more general – numerical quadrature using scattered nodes over bounded curved surfaces in 3-D space. To verify that the RBF-FD (radial basis function-generated finite difference) approach employed was computationally competitive in its handling of surface edges, an extremely simplified test case

[5] In the non-astronomical context of approximating volumes of wine barrels.

was considered: 1-D equi-spaced nodes on a finite interval. It then transpired that this approach produced Gregory-like methods of high orders, but with much reduced Runge phenomenon (for a general background on RBF-FD approximations, see [7, 8]). These observations motivated the work in [9]. In yet another context (numerical evaluation of singular integrals), ideas similar to the ones exploited here were considered in [2], in turn building on ideas in [13] (for non-equi-spaced quadrature).

3.1 Derivation of linear system for TR corrections

Weights w_k that make (4) exact for all functions $e^{-z\,x}$ would satisfy $\frac{1}{z} = \sum_{k=0}^{\infty} w_k e^{-z\,k}$ (with Re $z > 0$). Subtracting from this the identity $\frac{1}{1-e^{-z}} = \sum_{k=0}^{\infty} 1 \cdot e^{-z\,k}$ gives

$$\frac{1}{z} - \frac{1}{1-e^{-z}} = \sum_{k=0}^{\infty} d_k e^{-z\,k}, \tag{10}$$

where $d_k = w_k - 1$. The order of accuracy of the quadrature method can be shown to correspond to the number of Taylor coefficients that match between the two sides of (10) when expanding around $z = 0$.[6] The variable change (7) gives

$$\begin{aligned} -\left(\frac{1}{\log(1-w)} + \frac{1}{w}\right) &= \sum_{k=0}^{\infty} d_k e^{k\,\log(1-w)} \\ &= \sum_{k=0}^{\infty} d_k (1-w)^k = \sum_{k=0}^{\infty} d_k \left(\sum_{i=0}^{k} \binom{k}{i}\right) (-1)^i w^i = \\ &= \sum_{i=0}^{\infty} (-1)^i w^i \left(\sum_{k=i}^{\infty} d_k \binom{k}{i}\right). \end{aligned}$$

By (8), the left hand side (LHS) above equals $\sum_{i=0}^{\infty} (-1)^i b_i w^i$. We next equate the coefficients for powers w^i, $i = 0, 1, 2, \ldots, n$, and include coefficients d_k, $k = 0, 1, 2, \ldots, N$, with $N \geq n$. This gives rise to $n+1$ linear equations in $N+1$ variables, under-determined if $N > n$:

$$\begin{bmatrix} 1 & 1 & 1 & 1 & 1 & \cdots & \cdots & \cdots \\ & 1 & 2 & 3 & 4 & \cdots & \{\text{Pascal's} & \cdots \\ & & 1 & 3 & 6 & \cdots & \text{triangle}\} & \cdots \\ & & & 1 & 4 & \cdots & \cdots & \cdots \\ & & & & 1 & \cdots & \cdots & \cdots \\ & & & & & \ddots & \cdots & \cdots \end{bmatrix}_{(n+1)\times(N+1)} \begin{bmatrix} d_0 \\ d_1 \\ \vdots \\ d_n \\ \vdots \\ d_N \end{bmatrix}_{N+1} = \begin{bmatrix} b_0 \\ b_1 \\ \vdots \\ \vdots \\ b_n \end{bmatrix}_{n+1}. \tag{11}$$

This formulation is well suited for numerical calculation of the correction coefficients d_k, $k = 0, 1, 2, \ldots, N$, as described in Section 4.

3.2 A re-formulation of the linear system

We get additional analytic insights by noting that the inverse of the square leftmost $(n+1)\times(n+1)$ block of the Pascal matrix in (11) becomes again an upper triangular Pascal matrix, but with

[6] The topic of *Prony's method* is to obtain good matches not just at $z = 0$ but along lines satisfying Re $z \geq 0$.

alternating signs:

$$\begin{bmatrix} 1 & 1 & 1 & 1 & 1 & \cdots \\ & 1 & 2 & 3 & 4 & \cdots \\ & & 1 & 3 & 6 & \cdots \\ & & & 1 & 4 & \cdots \\ & & & & 1 & \cdots \\ & & & & & \ddots \end{bmatrix}^{-1} = \begin{bmatrix} 1 & -1 & 1 & -1 & 1 & \cdots \\ & 1 & -2 & 3 & -4 & \cdots \\ & & 1 & -3 & 6 & \cdots \\ & & & 1 & -4 & \cdots \\ & & & & 1 & \cdots \\ & & & & & \ddots \end{bmatrix}.$$

Multiplying (11) by this inverse from the left produces

$$\left[\begin{array}{ccccc|ccc} 1 & & & & & \cdots & \cdots & \cdots \\ & 1 & & & & \cdots & \cdots & \cdots \\ & & 1 & & & \cdots & B & \cdots \\ & & & 1 & & \cdots & \cdots & \cdots \\ & & & & 1 & \cdots & \cdots & \cdots \\ & & & & & \cdots & \cdots & \cdots \end{array}\right] \begin{bmatrix} d_0 \\ d_1 \\ \vdots \\ d_n \\ \vdots \\ d_N \end{bmatrix} = \begin{bmatrix} 1 & -1 & 1 & -1 & 1 & \cdots \\ & 1 & -2 & 3 & -4 & \cdots \\ & & 1 & -3 & 6 & \cdots \\ & & & 1 & -4 & \cdots \\ & & & & 1 & \cdots \\ & & & & & \ddots \end{bmatrix} \begin{bmatrix} b_0 \\ b_1 \\ \vdots \\ b_n \end{bmatrix}. \tag{12}$$

By (5) and (6), the RHS becomes the Gregory method's correction coefficients. In the $N = n$ case (meaning that the B-matrix is absent), this vector obviously matches the LHS. With the b-sequence (9) having alternating signs, there will be no cancellations when the RHS is evaluated - and it is clear that the Gregory coefficients will grow in magnitude at about the same exponential rate as do the Pascal triangle entries.

The opportunity to eliminate this growth in the d_k-entries for increasing orders p arises from choosing $N > n = p - 2$, i.e with the B-matrix present in the LHS of (12). Its entries can readily be expressed in closed form. Counting its rows $i = 0, 1, \ldots, n$ and its columns $j = 1, 2, \ldots, N - n$, then $B_{i,j} = \frac{(-1)^i\, j}{n+j-i} \binom{n+j}{j} \binom{n}{i}$. For example in the case of $n = 8$ and $N = 10$, equation (12) becomes

$$\left[\begin{array}{ccccccccc|cc} 1 & & & & & & & & & 1 & 9 \\ & 1 & & & & & & & & -9 & -80 \\ & & 1 & & & & & & & 36 & 315 \\ & & & 1 & & & & & & -84 & -720 \\ & & & & 1 & & & & & 126 & 1050 \\ & & & & & 1 & & & & -126 & -1008 \\ & & & & & & 1 & & & 84 & 630 \\ & & & & & & & 1 & & -36 & -240 \\ & & & & & & & & 1 & 9 & 45 \end{array}\right] \left[\begin{array}{c} d_0 \\ d_1 \\ \vdots \\ \vdots \\ \vdots \\ \vdots \\ d_8 \\ \hline d_9 \\ d_{10} \end{array}\right] = \begin{bmatrix} -\frac{63887}{89600} \\ \frac{427487}{725760} \\ -\frac{3498217}{3628800} \\ \frac{500327}{403200} \\ -\frac{6467}{5670} \\ \frac{2616161}{3628800} \\ -\frac{24019}{80640} \\ \frac{263077}{3628800} \\ -\frac{8183}{1036800} \end{bmatrix}, \tag{13}$$

where the RHS vector exactly matches the $p = 10$ row in Table 1. This system can equivalently be

written as

$$\begin{bmatrix} d_0 \\ d_1 \\ \vdots \\ \vdots \\ \vdots \\ \vdots \\ d_8 \end{bmatrix} = \begin{bmatrix} -\frac{63887}{89600} \\ \frac{427487}{725760} \\ -\frac{3498217}{3628800} \\ \frac{500327}{403200} \\ -\frac{6467}{5670} \\ \frac{2616161}{3628800} \\ -\frac{24019}{80640} \\ \frac{263077}{3628800} \\ -\frac{8183}{1036800} \end{bmatrix} - \begin{bmatrix} 1 & 9 \\ -9 & -80 \\ 36 & 315 \\ -84 & -720 \\ 126 & 1050 \\ -126 & -1008 \\ 84 & 630 \\ -36 & -240 \\ 9 & 45 \end{bmatrix} \begin{bmatrix} d_9 \\ d_{10} \end{bmatrix}. \tag{14}$$

It is from this form clear that we can obtain smaller (in magnitude) values for $d_0, d_1, \ldots, d_8$ by selecting appropriate (also small) values for d_9, d_{10} (rather than leaving both of these as zero; for any values of these coefficients, the accuracy order remains $p = 10$). This concept generalizes to any $N > n$, and offers the opportunity to eliminate the exponential growth in the classical Gregory weights, at the expense of including a few additional weights.

4 Numerical implementation

4.1 Floating point calculation of weight sets

The strategy already outlined is to choose $N > n$, making (11) under-determined, and then look for solutions that, for example, minimize

$$(L_2): \qquad \sum_{k=0}^{N} s^{2k}\, d_k^2. \tag{15}$$

Here, s is a scalar number somewhat larger than one (forcing the $\{d_k\}$ sequence to be mostly decaying in magnitude). The choice of s represents a compromise between (i) s is close to one: The $N+1$ non-trivial weights $w_k,\ k = 0, 1, \ldots, N$ will not converge towards one, and (ii) s large: The first $n+1$ weights will approach a Gregory-type case, with large oscillations. Somewhere in-between, best found by trial-and-error, the weight oscillations will be mild and the weights approach the value of one (before becoming identically one). The under-determined system (11) is somewhat ill-conditioned. Regular double precision is adequate to find weights in schemes up to around order $p = 15$, and quad precision (about 34 decimal digits) easily suffices up to orders well into the 20's.

The Appendix contains a simple MATLAB code for solving the L_2 minimization problem (15) using the *pinv* function (Moore-Penrose pseudo-inverse), available in standard MATLAB as well as in the main extended precision toolboxes (e.g. MATLAB's Symbolic Toolbox and the Advanpix package [1]).

When going to very high orders of accuracy (well beyond $p = 20$), one might prefer to instead minimize using library routines that include options for imposing inequality constraints (such as not allowing any $d_k < -1$, corresponding to no weight $w_k = d_k + 1$ becoming negative). Such routines are available both for L_2 and L_1 minimization (i.e. of minimizing $\sum_{k=0}^{N} s^k\ |d_k|$).

4.2 Weight sets with rational coefficients

Considering for example order $p = 10$ schemes, the Gregory version is given in the last row of Table 1 ($n = N = 8$). After factoring out $1/10$, we can write the corrections d_k as

$$\frac{1}{10}\ \left\{-\frac{63887}{8960},\ \frac{427487}{72576},-\frac{3498217}{362880},\ \frac{500327}{40320},-\frac{6467}{567},\ \frac{2616161}{362880},-\frac{24019}{8064},\ \frac{263077}{362880},-\frac{8183}{103680}\right\}. \tag{16}$$

As well as striving for smaller corrections (in magnitude) and non-negative weights, one can simultaneously search for corrections d_k that involve rational numbers with relatively small numerators and denominators. Again for accuracy order $p = 10$, there are for ex. many $n = 8,\ N = 10$ schemes with no numerator or denominator above 5 and 4 digits, respectively. Two somewhat arbitrarily chosen cases include

$$\frac{1}{504}\ \left\{-\frac{22763}{64},\ \frac{59501}{225},-\frac{64849}{180},\ \frac{11027}{32},-\frac{40069}{225},\frac{6071}{7200},\ \frac{45847}{800},-\frac{40171}{1440},-\frac{289}{2880},\ \frac{2917}{800},-\frac{1957}{2400}\right\}, \tag{17}$$

and

$$\frac{1}{480}\ \left\{-\frac{35351}{105},\ \frac{18751}{80},-\frac{60167}{216},\ \frac{4643}{24},\ \frac{4777}{7560},-\frac{47189}{378},\ \frac{26249}{280},-\frac{17389}{1512},-\frac{29921}{1512},\ \frac{6143}{560},-\frac{6949}{3780}\right\}. \tag{18}$$

One strategy to search for such cases starts by noting that the *least common multiple* of the denominators in (14) is $lcm = 7257600$. It follows then from (14) that, if we choose d_9 and d_{10} to be integer multiples of $1/lcm$, that will also become the form of $d_0, d_1, \ldots, d_8$. A random selection of such values for d_9 and d_{10} will reveal many cases (such as (17) and (18)) when all the rational numbers $d_0, d_1, \ldots, d_8, d_9, d_{10}$ simplify greatly.[7]

4.3 Numerical tests

Figure 4 illustrates five different weight sets based on the discussion above. In none of the cases are there any negative weights present[8].

We consider at first the following two functions, defined over $[0, 1]$:

$$f(x) = \cos(20\sqrt{x})\ , \tag{19}$$

(also used in [9]), and

$$f(x) = e^{-1000(x-\frac{1}{2})^2}, \tag{20}$$

shown in Figures 5 (a,b). Both functions are infinitely differentiable, with $\int_0^1 \cos(20\sqrt{x})dx = \frac{1}{200}(\cos 20 + 20\sin 20 - 1)$ and $\int_0^1 e^{-1000(x-\frac{1}{2})^2}dx = \frac{1}{10}\sqrt{\frac{\pi}{10}}\ \mathrm{Erf}\ (5\sqrt{10})$, respectively. The two functions have quite different character in that the first integrand is most challenging very near the left boundary, whereas the second one mainly tests quadrature accuracy in the domain interior.

[7]Acknowledgment: The rational solutions (17), (18) were found using a Quadro P6000 GPU board, donated by Nvidia.

[8]In the last case of order $p = 20$, corresponding Gregory and NC schemes have weights in the ranges $[-276,\ 273]$ and $[-496,\ 546]$, respectively.

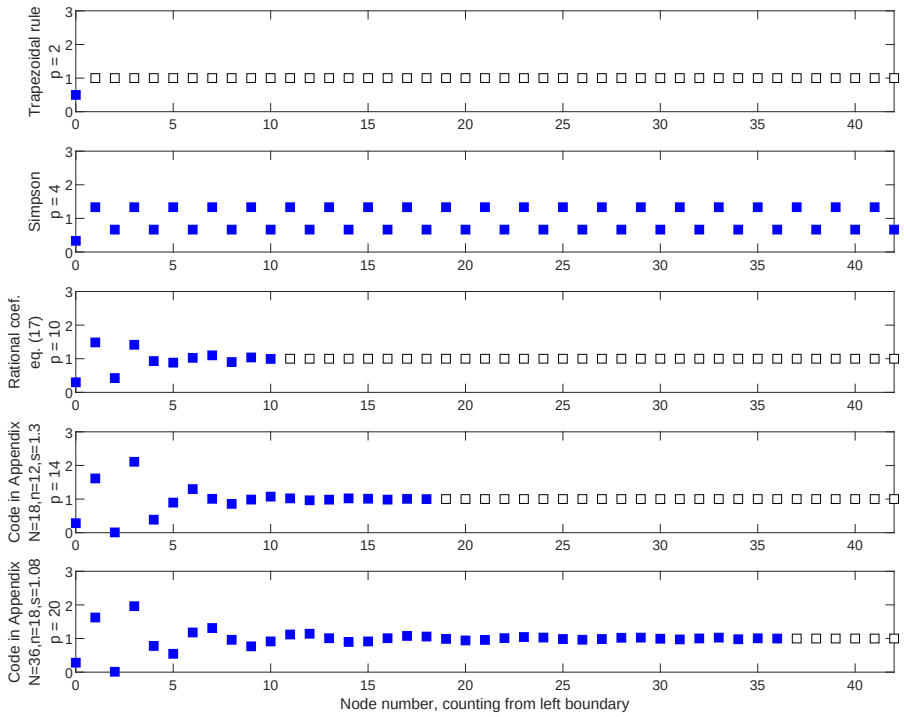


Figure 4: The equi-spaced weights used for the test problems. Non-trivial weights (meaning weights not equal to one) are shown in blue. Only in the Simpson case do these extend past what is illustrated here.

4.4 Some comments on Gaussian quadrature

While this study is focused on quadrature based on equi-spaced nodes (in many applications imposed by the data availability), there also are applications where the nodes can be chosen freely - and thus can be selected purely for the purpose of optimizing quadrature accuracy. In such cases, domain decomposition (adaptive quadrature) is often used. Clustering nodes towards the ends of each sub-interval provides the most common approach for overcoming the Runge phenomenon. In order to place the presently developed TR enhancements in a broader context, we include in the first two tests also Gaussian quadrature and the Clenshaw-Curtis method (which amounts to integrating exactly the Chebyshev polynomial interpolant obtained through a fast cosine transform). These two approaches are surveyed in [15]. A hybrid Gauss-trapezoidal rule is described in [3] (using equi-spaced nodes across most of the domain interior, and clustering only near each end).

4.5 Test results and discussion

4.5.1 First test problem

Figure 6 (a) shows the convergence rates when integrating the first test function (19). Since the steepest gradients occur at a boundary, it is not surprising that Clenshaw-Curtis and Gauss quadrature (dotted curves) excel, since these both cluster their nodes towards the boundaries (however, for the entirely different reason of suppressing the Runge phenomenon). The theoretically known factor-of-two difference between their convergence rates is clearly evident.

Regarding the equi-spaced methods, the higher order ones need increasing numbers of sub-intervals before they can be applied. From the slopes in this log-log display, it is clear that their convergence rates correspond well to their theoretically known orders of accuracy.

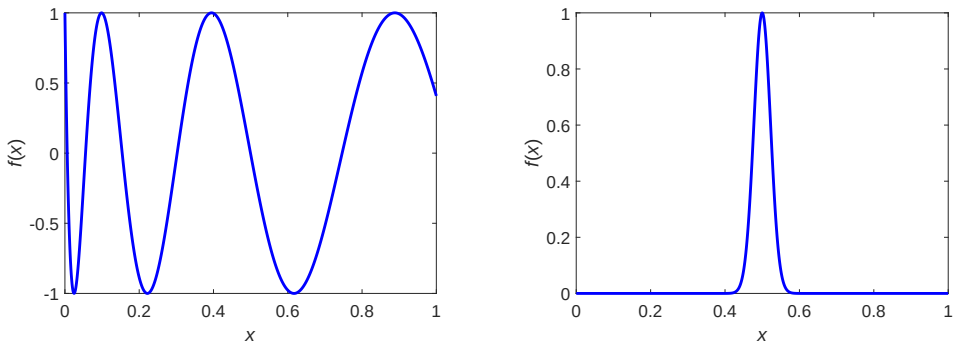
4.5.2 Second test problem

Figure 6 (b), with emphasis of the interval interior, shows that the Clenshaw-Curtis and Gauss quadrature methods both suffer in performance from having their node density in the interior reduced by a factor of $2/\pi$ (as a consequence of having clustered nodes towards the boundaries). In line with the analysis in [15, 19], these two methods perform here similarly at low accuracies, before their asymptotic factor-of-two rate difference enters.

The curves for the equi-spaced TR versions of orders $p = 2$ and $p = 10$ in Figure 6 (b) overlap from an early point on, to be joined shortly thereafter by the $p = 14$ scheme (and eventually, outside the diagram, by the $p = 20$ scheme). The $p = 4$ Simpson method requires about twice as many nodes for comparable accuracy (with higher order Newton-Cotes schemes needing still more; not displayed here).

4.5.3 Third test problem

The integrand in this case is chosen as the sum of the ones in the previous two cases, thus representing a function that is challenging for numerical quadrature both at an end and in the interior. Since our primary interest is quadrature using equi-spaced nodes, the Clenshaw-Curtis and Gauss quadrature schemes have not been included. The results in Figure 6 (c) show that the new higher order schemes decisively outperform both the trapezoidal and Simpson rules. For very large number



(a) $f(x) = \cos(20\sqrt{x})$ (b) $f(x) = e^{-1000(x-\frac{1}{2})^2}$

Figure 5: The two test functions used in Section 4.3.

of nodes, higher order is better. However, the 'robust' performance and simplicity of the $p = 10$ scheme (17) makes it an attractive replacement of the classical trapezoidal and Simpson's rules.

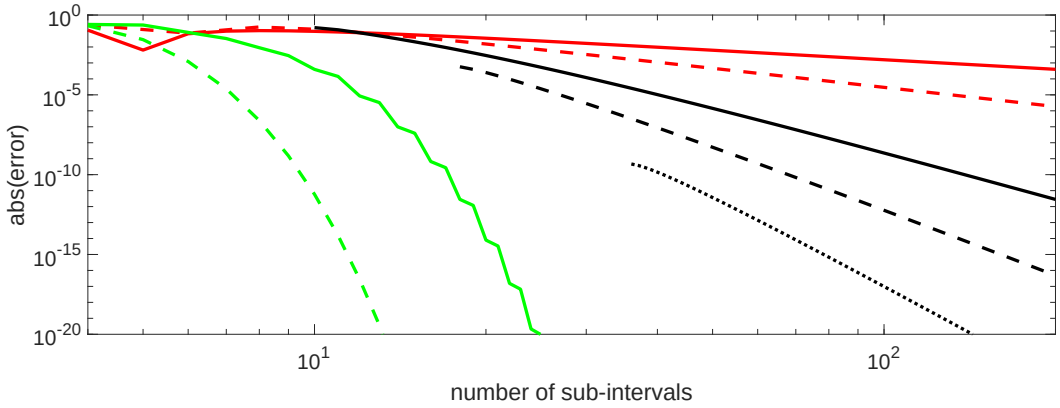
5 Conclusions

Computational algorithms often become more cost-effective when their formal order of accuracy is increased, be it for solving ODEs, PDEs, interpolation, or numerical quadrature. In cases where only equi-spaced (non-periodic) data is available, the well-known Runge phenomenon commonly arises. Traditionally, numerical quadrature methods for this case have been limited to low orders of accuracy. Key features of the higher order methods described here include

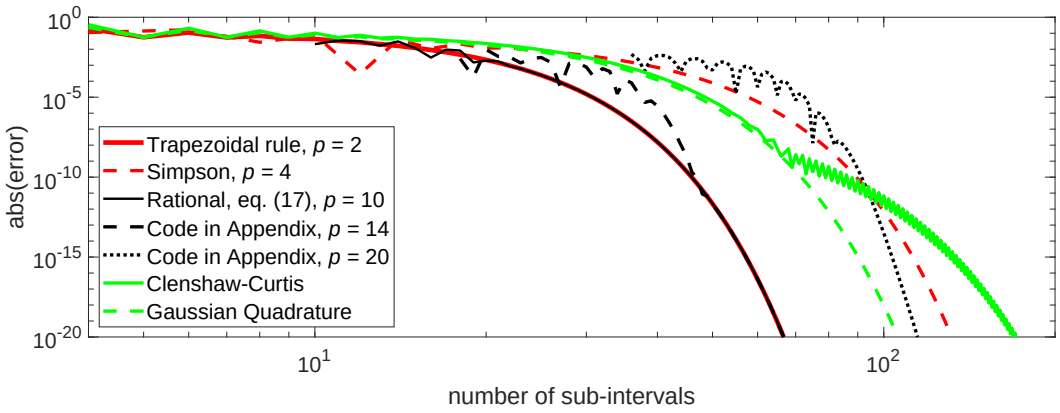
1. All weights are non-negative,
2. Apart from very near to the ends of the interval, the weights are identical to those of the trapezoidal rule (making the scheme fully able to utilize the trapezoidal rule's spectral accuracy over the interior of intervals).

For everyday usage, the scheme (17) is an attractive compromise between simplicity and accuracy (10^{th} order). However, for applications that require results close to standard machine accuracy of 16 digits (and assuming integrands of sufficient smoothness), still higher orders of accuracy may become even more more cost-effective.

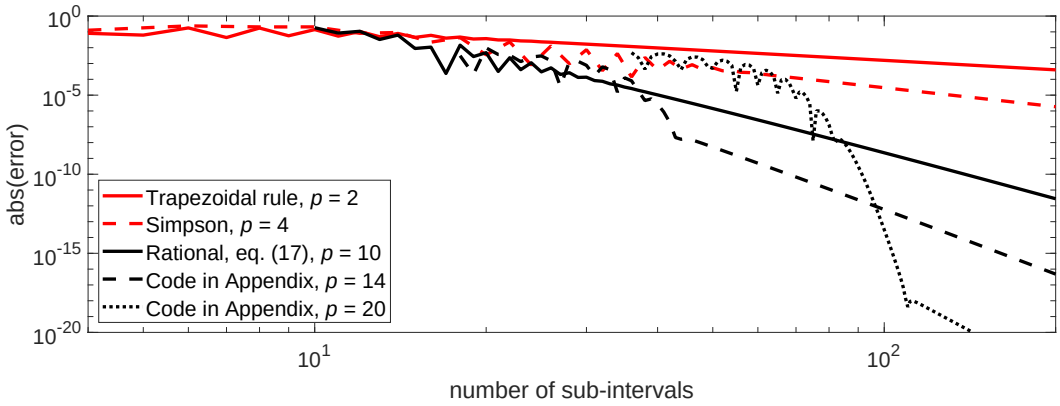
In applications where function data can be obtained at arbitrary spatial locations (rather than only being available at equi-spaced points), a more traditional approach for eliminating the Runge phenomenon is to cluster nodes heavily towards the ends, as in Gaussian quadrature methods. However, doing this depletes nodes throughout the rest of the interval. Certain applications, especially in *Experimental Mathematics*, require integrals to be evaluated to several thousands of decimal places. It is noted in [4] that the *tanh-sinh* method [14] then often is viewed as the best option. In this approach, a variable change designed to reduce errors from the ends of the interval is followed by equi-spaced TR integration.



(a) Errors vs. the number of sub-intervals for the first test problem (for legend, see part (b)).



(b) Errors vs. the number of sub-intervals for the second test problem.



(c) Errors vs. the number of sub-intervals for the third test problem.

Figure 6: Log-log plots of errors vs. the number of sub-intervals for the two test problem in Section 4.3, when using the five different methods illustrated in Figure 4 and also, with non-equi-spaced nodes, Clenshaw-Curtis and Gaussian quadrature results.

References

[1] Advanpix, *Multiprecision Computing Toolbox for MATLAB*, http://www.advanpix.com/, Advanpix LLC., Yokohama, Japan.

[2] B.K. Alpert, *High-order quadrature for integral operators with singular kernels*, J. Comp. Appl. Math. **60** (1995), 367–378.

[3] ———, *Hybrid Gauss-trapezoidal quadrature rules*, SIAM J. Sci. Comput. **20** (1999), no. 5, 1551–1584.

[4] D.H. Bailey and J.M. Borwein, *High-precision numerical integration: Progress and challenges*, J. Symbolic Comp. **46** (2011), 741–754.

[5] B. Fornberg, *Numerical differentiation of analytic functions*, ACM Transactions on Mathematical Software **7** (1981), no. 4, 512–526.

[6] B. Fornberg, *A Practical Guide to Pseudospectral Methods*, Cambridge University Press, 1996.

[7] B. Fornberg and N. Flyer, *A Primer on Radial Basis Functions with Applications to the Geosciences*, SIAM, Philadelphia, 2015.

[8] ———, *Solving PDEs with radial basis functions*, Acta Numerica **24** (2015), 215–258.

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

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

[11] M. Ossendrijver, *Ancient Babylonian astronomers calculated Jupiter's position from the area under a time-velocity graph*, Science **351** (2016), 482–484.

[12] J.A. Reeger and B. Fornberg, *Numerical quadrature over smooth surfaces with boundaries*, J. Comput. Phys. **355** (2018), 176–190.

[13] H.P. Starr, *On the numerical solution of one-dimensional integral and differential equations*, Ph.D. thesis, Yale University, 1991.

[14] H. Takahasi and M. Mori, *Double exponential formulas for numerical integration*, Kyoto University. Research Institute for Mathematical Sciences. Publications **9** (1974), 721–741.

[15] L.N. Trefethen, *Is Gauss quadrature better than Clenshaw-Curtis?*, SIAM Review **50** (2008), no. 1, 67–87.

[16] L.N. Trefethen, *Approximation Theory and Approximation Practice*, SIAM, 2013.

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

[18] J.A.C. Weideman, *Numerical integration of periodic functions: A few examples*, The Americam Math. Monthly **109** (2002), no. 1, 21–36.

[19] J.A.C. Weideman and L.N. Trefethen, *The kink phenomenon in Fejér and Clenshaw-Curtis quadrature*, Numer. Math. **107** (2007), 707–727.

6 Appendix: MATLAB code for finding weights by L_2 minimization

The following shows a MATLAB script that specifies a weight set to be calculated by L_2 minimization and plots the result. This is followed by the function that carries out the calculation.

```
% Main script to test the function  L_2_corr
N = 14;                      % Weight correction set extending over nodes 0:N
n = 10;                      % Match Taylor powers 0:n; accuracy order p = n+2
s = 1.3;                     % Weighting to be used in the minimization
d = L_2_corr(N,n,s);         % Calculate the non-trivial quadrature weights
plot(0:N,(d+1)','bs','MarkerFaceColor','b'); ylim([0,2]); % Display the weights
```

```
function d = L_2_corr(N,n,s)
%  Input parameters
%    N     Stencil extent 0:N
%    n     Taylor coefficients 0:n matching, i.e. order of accuracy p = n+2
%    s     Scaling parameter to control rate by which weights approach one
%  Output parameter
%    d     Column vector with computed corrections

b_k = ones(1,n+2);                                  % Create a column vector with
v = -cumprod(-ones(1,n+1))./(2:n+2);                % the n+1 first Gregory coefficients
for k = 2:n+2                                       % b_k, k=0,1,...,n
    b_k(k) = v(k-1:-1:1)*b_k(1:k-1)';
end
b_k = -b_k';    b_k(1) = [];

P = abs(pascal(N+1,1)');    P = P(1:n+1,1:N+1); % Create the Pascal matrix
sv = cumprod([1,s*ones(1,N)]);    Ps = P./sv;   % Scale its colums
d = pinv(Ps)*b_k;    d = d./sv';                % Find the minimal norm solution
end
```