Available online at www.sciencedirect.com
SCIENCE DIRECT
Theoretical Computer Science 351 (2006) 82–100
Theoretical Computer Science
www.elsevier.com/locate/tcs

Quadrature using 64-bit IEEE arithmetic for integrands over [0,1] with a singularity at 1

M. Hill*, I. Robinson

Department of Computer Science and Computer Engineering, La Trobe University, Victoria 3086, Australia

Abstract

We present a detailed study of some problems encountered when quadrature over [0, 1] is attempted with integrands that have a singularity at 1. Methods designed to increase the accuracy of such quadratures, for example, the application of periodising transformations, are examined in the context of the representational limitations of 64-bit IEEE arithmetic near 1 in [0, 1]. A heuristic is proposed for the forecasting of a lower bound on the irremovable error due to these limitations. We conclude by affirming the commonly accepted procedure that where possible, integrals should be symbolically transformed so that any remaining singularity occurs at 0.

Keywords: Automatic quadrature; Periodising transformations; 64-bit IEEE arithmetic; Extrapolation

1. Introduction

The problems encountered when performing numerical quadrature over a finite interval with an integrand exhibiting endpoint singularities are many and varied. Methods to overcome these problems, both general and specific to particular classes of integrands, have been extensively researched. Comprehensive surveys of such methods appear in [1,2]. In this paper, we discuss the special problems encountered in a 64-bit IEEE arithmetic environment [4] when a quadrature algorithm is used to compute an integral over [0, 1] with the integrand having a singularity at 1 or when the algorithm transforms an integral onto [0, 1] in such a fashion as to result in a singularity at 1. In all cases, we presume that the integral exists.

It is well known that for integrands belonging to certain specified classes, the theoretical quadrature error can be reduced significantly through the use of a periodising transformation *before* application of the quadrature algorithm or with the use of extrapolation *after* (and during) generation of a sequence of quadratures based on geometrically increasing numbers of points. We examine whether these methods, applied separately or in combination, are likely to be successful in also alleviating the rounding error problems near 1. We also examine the effect on the rounding error near 1 of applying general, non-periodising transformations. The investigation leads to the creation of a potentially useful heuristic that estimates the maximum relative accuracy a quadrature algorithm can obtain in determining $\int_a^b g(x)\,\mathrm{d}x$, where $[a, b]$ is a finite interval.

* Corresponding author.
E-mail addresses: hillmj@cs.latrobe.edu.au (M. Hill), i.robinson@latrobe.edu.au (I. Robinson).

0304-3975/$ - see front matter © 2005 Elsevier B.V. All rights reserved.
doi:10.1016/j.tcs.2005.09.057

In the Conclusion, we draw together the threads from the previous sections and offer some general advice on how to approach integrals over finite domains $[a, b]$, where the integrand has a singularity at $x = b$.

2. Preliminary theory, definitions and notation

Let $Ig(a, b) = \int_a^b g(x)\,\mathrm{d}x$ and $Ig(0, 1) = Ig$. Let $Q_N(g; a, b)$ be the result of applying the quadrature rule Q to the integrand g over the interval $[a, b]$ using N abscissae, and let $Q_N g = Q_N(g; 0, 1)$.

2.1. Simple quadrature rules

Let $X_N = \{a = x_0, x_1, \ldots, x_{N-1}, x_N = b\}$, where $x_{j-1} < x_j$, $j = 1, \ldots, N$. A simple quadrature rule to estimate $Ig(a, b)$ is given by the Riemann sum $S_N(g; a, b) = \sum_{j=1}^{N}(x_j - x_{j-1})g(\omega_j)$, $\omega_j \in [x_{j-1}, x_j]$. The following two quadrature rules are special cases of $S_N(g; a, b)$.

Definition 1. The compound rectangle rule is given by

$$Ig(a, b) \cong R_N(g; a, b) = \frac{b-a}{N}\sum_{j=0}^{N-1} g\left(a + j\frac{b-a}{N}\right). \tag{1}$$

$R_N(g; 0, 1)$ is the N-point one-dimensional form of a lattice rule. These otherwise multi-dimensional rules are defined on the domain $[0, 1]^s$, and perform best when the integrand has a smooth 1-periodic extension in all dimensions.

Definition 2. The compound trapezoidal rule is given by

$$Ig(a, b) \cong T_N(g; a, b) = \frac{b-a}{N}\left[\frac{g(a)}{2} + \sum_{j=1}^{N-1} g\left(a + j\frac{b-a}{N}\right) + \frac{g(b)}{2}\right]. \tag{2}$$

The quadrature points of $R_N(g; a, b)$ and $T_N(g; a, b)$ are equally spaced in $[a, b]$. $S_N g$, $R_N g$ and $T_N g$ will denote the special case of $[a, b] \equiv [0, 1]$, using N points.

2.2. Quadrature error

If $g(x)$ and $g'(x)$ exist in $[0, 1]$, the error expansions associated with $T_N g$ and $R_N g$ are given by

$$T_N g - Ig = \sum_{j=1}^{m} \frac{B_{2j}}{(2j)!N^{2j}}\left(g^{(2j-1)}(1) - g^{(2j-1)}(0)\right) + \frac{1}{N^{2m+1}}\int_0^1 P_{2m+1}(Nx)g^{(2m+1)}(x)\,\mathrm{d}x, \tag{3}$$

which is the Euler–Maclaurin-type error expansion for $[0, 1]$, and

$$\begin{aligned} R_N g - Ig = &-\frac{1}{2N}(g(1) - g(0)) + \sum_{j=1}^{m} \frac{B_{2j}}{(2j)!N^{2j}}\left(g^{(2j-1)}(1) - g^{(2j-1)}(0)\right) \\ &+ \frac{1}{N^{2m+1}}\int_0^1 P_{2m+1}(Nx)g^{(2m+1)}(x)\,\mathrm{d}x, \end{aligned} \tag{4}$$

where

(1) $B_{2m} = 2(-1)^{m-1}(2m)!(2\pi)^{-2m}\sum_{k=1}^{\infty}k^{-2m}$, $m = 1, 2, \ldots$, are the Bernoulli numbers, and
(2) $P_{2k+1}(x) = (-1)^{k-1}\sum_{j=1}^{\infty}2(2kj\pi)^{-2j-1}\sin(2\pi kjx)$.

(3) and (4) indicate that there are significant advantages to be gained in terms of low quadrature error if an integrand $g(x)$ has a smooth, 1-periodic extension beyond $[0, 1]$; i.e. $g(x)$ and at least some of its initial successive derivatives are zero at the endpoints of $[0, 1]$. The fewer the number of these non-zero leading summation terms, the smaller the quadrature error is expected to be. One method to achieve this end is to apply a *periodising* transformation.

2.3. Periodising transformations

Definition 3. $\phi(t)$ is a periodising transformation on $[0, 1] \Rightarrow$

1. $\phi(0) = 0, \phi(1) = 1$,
2. $\phi(t)$ is monotonically increasing on $[0, 1]$,
3. $\phi'(t)$ is continuous on $[0, 1]$, and
4. $\phi'(0) = \phi'(1) = 0$. We will assume in this paper that the transformation is symmetrical on $[0, 1]$, with
5. $\phi(t) + \phi(1 - t) = 1$, and
6. $\phi'(t) = \phi'(1 - t)$.

As a means of classification, and a measure of the potential effectiveness of a periodising transformation, we will use the notion of a *damping factor* [7].

Definition 4. $\phi(t)$ has a damping factor of n if in the neighbourhood of $t = 0$, $\phi'(t) \sim At^n$, where A is a non-zero constant.

Thus, using the symmetry of $\phi(t)$,

$$n \text{ is the damping factor of } \phi(t) \Rightarrow \phi^{(j)}(0) = \phi^{(j)}(1) = 0, \quad j = 1, \ldots, n + 1. \tag{5}$$

If $g(x)$ is continuous on $(0, 1)$, and $\phi(t)$ is a periodising transformation with a sufficiently high damping factor, then there exists $r \in \mathrm{Z}^+$ such that for the integrand $f(t) = \phi'(t)g(\phi(t))$,

$$\int_0^1 g(x)\,\mathrm{d}x = \int_0^1 f(x)\,\mathrm{d}x \quad \text{and} \quad \frac{\mathrm{d}^j}{\mathrm{d}t^j} f(t)\bigg|_{t=0,1} = 0, \quad j = 0, \ldots, r. \tag{6}$$

Example 5. $\phi(t) = t^5(70t^4 - 315t^3 + 540t^2 - 420t + 126)$.

$\phi(t)$ is one of the Korobov periodising transformations [5] and has a damping factor of 4. Let $g_1(x) = x^{1/2}$. $g_1(x)$ has a singularity in the derivative at $x = 0$, and $\int_0^1 g_1(x)\,\mathrm{d}x = 2$. If $f_1(t) = \phi'(t)g_1(\phi(t))$, then by (6), we also have $\int_0^1 f_1(t)\,\mathrm{d}t = 2$. However, $f_1(t)$ does not have a singularity in the derivative at $t = 0$. Indeed, for $r = 0, \ldots, 3$, we have $f_1^{(r)}(0) = f_1^{(r)}(1) = 0$.

Example 6. Let $g_2(x) = x^{-1/2}$ and $f_2(t) = \phi'(t)g_2(\phi(t))$, where $\phi(t)$ is the transformation used in Example 5.

In this case, $f_2^{(r)}(0) = f_2^{(r)}(1) = 0$ for $r = 0, 1$, indicating that the 1-periodic extension of $f_2(t)$ is notably less smooth than that for $f_1(t)$. Thus, we would expect that $R_N g$ applied to $f_2(t)$ would return a less accurate quadrature than for $f_1(t)$. This is indeed the case, as illustrated in Table 1.

Our purpose is to demonstrate how measures can be taken to minimise the effects that the numerical limitations of 64-bit IEEE arithmetic can have on the quadrature of periodised integrands. The periodising transformations we will employ to illustrate certain behaviours and limitations inherent in working in a 64-bit IEEE environment will be the Tanh transformation [9], given by

$$\phi(x) = \frac{1}{2}\left[1 + \tanh\left(\frac{1}{1-x} - \frac{1}{x}\right)\right]$$

and

$$\phi'(x) = \frac{1}{2}\left[\frac{1}{(1-x)^2} + \frac{1}{x^2}\right]\operatorname{sech}^{-2}\left(\frac{1}{1-x} - \frac{1}{x}\right),$$

Table 1
Examples 5 and 6 using Q_{CR}, with and without the Korobov transformation

N	g_1	f_1	g_2	f_2
	Rel. error	Rel. error	Rel. error	Rel. error
1	1.00E + 00	1.00E + 00	1.00E + 00	1.00E + 00
2	4.70E − 01	3.05E − 01	6.46E − 01	1.30E − 01
4	2.23E − 01	1.91E − 03	4.29E − 01	2.51E − 02
8	1.07E − 01	5.59E − 05	2.90E − 01	4.30E − 03
16	5.15E − 02	9.04E − 07	1.98E − 01	7.32E − 04
32	2.51E − 02	1.41E − 08	1.37E − 01	1.26E − 04
64	1.23E − 02	2.19E − 10	9.52E − 02	2.21E − 05
128	6.07E − 03	3.41E − 12	6.65E − 02	3.88E − 06
256	3.00E − 03	5.34E − 14	4.66E − 02	6.84E − 07
512	1.49E − 03	7.77E − 16	3.28E − 02	1.21E − 07
1024	7.42E − 04	5.55E − 17	2.31E − 02	2.13E − 08
2048	3.70E − 04	5.55E − 17	1.63E − 02	3.77E − 09

and three of the Sidi transformations [10], which we denote as Sidi6, Sidi8 and Sidi10, where for Sidi$n$,

$$\phi(x) = \frac{1}{C}\int_0^x \sin^n(\pi t)\,\mathrm{d}t \quad \text{and} \quad \phi'(x) = \frac{1}{C}\sin^n(\pi x) \quad \text{with } C = \int_0^1 \sin^n(\pi t)\,\mathrm{d}t,$$

$n = 1, 2, 3, \ldots$. The Sidi$n$ transformations have a damping factor of n, whereas the Tanh transformation is a member of a class of periodising transformations that exhibits (effectively) no upper bound for the damping factor. In the context of real-world applications (and specifically 64-bit IEEE arithmetic), members from this latter class of transformations do not necessarily perform any better than transformations with finite damping factors, as will be demonstrated in subsequent numerical examples.

2.4. Notation

We adopt the *hat* notation to indicate when reference is made to the value of a variable, function or result of a process performed in 64-bit IEEE arithmetic. For example, $\hat{g}$ and $\hat{x}$ will denote the machine arithmetic representations of g and x, respectively, and $\hat{R}_N(g; a, b)$ is the result of evaluating $R_N(g; a, b)$ in 64-bit IEEE arithmetic. $\hat{g}(\hat{x})$ and $\hat{g}(x)$ are treated as equivalent expressions.

For integrand $g(x)$, we define $g^*(x)$, where

$$g^*(x) = \begin{cases} g(x), & g(x) \neq \pm\infty, \\ 0, & g(x) = \pm\infty. \end{cases}$$

In a 64-bit IEEE arithmetic environment, let ε be such that

$$x \in [1-\varepsilon, 1-\varepsilon/2) \Rightarrow \hat{x} = 1-\varepsilon \quad \text{and} \quad x \in [1-\varepsilon/2, 1] \Rightarrow \hat{x} = 1.$$

This ε has the value 2^{-53} and $1-\varepsilon$ is the closest number to 1 in $[0, 1)$ that can be represented in IEEE 64-bit arithmetic. Note that $1-\hat{\varepsilon} = 1-\varepsilon$. 64-bit IEEE arithmetic permits 2^{53} representable numbers in $\left[\frac{1}{2}, 1\right]$, and the representational density (points per unit) for this interval is 2^{54}. In a similar fashion, there are 2^{53} representable numbers in $[(\frac{1}{2})^5, (\frac{1}{2})^4]$, with a corresponding representational density of 2^{58}. As we shall see, many of the problems that are encountered in performing quadratures on integrands with an endpoint singularity stem from the decreasing point density of representable numbers at the endpoint with the singularity, when this endpoint increasingly differs from 0.

2.5. Schema for the numerical examples

In our comparisons of the relative performances of various quadrature rules and periodising transformations when applied to sample integrals, we will use the sequence of quadratures $\{R_N g\}$, where $N = 2^k$, $k = 0, 1, 2, \ldots, 11$,

unless specifically stated otherwise. The schema ensures that the values of any un-transformed abscissae can be exactly represented in 64-bit IEEE arithmetic.

3. Extracting integrand information

We commence this section with an example taken from [8].

Example 7. $g(x) = (1-x)^{-2/3}$, $\int_0^1 g(x)\,\mathrm{d}x = 3$, and $g(x)$ has a singularity at $x = 1$.

Table 2 contains data generated using 64-bit IEEE arithmetic, with $g^*(x)$ used in place of $g(x)$. The 2048 quadrature points of the compound rectangle rule are indexed $j : x_j = j/2048$, $j = 0, \ldots, 2047$ as per the schema in Section 2.5. Weights and abscissae corresponding to each transformation were pre-calculated and stored to the full accuracy that 64-bit IEEE arithmetic permits.

We observe that the quadratures in Table 2 are of relatively low accuracy and, with the exception of those produced when no transformation is applied, become contaminated by rounding error. That these poor results cannot be explained solely by the transformations being of insufficient power is evidenced by Example 8.

Example 8. $g(x) = x^{-2/3}$, $\int_0^1 g(x)\,\mathrm{d}x = 3$, and $g(x)$ has a singularity at $x = 0$.

The integrand in Example 8 is the integrand in Example 7 reflected in $x = \frac{1}{2}$, so that the singularity has been moved from $x = 1$ to $x = 0$. Again, using pre-calculated weights and abscissae and all subsequent calculations being performed in 64-bit IEEE arithmetic, the resulting quadratures are displayed in Table 3. We observe in the data of Table 3

(a) a notable increase in quadrature accuracy for all transformations, and
(b) that rounding error contamination is negligible.

The differences between the data in Tables 2 and 3 are a direct consequence of 64-bit IEEE arithmetic's incapacity to distinctly represent numbers close to 1 in [0, 1] with the same point density that it can near 0. (Even the quadratures determined without any transformation show minor differences between their respective relative errors.)

That such differences exist is well known. However, in the next section, we will examine in detail the mechanisms that operate to produce these quadratures of diminished accuracy. The mathematics of the analysis does not need to reflect some of the slightly more subtle aspects of machine arithmetic. For instance, simplifications such as $\varepsilon a + \varepsilon b = \varepsilon(a + b)$ will be permitted, without unduly compromising any conclusions we draw. Much of Section 3 concentrates

Table 2
$R_N g$ after applying different periodising transformations. $g(x) = (1-x)^{-2/3}$

N	None	Sidi6	Sidi8	Sidi10	Tanh
	Rel. error	Rel. error	Rel. error	Rel. error	Rel. error
1	6.7E − 01	1.0E + 00	1.0E + 00	1.0E + 00	1.0E + 00
2	5.7E − 01	1.5E − 01	3.2E − 02	7.5E − 02	5.8E − 02
4	4.7E − 01	3.0E − 02	1.0E − 03	1.3E − 02	4.1E − 02
8	3.9E − 01	5.6E − 03	1.0E − 07	5.6E − 04	1.1E − 04
16	3.1E − 01	1.1E − 03	1.8E − 12	4.2E − 05	4.7E − 05
32	2.5E − 01	2.2E − 04	3.1E − 10	3.3E − 06	1.8E − 06
64	2.0E − 01	4.3E − 05	1.4E − 08	2.8E − 05	6.4E − 06
128	1.6E − 01	8.6E − 06	2.2E − 05	1.6E − 05	2.1E − 05
256	1.3E − 01	4.4E − 05	1.4E − 05	2.5E − 05	3.3E − 05
512	1.0E − 01	3.0E − 05	1.9E − 05	2.1E − 05	2.8E − 05
1024	8.1E − 02	2.4E − 05	2.2E − 05	2.3E − 05	2.6E − 05
2048	6.4E − 02	2.7E − 05	2.4E − 05	2.4E − 05	2.4E − 05

Table 3
$R_N g$ after applying different periodising transformations. $g(x) = x^{-2/3}$

N	None	Sidi6	Sidi8	Sidi10	Tanh
	Rel. error	Rel. error	Rel. error	Rel. error	Rel. error
1	1.0E + 00	1.0E + 00	1.0E + 00	1.0E + 00	1.0E + 00
2	7.4E − 01	1.5E − 01	3.2E − 02	7.5E − 02	5.8E − 02
4	5.6E − 01	3.0E − 02	1.0E − 03	1.3E − 02	4.1E − 02
8	4.3E − 01	5.6E − 03	1.0E − 07	5.6E − 04	1.1E − 04
16	3.3E − 01	1.1E − 03	6.2E − 14	4.2E − 05	4.7E − 05
32	2.6E − 01	2.2E − 04	0.0E + 00	3.3E − 06	1.8E − 06
64	2.1E − 01	4.3E − 05	0.0E + 00	2.6E − 07	1.8E − 09
128	1.6E − 01	8.6E − 06	0.0E + 00	2.1E − 08	1.2E − 13
256	1.3E − 01	1.7E − 06	0.0E + 00	1.6E − 09	0.0E + 00
512	1.0E − 01	3.4E − 07	0.0E + 00	1.3E − 10	0.0E + 00
1024	8.1E − 02	6.7E − 08	0.0E + 00	1.0E − 11	0.0E + 00
2048	6.4E − 02	1.3E − 08	0.0E + 00	7.9E − 13	0.0E + 00

on matters relating to the interval $[1-\varepsilon, 1]$. The context for this is the evaluation of $R_N g$, where N is sufficiently large that the m points (of this N) that fall within $[1-\varepsilon, 1]$ can be regarded as the m points used in the evaluation of $R_m(g; 1-\varepsilon, 1)$.

3.1. Applying S_m near 1 in [0,1].

Let $g(x)$ be continuous and bounded on $[0, 1]$. Formally, we estimate $\int_{1-\varepsilon}^{1} g(x)\,\mathrm{d}x$ by $S_m(g; 1-\varepsilon, 1)$ using the points $\{x_0, x_1, \ldots, x_m\}$, where $\forall j, x_{j-1} < x_j, x_0 = 1-\varepsilon$, and $x_m = 1$. Specifically, $\int_{1-\varepsilon}^{1} g(x)\,\mathrm{d}x \cong \sum_{j=1}^{m}(x_j - x_{j-1})\, g(x_{j-1})$. We know nothing of the distribution of the quadrature points beyond their ordering. In the context of 64-bit IEEE arithmetic, we proceed as follows:

$$\exists a : \hat{x}_j = \begin{cases} 1-\varepsilon, & 0 \leqslant j < a, \\ 1, & a \leqslant j \leqslant m \end{cases}$$

$$\Rightarrow \sum_{j=1}^{m}(x_j - x_{j-1})g(x_{j-1}) \cong \sum_{j=1}^{a-1}(\hat{x}_j - \hat{x}_{j-1})g(\hat{x}_{j-1}) + (\hat{x}_a - \hat{x}_{a-1})g(\hat{x}_{a-1}) + \sum_{j=a+1}^{m}(\hat{x}_j - \hat{x}_{j-1})g(\hat{x}_{j-1}) = \varepsilon\hat{g}(1-\varepsilon),$$

which is $\hat{R}_1(g; 1-\varepsilon, 1)$. So, *effectively*, $S_m(g; 1-\varepsilon, 1)$ uses only a single integrand value and is therefore independent of m.

3.2. Applying R_m near 1 in [0,1].

Let $g(x)$ be continuous and bounded on $[0, 1]$. We wish to estimate $\int_{1-\varepsilon}^{1} g(x)\,\mathrm{d}x$ using R_m and 64-bit arithmetic. Unlike the circumstances in Section 3.1, the use of R_m implies that we know that the quadrature points are equally spaced in $[1-\varepsilon, 1]$, with $x_0 = 1-\varepsilon$.

$$\int_{1-\varepsilon}^{1} g(x)\,\mathrm{d}x \cong \frac{\varepsilon}{m}\sum_{j=0}^{m-1} g(x_j) \quad \text{where } x_j = 1-\varepsilon+j\frac{\varepsilon}{m}.$$

There are two cases to consider.

m even: $\hat{x}_j = 1-\varepsilon$, for $j = 1, \ldots, m/2-1$ and $\hat{x}_j = 1$ for $j = m/2, \ldots, m$

$$\Rightarrow \frac{\varepsilon}{m}\sum_{j=0}^{m-1} g(\hat{x}_j) = \frac{\varepsilon}{m}\left(\frac{m}{2}\hat{g}(1-\varepsilon) + \frac{m}{2}\hat{g}(1)\right)$$
$$= \hat{R}_2(g; 1-\varepsilon, 1).$$

m odd: $\hat{x}_j = 1-\varepsilon$ for $j = 1, \ldots, (m-1)/2$ and $\hat{x}_j = 1$ for $j = (m-1)/2+1, \ldots, m-1$. Thus,

$$\frac{\varepsilon}{m}\sum_{j=0}^{m-1} g(\hat{x}_j) = \frac{\varepsilon}{m}\left(\frac{m+1}{2}\hat{g}(1-\varepsilon) + \frac{m-1}{2}\hat{g}(1)\right),$$

which approaches $\hat{R}_2(g; 1-\varepsilon, 1)$ as $m \to \infty$. In the limit, the number of required integrand values (in this case, two) is independent of m. Neither here, nor in Section 3.1, does the value of m play any substantial part in determining the quadrature for $[1-\varepsilon, 1]$.

3.3. What is the effect of applying a linear transformation?

Is it possible to use a linear transformation $[1-\varepsilon, 1] \to [0, 1]$, applied automatically via coding (i.e. not "by hand", where it is possible to perform symbolic simplifications) to obtain a more accurate estimate of $\int_{1-\varepsilon}^{1} g(x)\,\mathrm{d}x$ using 64-bit IEEE arithmetic? Proceeding formally, the transformation

$$\int_{1-\varepsilon}^{1} g(x)\,\mathrm{d}x = \int_0^1 \varepsilon g(\varepsilon y + 1 - \varepsilon)\,\mathrm{d}y, \tag{7}$$

and (for instance) $R_N g$ with (for the sake of simplicity) an even number N of quadrature points, allows us to estimate the right side of (7)

$$\begin{aligned}\int_0^1 \varepsilon g(\varepsilon y + 1 - \varepsilon)\,\mathrm{d}y &\cong \frac{\varepsilon}{N}\sum_{j=0}^{N-1} g\left(\varepsilon\frac{j}{N} + 1 - \varepsilon\right)\\ &= \frac{\varepsilon}{N}\sum_{j=0}^{N/2-1} g\left(\varepsilon\frac{j}{N} + 1 - \varepsilon\right) + \frac{\varepsilon}{N}\sum_{j=N/2}^{N-1} g\left(\varepsilon\frac{j}{N} + 1 - \varepsilon\right)\\ &= \frac{\varepsilon}{2}(\hat{g}(1-\varepsilon) + \hat{g}(1)),\end{aligned}$$

which is $\hat{R}_2(g; 1-\varepsilon, 1)$. Similar results follow from applying $R_N g$ (with N odd) or $S_N g$. Thus, it is readily seen that using a simple linear transformation $[1-\varepsilon, 1] \to [0, 1]$ does not improve upon the results in Section 3.1 or 3.2. Moreover, we have in fact *not* changed the set of points at which g is actually evaluated. Such a change would only be possible if $\varepsilon g(\varepsilon y + 1 - \varepsilon)$ were simplified symbolically before the quadrature is performed.

3.4. What is the effect of applying a non-linear transformation?

Consider the transformations $\psi(y)$ such that

$$\int_0^1 g(x)\,\mathrm{d}x = \int_0^1 \psi'(y) g(\psi(y))\,\mathrm{d}y. \tag{8}$$

These transformations form a superset of the periodising transformations of Definition 3. Focusing on the interval $[1-\varepsilon, 1]$, can the application of a non-linear transformation before using a quadrature rule result in a more accurate estimate of $\int_{1-\varepsilon}^{1} g(x)\,\mathrm{d}x$ within a 64-bit IEEE environment? The short answer is "No". A longer answer is "No, mostly". Clearly, $\int_{1-\varepsilon}^{1} g(x)\,\mathrm{d}x = \int_{\psi^{-1}(1-\varepsilon)}^{1} \psi'(y) g(\psi(y))\,\mathrm{d}y$. We assume that the values of ψ and ψ' have been precomputed, and are correct to the accuracy that 64-bit IEEE representation permits. In evaluating $R_N g$, suppose that the m quadrature points closest to 1 (where for the sake of simplicity, m is assumed even) are used to estimate $\int_{\psi^{-1}(1-\varepsilon)}^{1} \psi'(y) g(\psi(y))\,\mathrm{d}y$.

Assume that $\psi^{-1}(1-\varepsilon)$ and $\psi^{-1}(1-\varepsilon/2)$ are determined to full 64-bit IEEE precision before any quadrature is undertaken. By the construction of $R_m(g;1-\varepsilon,1)$, $\exists q : \psi(y_{q-1}) < 1-\varepsilon/2$ and $\psi(y_q) \geqslant 1-\varepsilon/2$, and so (proceeding formally), we have

$$\frac{1-\psi^{-1}(1-\varepsilon)}{m}\sum_{j=0}^{m-1}\psi'(y_j)g(\psi(y_j)) = \frac{1-\psi^{-1}(1-\varepsilon)}{m} \times\left[\sum_{j=0}^{q-1}\psi'(y_j)g(\psi(y_j)) + \sum_{j=q}^{m-1}\psi'(y_j)g(\psi(y_j))\right] \tag{9}$$

$$= \frac{1-\psi^{-1}(1-\varepsilon)}{m}\left[\hat{g}(1-\varepsilon)\sum_{j=0}^{q-1}\psi'(y_j) + \hat{g}(1)\sum_{j=q}^{m-1}\psi'(y_j)\right]. \tag{10}$$

Have we extracted any information about $g(x)$ when $x \in [1-\varepsilon, 1]$ that results in a more accurate estimate of $\int_{1-\varepsilon}^{1} g(x)\,\mathrm{d}x$ than the simple expression $\varepsilon[\hat{g}(1)+\hat{g}(1-\varepsilon)]/2$? Unfortunately, no. 64-bit IEEE arithmetic reduces the integrand in $[1-\varepsilon, 1]$ to

$$\hat{g}(x) = \begin{cases} \hat{g}(1-\varepsilon), & 1-\varepsilon \leqslant x < 1-\varepsilon/2, \\ \hat{g}(1), & 1-\varepsilon/2 \leqslant x \leqslant 1, \end{cases} \tag{11}$$

which carries with it certain consequences. Presume the impossibly best-case scenario wherein the only 64-bit IEEE representational error occurs as in (11). Then

$$\begin{aligned} &\frac{1-\psi^{-1}(1-\varepsilon)}{m}\left[\hat{g}(1-\varepsilon)\sum_{j=0}^{q-1}\psi'(y_j) + \hat{g}(1)\sum_{j=q}^{m-1}\psi'(y_j)\right] \\ &= \hat{g}(1-\varepsilon)\left(\frac{1-\psi^{-1}(1-\varepsilon)}{\psi^{-1}(1-\varepsilon/2)-\psi^{-1}(1-\varepsilon)}\right)\left(\frac{q}{m}\right)\left(\frac{\psi^{-1}(1-\varepsilon/2)-\psi^{-1}(1-\varepsilon)}{q}\sum_{j=0}^{q-1}\psi'(y_j)\right) \\ &\quad + \hat{g}(1)\left(\frac{1-\psi^{-1}(1-\varepsilon)}{\psi^{-1}(1)-\psi^{-1}(1-\varepsilon/2)}\right)\left(\frac{m-q}{m}\right)\left(\frac{\psi^{-1}(1)-\psi^{-1}(1-\varepsilon/2)}{m-q}\sum_{j=q}^{m-1}\psi'(y_j)\right). \end{aligned} \tag{12}$$

Consider the first summand in the right hand side of (12). (Arguments and observations similar to those that follow are also true for the second summand.) For large m (and hence, large q),

$$\frac{(\psi^{-1}(1-\varepsilon/2)-\psi^{-1}(1-\varepsilon))}{q}\sum_{j=0}^{q-1}\psi'(y_j)$$

is approximately $R_q(\psi';\psi^{-1}(1-\varepsilon),\psi^{-1}(1-\varepsilon/2))$ for $\int_{\psi^{-1}(1-\varepsilon)}^{\psi^{-1}(1-\varepsilon/2)}\psi'(y)\,\mathrm{d}y$. Since for increasing m, the ratio $m:q$ approaches the ratio

$$\left[\psi^{-1}(1-\varepsilon),1\right] : \left[\psi^{-1}(1-\varepsilon),\psi^{-1}(1-\varepsilon/2)\right],$$

the product

$$\left(\frac{1-\psi^{-1}(1-\varepsilon)}{\psi^{-1}(1-\varepsilon/2)-\psi^{-1}(1-\varepsilon)}\right)\cdot\left(\frac{q}{m}\right) \to 1.$$

Taking the limit $m \to \infty$, (10) becomes

$$\begin{aligned} &\lim_{m\to\infty}\frac{1-\psi^{-1}(1-\varepsilon)}{m}\left[\hat{g}(1-\varepsilon)\sum_{j=0}^{q-1}\psi'(y_j) + \hat{g}(1)\sum_{j=q}^{m-1}\psi'(y_j)\right] \\ &= \hat{g}(1-\varepsilon)\int_{\psi^{-1}(1-\varepsilon)}^{\psi^{-1}(1-\varepsilon/2)}\psi'(y)\,\mathrm{d}y + \hat{g}(1)\int_{\psi^{-1}(1-\varepsilon/2)}^{1}\psi'(y)\,\mathrm{d}y \\ &= \cdots = \frac{\varepsilon}{2}[\hat{g}(1-\varepsilon)+\hat{g}(1)], \end{aligned} \tag{13}$$

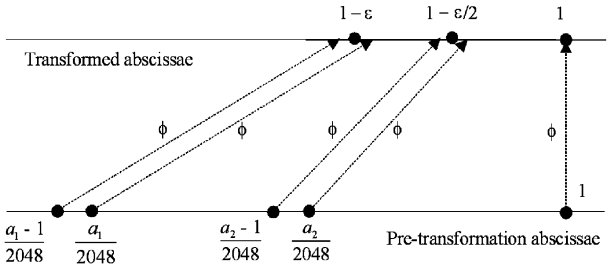

Fig. 1. The meanings of a_1 and a_2 in Table 4. (a) $[1/2, (a_1 - 1)/2048]$: all abscissae distinctly represented. (b) $[a_1/2048, (a_2 - 1)/2048]$: rounding down to $1 - \varepsilon$. (c) $[a_2/2048, 1]$: rounding up to 1.

Table 4
Values of a_1 and a_2 (see Fig. 1) with IEEE 64-bit arithmetic $\varepsilon = 2^{-53}$

ϕ	$\phi^{-1}(1 - 2^{-53})$	a_1	$\phi^{-1}(1 - 2^{-54})$	a_2
Sidi6	$1 - 2^{-8.8275}$	2044	$1 - 2^{-8.9685}$	2044
Sidi8	$1 - 2^{-7.2125}$	2035	$1 - 2^{-7.3236}$	2036
Sidi10	$1 - 2^{-6.1886}$	2020	$1 - 2^{-6.2795}$	2022
Tanh	$1 - 2^{-4.2793}$	1943	$1 - 2^{-4.3051}$	1945

Table 5
Values of $N = 2^k$ for which weights and abscissae of $R_N g$ near 1 lose all significant figures in 64-bit IEEE arithmetic

Transformation	Sidi6	Sidi8	Sidi10	Tanh
	$N \geqslant 512 = 2^9$	$N \geqslant 256 = 2^8$	$N \geqslant 128 = 2^7$	$N \geqslant 32 = 2^5$

which is still $\hat{R}_2(g; 1 - \varepsilon, 1)$, suggesting that the application of a generic non-linear transformation as in (8) normally cannot be expected to improve the estimate of $\int_{1-\varepsilon}^{1} g(x)\,\mathrm{d}x$.

In a sense, this is a worst case result. The analysis does not take into consideration any special characteristics that a particular transformation may possess. For instance, we have not specifically considered periodising transformations that, when applied to an integrand, can remove some or all of the non-remainder terms in (3) and (4). However, the data in Tables 2 and 3 have already suggested that even some very powerful periodising transformations do not overcome the problems that 64-bit IEEE arithmetic encounters with singular values at $x = 1$ rather than $x = 0$, an observation that will be reinforced by data presented later in this paper. Exceptional cases will be dealt with in Section 3.7; thus the answer "No, mostly" at the beginning of this section. As an observation foreshadowing the discussion in Section 3.7, note that in Tables 2 and 3, the performance of the Sidi8 transformation is unexpectedly better than that of both Sidi10 and Tanh.

3.5. Numerical data for the chosen periodising transformations

The results in Sections 3.1–3.4 are consistent with the data in Table 2. Except for Sidi8 (see Section 3.7), none of the transformations can extract any more than about five significant figures of accuracy, even though the respective damping factors of these transformations are markedly different.

To see what is happening near 1 to the values of the weights and abscissae for the transformations chosen, we will again concentrate on the interval $[1 - \varepsilon, 1]$. Keeping in mind that $1 - \varepsilon \equiv 1 - 2^{-53}$ and that the 64-bit IEEE rounding behaviour is as described in Section 2.4.3, the data in Table 4 was collected, where the display format for the values of $\phi^{-1}(1 - \varepsilon) = \phi^{-1}(1 - 2^{-53})$ and $\phi^{-1}(1 - \varepsilon/2) = \phi^{-1}(1 - 2^{-54})$ has been chosen so that comparisons with the value of $1 - \varepsilon = 1 - 2^{-53}$ are made easier. The meanings of a_1 and a_2 used in Table 4 are given in Fig. 1.

From Table 4, we can infer (see Table 5) when a complete loss of accuracy in $[1 - \varepsilon, 1]$ first occurs as a result of the limitations of 64-bit IEEE numerical representation.

The situation near 0 is completely different. Of the chosen Sidi transformations, no weights and abscissae values underflow to 0 if these values are pre-calculated using extended precision arithmetic. In the case of Tanh, the values of the weights and abscissae succumb to 64-bit IEEE underflow only for $j = 1, \ldots, 5$. Thus, in the case of Tanh, we lose (as a result of 64-bit IEEE rounding) only about 0.25% of points to underflow near 0, but 5% of points near 1, rendering integrand evaluation at these latter points a non-productive exercise. If the integrand happens to have an unbounded singularity at 1 (as in Example 7), then this 5% constitutes a significant loss of information in relation to the behaviour of the integrand in a region where such a loss is likely to do the most damage to the accuracy of the quadrature.

Using numbers of abscissae greater than those in Table 5 may result in improvements in the accuracies of some quadratures. The point is that for such quadratures, the improvements in accuracy could well be significantly compromised by the problems associated with $[1 - \varepsilon, 1]$. To emphasize this point we will build upon Example 7, and use for illustration the following family of integrands:

$$h(x) = (1 - x)^{\alpha}, \quad x \in [0, 1], \quad 0 < |\alpha| < 1, \tag{14}$$

with the corresponding symbolic results

$$\int_0^1 h(x)\,\mathrm{d}x = \frac{1}{1+\alpha} \quad \text{and} \quad \int_{1-\varepsilon}^1 h(x)\,\mathrm{d}x = \frac{\varepsilon^{1+\alpha}}{1+\alpha}.$$

When $h(x)$ is coded as $h^*(x)$, (13) becomes $\varepsilon^{1+\alpha}/2$, which is an underestimate of $\int_{1-\varepsilon}^1 h(x)\,\mathrm{d}x$. The relative quadrature error associated with the integral on $[1 - \varepsilon, 1]$ is given by

$$\varphi = (1+\alpha)\left(\frac{\varepsilon^{1+\alpha}}{1+\alpha} - \frac{\varepsilon^{1+\alpha}}{2}\right) = \cdots = \frac{\varepsilon^{1+\alpha}(1-\alpha)}{2}.$$

As $\alpha \to -1$, $\varphi \to 1$, indicating that in the limit, the relative error associated with $[1 - \varepsilon, 1]$ accounts (as expected) for 100% of the total relative error in the quadrature of $\int_0^1 h(x)\,\mathrm{d}x$.

3.6. The minimum relative error for $R_N(g; a, b)$: a heuristic estimate

Consider a quadrature algorithm for the general integral $\int_a^b g(x)\,\mathrm{d}x$, a, b finite, that automatically maps $[a, b]$ onto $[0, 1]$ before implementing its core algorithm. It would be useful if we had a safe, efficient, and transformation-independent method of estimating a lower bound for the relative error that can be achieved by the quadrature algorithm operating in a 64-bit IEEE environment. If, for instance, a user requests a high quadrature accuracy, and a simple calculation involving a small number of integrand evaluations indicates that this accuracy cannot or is very unlikely to be achieved, the algorithm can be automatically terminated.

In working towards such an estimate, we again consider the case of $h(x)$ as in (14), and let

$$\lambda_h = \frac{\varepsilon(h^*(1-\varepsilon) + h^*(1))}{2\int_0^1 h(x)\,\mathrm{d}x} = \cdots = \frac{\varepsilon^{1+\alpha}(1+\alpha)}{2}. \tag{15}$$

λ_h is the quadrature over $[1 - \varepsilon, 1]$ expressed as a proportion of the value of the integral over the entire interval $[0, 1]$. Values of λ_h for $\alpha = -0.9, -0.7, \ldots, 0.9$ are found in Table 6. Although 64-bit IEEE arithmetic has been used to generate these values, the evaluation has been of closed-form symbolic expressions and not the results of actual quadratures being performed. Since in a 64-bit IEEE environment, relative quadrature error is essentially limited to no more than 16 significant figures, we note that the relative quadrature error associated with the interval $[1 - \varepsilon, 1]$ becomes a significant factor only when $\alpha < 0$, approximately.

Of course, for a numerical computation, the value of $Ih = \int_0^1 h(x)\,\mathrm{d}x$ in (15) would not be known. The values $\hat{\lambda}_h$ in Table 7 were calculated using values of $\hat{I}h$ that were determined by applying the rectangle rule, where the relative error has been estimated by the simple expression

$$|(R_n(g) - R_{n/2}(g))/R_n(g)|, \tag{16}$$

Table 6
Closed-form error estimate λ_h associated with $h(x)$ and the interval $[1-\varepsilon, 1]$

α	-0.9	-0.7	-0.5	-0.3	-0.1	0.1	0.3	0.5	0.7	0.9
λ_h	1.3E − 03	2.5E − 06	2.6E − 09	2.4E − 12	2.0E − 15	1.5E − 18	1.2E − 21	8.8E − 25	6.4E − 28	4.6E − 31

Table 7
Quadrature-based relative error estimates associated with $h(x) = (1-x)^\alpha$ and the interval $[1-\varepsilon, 1]$

α	-0.9	-0.7	-0.5	-0.3	-0.1	0.1	0.3	0.5	0.7	0.9
$\hat{\lambda}_h$	1.4E − 03	2.5E − 06	2.6E − 09	2.4E − 12	2.0E − 15	1.5E − 18	1.2E − 21	8.8E − 25	6.4E − 31	4.6E − 31
n	16	8	8	8	8	8	8	8	8	8
β	4.0E − 02	6.9E − 05	2.6E − 08	1.3E − 11	2.7E − 13	5.2E − 16	8.9E − 17	5.6E − 17	1.6E − 16	1.6E − 16

with $g(x) = \phi'(x)h(\phi(x))$, $\phi(x)$ is the Sidi6 transformation, and n is sufficiently large to provide $\hat{I}h$ to 1 significant figure. Using the schema of Section 2.5, the highest relative quadrature accuracy achieved before the onset of rounding error is given in Table 7 by β. The observed values of β reflect the limitations of 64-bit IEEE arithmetic.

Clearly, for $h(x) = (1-x)^\alpha$,

(a) $\hat{\lambda}_h$ can be evaluated using relatively few points.
(b) $\hat{\lambda}_h$ provides a sound estimate of λ_h.
(c) For $\alpha < 0$, we have $\hat{\lambda}_h < \beta$, with this difference being about one or two orders of magnitude.

Observation (c) suggests a heuristic for estimating the maximum relative accuracy that can be obtained by a quadrature method. Assume

(a) $f(x)$ and $f'(x)$ are continuous in (a, b).
(b) $[a, b]$ is a finite interval.
(c) IEEE 64-bit arithmetic is being used, $\varepsilon = 2^{-53}$.
(d) The quadrature rule is a weighted sum of $f(x)$ evaluated in $[a, b]$.
(e) The quadrature algorithm performs the mapping $x \in [a, b] \to y \in [0, 1]$, $x = (b-a)y + a$.
(f) $f(x)$ is coded as $f^*(x)$.

Then an estimate of the maximum relative accuracy obtained by a quadrature method is

$$\hat{\lambda} = \frac{\varepsilon(f^*(1-\varepsilon) + f^*(1))}{2\hat{I}f}, \tag{17}$$

where $\hat{I}f$ is the value of If computed to one significant figure using the error estimate within the quadrature method. It is understood that when $\hat{\lambda} < 10^{-16}$, the heuristic implies that full 64-bit IEEE accuracy is possible.

The rationale for this heuristic is readily explained. For the purposes of this argument, the result (13) establishes an estimate of $\int_{1-\varepsilon}^{1} f(x)\,\mathrm{d}x$. If $f(x)$ has a singularity at $x = 1$ such that $f'(x)$ is one-signed on $[1-\varepsilon, 1]$, then $f(x)$ is coded as $f^*(x)$ and (13) becomes $\varepsilon f^*(1-\varepsilon)/2$. Without loss of generality, let $f(x) \to +\infty$ as $x \to 1$. Since $|\int_{1-\varepsilon}^{1} f(x)\,\mathrm{d}x - \varepsilon f^*(1-\varepsilon)/2| > |\varepsilon f^*(1-\varepsilon)/2|$ (see Fig. 2), let

$$\lambda = \left| \frac{\varepsilon f^*(1-\varepsilon)}{2\int_0^1 f(x)\,\mathrm{d}x} \right| < \left| \frac{\int_{1-\varepsilon}^{1} f(x)\,\mathrm{d}x - (\varepsilon/2) f^*(1-\varepsilon)}{\int_0^1 f(x)\,\mathrm{d}x} \right| = E. \tag{18}$$

It is reasonable to expect, and observed to be the case, that when $f(x)$ has a singularity at $x = 1$ of the type described, E is sufficiently large so as to make a major contribution to the overall relative error in the quadrature. On the other

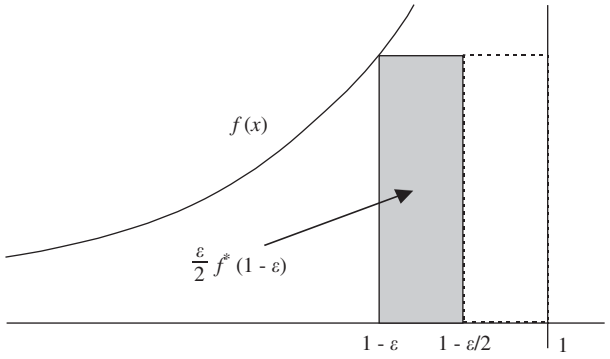

Fig. 2. $f(x) \to +\infty$ as $x \to 1$.

hand, if $f(x)$ is finite and not extremely large at $x = 1$, the contribution of E to the overall relative error is greatly diminished, most often to the point where it is at or below the inherent "noise" level of 64-bit IEEE arithmetic.

In the case of $h(x) = (1 - x)^{\alpha}$, (18) expressed symbolically becomes $\varepsilon^{1+\alpha}(1 + \alpha)/2 < \varepsilon^{1+\alpha}(1 - \alpha)/2$, which is satisfied when $\alpha < 0$. When $\alpha \geqslant 0$, the contribution to the relative quadrature error from $[1 - \varepsilon, 1]$ is sufficiently small as to be insignificant in the context of 64-bit IEEE rounding error.

To illustrate the effectiveness of the heuristic, consider Example 7 for which $\hat{\lambda} = 8.0\text{e}-07$. If, in Table 2, we exclude the Sidi8 data (see Section 3.7), then $\hat{\lambda}$ is indeed a safe upper bound for the cubature accuracy. Integrals T8(1)–T8(8) (see Tables 8 and 9) provide further demonstration of the effectiveness, efficiency and safety of this heuristic. The transformation $x \in [a, b] \to y \in [0, 1]$, $x = (b - a)y + a$ has been used with integral T8(2). It should be noted that although data associated with only one transformation has been supplied in each case, the degree of accuracy of the error bound $\hat{\lambda}$ for the integrals was seen to be independent of the transformation employed.

We point out that for the integrals T8(1)–T8(8), the integrands $f(x)$ are suitably scaled. In [3], the authors illustrate with several numerical examples, the adverse effects on the accuracy of two-dimensional cubatures that can result from the integrand having consistently large values in the domain, as compared to an integrand having moderately sized values except close to a singularity. The advise in [3] is, where possible, to symbolically alter the integrand so as to avoid these consistently large values. This advice is also relevant to the one-dimensional case. There are a number of things to note in the Table 9 data.

- The heuristic is effective, efficient, and but for the case of integral T8(3), accurate.
- Integral T8(3). When $N = 512$, $\hat{\lambda}$ misses as an upper bound for the relative accuracy by about an order of magnitude. However, in practice, it is unlikely that a quadrature algorithm's error estimate could recognise and report this extra accuracy.
- With integrals T8(1)–T8(4) and T8(7), we note the presence of accumulating rounding error that presumably would also be detected by an automatic quadrature algorithm, resulting in the termination of the calculations.
- With integrals T8(5) and T8(8), full machine accuracy is achieved.
- With integral T8(7), the heuristic significantly overestimates the achievable accuracy, at least within the schema of Section 2.5. This is a "safe" error: time is wasted in pursuing the user's requested relative error until the algorithm's internal test for the presence of rounding error terminates the calculations. In such a case, at least the quadrature obtaining the maximum achievable accuracy will most likely be returned.

3.7. When the heuristic fails: an example

There is a serendipitous combination of integrand and transformation that occurs in Examples 7 and 8, as illustrated in Tables 2 and 3. Studying the data therein, one sees that the Sidi8 transformation performs unexpectedly well, particularly when compared to Sidi10 and Tanh. The difference is even more pronounced when the quadratures for Example 7 are performed using 32-figure precision (Table 10). To what can this behaviour be ascribed?

Table 8
Integrals T8(1)–T8(8)

Integral	$[a, b]$	$g(x)$	$\int_a^b g(x)\,\mathrm{d}x$
T8(1)	[0, 1]	$(1-x)^{-1/2}$	2
T8(2)	[0, 3]	$(3-x)^{-9/10}$	$10 \times 3^{1/10}$
T8(3)	[0, 1]	$\sin^3(10\pi x) + (x+2)^4 \cos(20\pi x)$	$(3800\pi^2 - 3)/(20000\pi^4)$
T8(4)	[0, 1]	$\pi(1-x^2)^{-1/2}/2$	$\pi^2/4$
T8(5)	[0, 1]	$\sin(x) + x^2 - \mathrm{e}^{x-2}$	$4/3 + \mathrm{e}^{-2} - \mathrm{e}^{-1} - \cos(1)$
T8(6)	[0, 1]	$x^{-2/3}\log(x)$	-9
T8(7)	[0, 1]	$(1-x)^{-2/3}\log(1-x)$	-9
T8(8)	[0, 1]	$(1-x)^{-1/2}\log(1-x)$	$-\frac{2}{5}$

Table 9
Relative quadrature error for $R_N(g; a, b)$. n is the number of points required to determine $\hat{\lambda}$

N	T8(1) Sidi6	T8(2) Tanh	T8(3) Sidi6	T8(4) Sidi8	T8(5) Sidi6	T8(6) Sidi10	T8(7) Tanh	T8(8) Sidi6
	$\hat{\lambda}$							
	2.6E − 09	1.3E − 03	4.7E − 13	2.4E − 09	2.9E − 16	0.0	8.1E − 06	0.0
	n							
	8	16	128	8	8	8	8	8
1	1.0E + 00	1.0E + 00	1.0E + 00	1.0E + 00	1.0E + 00	1.0E + 00	1.0E + 00	1.0E + 00
2	1.3E − 01	6.3E − 01	3.2E + 03	3.4E − 01	4.5E − 01	7.5E − 01	7.6E − 01	7.6E − 01
4	4.5E − 03	2.9E − 01	1.9E + 03	6.5E − 03	4.2E − 02	3.1E − 03	7.7E − 03	4.7E − 03
8	4.5E − 04	3.3E − 02	1.3E + 02	1.0E − 04	1.5E − 05	2.0E − 03	1.9E − 02	1.1E − 05
16	3.8E − 05	1.5E − 02	3.7E + 02	4.3E − 06	5.8E − 12	2.6E − 04	8.4E − 04	5.4E − 09
32	3.4E − 06	1.4E − 02	8.0E + 02	1.9E − 07	3.7E − 16	2.9E − 05	2.0E − 05	4.1E − 12
64	3.0E − 07	3.1E − 02	8.2E − 08	8.2E − 09	2.2E − 17	2.9E − 06	8.6E − 05	3.2E − 15
128	2.6E − 08	4.1E − 02	1.2E − 12	8.2E − 08	2.2E − 17	2.8E − 07	2.5E − 04	5.6E − 17
256	2.7E − 07	4.5E − 02	7.3E − 13	4.5E − 08	2.2E − 17	2.6E − 08	3.7E − 04	5.6E − 17
512	1.6E − 07	4.3E − 02	2.2E − 14	7.6E − 08	2.2E − 17	2.4E − 09	3.2E − 04	5.6E − 17
1024	1.2E − 07	4.2E − 02	1.3E − 13	9.4E − 08	2.2E − 17	2.1E − 10	3.0E − 04	5.6E − 17
2048	1.4E − 07	4.1E − 02	1.8E − 13	1.0E − 07	2.2E − 17	1.9E − 11	2.8E − 04	5.6E − 17

Table 10
Example 7. $\int_0^1 (1-x)^{-2/3}\,\mathrm{d}x = 3$. 32-figure precision. Q_{CR}

N	None	Sidi6	Sidi8	Sidi10	Tanh
	Rel. error	Rel. error	Rel. error	Rel. error	Rel. error
1	6.67E − 01	1.00E + 00	1.00E + 00	1.00E + 00	1.00E + 00
2	5.69E − 01	1.53E − 01	3.24E − 02	7.51E − 02	5.83E − 02
4	4.73E − 01	3.02E − 02	1.02E − 03	1.29E − 02	4.09E − 02
8	3.87E − 01	5.61E − 03	1.02E − 07	5.61E − 04	1.08E − 04
16	3.13E − 01	1.10E − 03	6.16E − 14	4.24E − 05	4.73E − 05
32	2.52E − 01	2.19E − 04	9.02E − 20	3.31E − 06	1.80E − 06
64	2.01E − 01	4.34E − 05	2.94E − 25	2.60E − 07	1.76E − 09
128	1.61E − 01	8.61E − 06	1.08E − 30	2.05E − 08	8.21E − 12
256	1.28E − 01	1.71E − 06	5.35E − 32	1.61E − 09	1.83E − 10
512	1.02E − 01	3.39E − 07	1.60E − 30	1.27E − 10	1.07E − 10
1024	8.08E − 02	6.72E − 08	2.09E − 28	1.00E − 11	7.64E − 11
2048	6.42E − 02	1.33E − 08	1.23E − 26	8.55E − 11	9.61E − 11

The answer lies in the expansion of $g(x) = \phi_n'(x) f(\phi_n(x))$, where $\phi_n(x)$ is Sidi$n$, $n = 2, 3, \ldots$, and $f(x) = (1-x)^{-2/3}$. Following [10], it is straightforward to derive

Near 1:

$$x \to 0^+ \Rightarrow g(1-x) \cong \sum_{i=0}^{\infty} a_i x^{2i+(n-2)/3}, \tag{19}$$

Near 0:

$$x \to 0^+ \Rightarrow g(x) \cong \begin{cases} \sum_{i=0}^{\infty} b_i x^{n+2i}, & n \text{ odd}, \\ \sum_{i=0}^{\infty} c_i x^{n+2i} + \sum_{i=0}^{\infty} d_i x^{2n+1+2i}, & n \text{ even}, \end{cases} \tag{20}$$

where $n = 2, 3, 4, 5, \ldots$, and the a_i, b_i, c_i and d_i are constants.

Assuming that we are permitted to study the behaviour of $g^{(m)}(x)$ at the endpoints of [0, 1] by differentiating (19) and (20) term-by term m times, we proceed in the context of (4) by treating $g^{(2j-1)}(x)$ (i.e. *odd*-numbered derivatives) at the endpoints of [0, 1] as separate cases.

Near 1:

$$g^{(2j-1)}(1-x) = \sum_{i=0}^{\infty} a_i \frac{\mathrm{d}^{2j-1}}{\mathrm{d}x^{2j-1}} x^{2i+(n-2)/3} = a_0 \frac{\mathrm{d}^{2j-1}}{\mathrm{d}x^{2j-1}} x^{(n-2)/3} + \sum_{i=1}^{\infty} a_i \frac{\mathrm{d}^{2j-1}}{\mathrm{d}x^{2j-1}} x^{2i+(n-2)/3}. \tag{21}$$

Clearly, for $2j-1 < (n-2)/3$, $g^{(2j-1)}(1) = 0$. However, if $n \equiv 2 \bmod 6$, then (19) is a summation of *even* integer powers of x, and thus $\forall j$, $g^{(2j-1)}(1) = 0$. Note that this includes the case $n = 8$.

Near 0, *n odd*:

$g(x) = \sum_{i=0}^{\infty} b_i x^{n+2i} = b_0 x^n + \sum_{i=1}^{\infty} b_i x^{n+2i}$, and thus

$$g^{(2j-1)}(x) = (-1)^{2j-1} b_0 \frac{\mathrm{d}^{2j-1}}{\mathrm{d}x^{2j-1}} x^n + (-1)^{2j-1} \sum_{i=1}^{\infty} b_i \frac{\mathrm{d}^{2j-1}}{\mathrm{d}x^{2j-1}} x^{n+2i}, \tag{22}$$

indicating that $g^{(2j-1)}(x)$ near 0 can be expressed as a sum of *even* powers of x. Additionally, when $2j-1 < n$, we have $g^{(2j-1)}(0) = 0$.

Near 0, *n even*:

$g(x) = \sum_{i=0}^{\infty} c_i x^{n+2i} + \sum_{i=0}^{\infty} d_i x^{2n+1+2i}$.

The first sum is of *even* powers of x, and the second sum is of *odd* powers of x. So,

$$g^{(2j-1)}(x) = (-1)^{2j-1} \left[\sum_{i=0}^{\infty} c_i \frac{\mathrm{d}^{2j-1}}{\mathrm{d}x^{2j-1}} x^{n+2i} + d_0 x^{2n+1} + \sum_{i=1}^{\infty} d_i x^{2n+1+2i} \right]. \tag{23}$$

The first summation in (23) is of *odd* powers of x and thus

$$\forall j, \quad \sum_{i=0}^{\infty} c_i \frac{\mathrm{d}^{2j-1}}{\mathrm{d}x^{2j-1}} x^{n+2i} \bigg|_{x0} = 0.$$

The second summation in (23) is of *even* powers of x. Clearly, $g^{(2j-1)}(0) = 0$ when $2j-1 < 2n$. This suggests that in the case at hand, *even*-order Sidi transformations will perform better than *odd*-order ones, a result proved more generally in [10]. Combining all these results, we see that the Sidi transformations of order n will perform optimally for $f(x) = (1-x)^{-2/3}$ and $f(x) = x^{-2/3}$ when $n \in \{2, 8, 14, 20, \ldots\}$, which is consistent with the data in Tables 2 and 3.

Example 9. $\int_0^1 (1-x)^{-6/7}\,\mathrm{d}x = 7$.

This example (see Table 11) is contrived so that when using the Sidi6 transformation, all the fractional indices in the expansions at $x = 0$ and $x = 1$ vanish, as was the case with $f(x) = (1-x)^{-2/3}$ and Sidi8. Thus, it is reasonable to expect that the quadratures determined using the Sidi6 transformation would benefit in accuracy. However, it does

Table 11
Example 9. $\int_0^1 (1-x)^{-6/7}\,\mathrm{d}x = 7$. 16-figure precision. $R_N f$

N	None	Sidi6	Sidi8	Sidi10	Tanh
	Rel. error	Rel. error	Rel. error	Rel. error	Rel. error
1	8.57E − 01	1.00E + 00	1.00E + 00	1.00E + 00	1.00E + 00
2	7.99E − 01	5.86E − 01	5.27E − 01	4.74E − 01	4.82E − 01
4	7.37E − 01	2.99E − 01	2.13E − 01	1.46E − 01	1.46E − 01
8	6.74E − 01	1.49E − 01	8.70E − 02	4.91E − 02	1.61E − 02
16	6.14E − 01	7.45E − 02	3.57E − 02	1.65E − 02	3.26E − 03
32	5.58E − 01	3.73E − 02	1.46E − 02	5.54E − 03	1.49E − 03
64	5.06E − 01	1.86E − 02	6.00E − 03	1.20E − 02	6.85E − 03
128	4.59E − 01	9.32E − 03	1.06E − 02	9.10E − 03	1.02E − 02
256	4.16E − 01	1.40E − 02	8.46E − 03	1.06E − 02	1.21E − 02
512	3.77E − 01	1.16E − 02	9.57E − 03	9.88E − 03	1.12E − 02
1024	3.41E − 01	1.05E − 02	1.01E − 02	1.03E − 02	1.08E − 02
2048	3.09E − 01	1.11E − 02	1.04E − 02	1.05E − 02	1.05E − 02

Table 12
Example 9. $\int_0^1 (1-x)^{-6/7}\,\mathrm{d}x = 7$. 32-figure precision. $R_N f$

N	None	Sidi6	Sidi8	Sidi10	Tanh
	Rel. error	Rel. error	Rel. error	Rel. error	Rel. error
1	8.57E − 01	1.00E + 00	1.00E + 00	1.00E + 00	1.00E + 00
2	7.99E − 01	5.86E − 01	5.27E − 01	4.74E − 01	4.82E − 01
4	7.37E − 01	2.99E − 01	2.13E − 01	1.46E − 01	1.46E − 01
8	6.74E − 01	1.49E − 01	8.70E − 02	4.91E − 02	1.61E − 02
16	6.14E − 01	7.45E − 02	3.57E − 02	1.65E − 02	3.25E − 03
32	5.58E − 01	3.73E − 02	1.46E − 02	5.54E − 03	1.79E − 04
64	5.06E − 01	1.86E − 02	6.00E − 03	1.86E − 03	2.76E − 06
128	4.59E − 01	9.32E − 03	2.46E − 03	6.27E − 04	2.78E − 05
256	4.16E − 01	4.66E − 03	1.01E − 03	2.11E − 04	7.37E − 05
512	3.77E − 01	2.33E − 03	4.14E − 04	7.10E − 05	5.48E − 05
1024	3.41E − 01	1.16E − 03	1.70E − 04	2.39E − 05	4.65E − 05
2048	3.09E − 01	5.82E − 04	6.96E − 05	5.18E − 05	5.10E − 05

not appear that the supposed suitability of the Sidi6 transformation has manifested itself in the collected data. Neither the 64-bit IEEE environment (Table 11) nor 32-digit accuracy (Table 12) reveal any obvious advantages in using the combination. It is not until Section 4, where we broadly discuss extrapolation methods, that the effectiveness of the Sidi6 transformation and Example 9 can be seen. As an observation to conclude this subsection, we note that in a similar fashion to the relationship between Example 7 and Example 8, there is an alternative form of Example 9.

Example 10. $\int_0^1 x^{-6/7}\,\mathrm{d}x = 7$.

The resulting data (Table 13) demonstrate that an appropriately chosen periodising transformation will deliver full 16-figure accuracy. In this case, Tanh performs significantly better than Sidi6, indicating that there are far more powerful factors at work here than the subtle interplay between transformation and integrand that characterised Example 10 and Sidi8. Although there are cases where one might predict on purely theoretical grounds that a particular integral and periodising transformation pairing will perform very well, the first step should be to ensure that the integral is expressed in a form that is likely to be most conducive to evaluation by numerical algorithm in an IEEE 64-bit environment.

Table 13
Example 10. $\int_0^1 x^{-6/7}\,dx = 7$. 16-figure precision. $R_N f$

N	None	Sidi6	Sidi8	Sidi10	Tanh
	Rel. error	Rel. error	Rel. error	Rel. error	Rel. error
1	1.00E + 00	1.00E + 00	1.00E + 00	1.00E + 00	1.00E + 00
2	8.71E − 01	5.86E − 01	5.27E − 01	4.74E − 01	4.82E − 01
4	7.72E − 01	2.99E − 01	2.13E − 01	1.46E − 01	1.46E − 01
8	6.92E − 01	1.49E − 01	8.70E − 02	4.91E − 02	1.61E − 02
16	6.23E − 01	7.45E − 02	3.57E − 02	1.65E − 02	3.25E − 03
32	5.62E − 01	3.73E − 02	1.46E − 02	5.54E − 03	1.79E − 04
64	5.08E − 01	1.86E − 02	6.00E − 03	1.86E − 03	2.48E − 06
128	4.60E − 01	9.32E − 03	2.46E − 03	6.27E − 04	5.42E − 09
256	4.16E − 01	4.66E − 03	1.01E − 03	2.11E − 04	1.33E − 13
512	3.77E − 01	2.33E − 03	4.14E − 04	7.10E − 05	0.00E + 00
1024	3.41E − 01	1.16E − 03	1.70E − 04	2.39E − 05	0.00E + 00
2048	3.09E − 01	5.82E − 04	6.96E − 05	8.04E − 06	0.00E + 00

Table 14
Extrapolated relative error using the ε-algorithm. No transformations applied. $T_N(g; a, b)$. 64-bit IEEE arithmetic

N	T8(1)	T8(2)	T8(3)	T8(4)	T8(5)	T8(6)	T8(7)	T8(8)
1	7.5E − 01	9.5E − 01	2.5E + 03	6.8E − 01	1.9E − 01	1.0E + 00	1.0E + 00	1.0E + 00
4	3.9E − 02	3.7E − 01	2.6E + 03	3.1E − 02	3.9E − 05	2.5E + 00	2.5E + 00	2.3E − 02
16	2.9E − 04	5.2E − 03	2.5E + 00	1.1E − 04	9.0E − 09	3.9E + 00	3.9E + 00	3.1E − 04
64	4.2E − 07	8.8E − 06	5.9E − 03	4.7E − 06	2.3E − 14	9.3E − 03	9.3E − 03	6.1E − 07
256	8.4E − 11	2.0E − 09	8.5E − 03	1.0E − 09	2.7E − 16	1.9E − 05	1.9E − 05	7.4E − 09
1024	8.5E − 11	2.0E − 09	4.7E − 06	9.3E − 10	9.2E − 11	4.2E − 09	4.2E − 09	8.6E − 13
ξ_k	2.3E − 02	4.7E − 01	3.1E − 04	2.1E − 02	1.9E − 07	2.9E − 01	2.9E − 01	1.2E − 04
2^k	1024	1024	1024	1024	1024	1024	1024	1024

Table 15
Extrapolated relative error using the ε-algorithm. $R_N(g; a, b)$. 64-bit IEEE arithmetic

N	T8(1)	T8(2)	T8(3)	T8(4)	T8(5)	T8(6)	T8(7)	T8(8)
	Sidi6	Tanh	Sidi6	Sidi8	Sidi6	Sidi10	Tanh	Sidi6
1	1.0E + 00	1.0E + 00	1.0E + 00	1.0E + 00	1.0E + 00	1.0E + 00	1.0E + 00	1.0E + 00
4	1.7E − 02	2.8E + 00	2.4E + 03	7.4E − 02	8.1E − 02	1.1E + 00	1.1E + 00	2.3E − 01
16	1.0E − 05	8.0E + 00	1.0E + 03	2.9E − 06	3.9E − 04	2.7E − 04	1.1E − 03	3.1E − 07
64	2.5E − 09	2.7E − 02	3.1E + 03	1.7E − 09	2.0E − 12	1.7E − 07	4.5E − 05	1.3E − 13
256	2.5E − 09	4.3E − 02	1.1E + 02	6.0E − 08	2.0E − 12	1.2E − 14	6.2E − 04	1.3E − 13
1024	1.7E − 07	4.2E − 02	1.7E + 02	5.9E − 08	2.0E − 12	4.6E − 12	2.9E − 04	5.9E − 11
ξ_k	2.6E − 09	1.4E − 02	2.2E − 14	8.2E − 09	2.2E − 17	2.1E − 10	2.0E − 05	5.6E − 17
2^k	128	32	512	64	64	1024	32	128

4. Extrapolation

Heretofore, we have not discussed the role that extrapolation techniques can play in extracting results of high accuracy from sequences of quadratures. As seen from the data in Tables 14 and 15 (using integrals T8(1)–T8(8)), even an unsophisticated application of an extrapolation method, in this case the ε-algorithm (e.g. see [6]), can significantly improve quadrature accuracy in a number of cases. For this simple implementation, errors along the diagonal are reported.

The ε-algorithm presupposes that the quadrature rule gives equal weights to the endpoints of the interval, something that is not true of $R_N(g; a, b)$. Thus, $T_N(g; a, b)$ was employed to generate the contents of Table 14, wherein no periodising transformations were applied. However, in the context of the types of integrals discussed in this section, and the characteristics of the periodising transformations illustrated with these examples, there is no effective difference between $R_N(g; a, b)$ and $T_N(g; a, b)$. As a result, $R_N(g; a, b)$ has been used to generate the data in Table 15.

The simplified implementation of the ε-algorithm employed in these examples delivers an extrapolated quadrature at 1024 points, the next extrapolant requiring 4096 points: outside the parameters of the data schema Section 2.5. As a result, the last lines of Tables 14 and 15 contain the relative errors ξ_k, $k \in \{0, 2, \ldots, 10\}$, for the most accurate non-extrapolated quadratures observed using up to and including a maximum of 1024 points.

The sequences of quadratures $\{Q_{2^k} g\}$, $k = 0, 1, \ldots, 10$, used as examples in Section 4 have (with one exception) been calculated using 64-bit IEEE arithmetic. Thus, when no periodising transformation has been applied, the relative error in a given quadrature arises from two sources:

(a) the error inherent in the quadrature rule itself, and
(b) the additional error introduced when 64-bit IEEE arithmetic is used in the application of the quadrature rule.

When a periodising transformation has been applied, there is an additional source of error:

(c) the error in the representation of the true abscissae values.

For extrapolation to be most effective, quadrature errors whose sources are (b) and (c) must be avoided if at all possible. However, we have already seen that there are circumstances where this is not possible. Extrapolation presumes that the expansion of $Q_{2^k}(g; a, b) - I(g; a, b)$ has one of a small number of very specific forms. Yet if g is the result of the application of a periodising transformation, then the expansion of $Q_{2^k}(g; a, b) - I(g; a, b)$ may not conform to one of these forms or may become rather complicated when it does. The Sidi transformations generally leave the error expansion in a state where extrapolation is theoretically justified. On the other hand, the Tanh transformation most often removes all but the remainder terms in (3) and (4), so extrapolation on whatever is left is somewhat problematic. Moreover, if the numerical instantiation of the expansion form, whatever that form might be, is corrupted by errors originating from (b) and (c), then the theoretical benefits that extrapolation offers can be lost.

Table 14 is based on data collected using our simple implementation of the ε-algorithm applied to the quadratures generated by the compound trapezoidal rule, where no periodising transformation has been applied to the integrands. Although not displayed, it should be noted that the quadrature sequences for all these examples contain no overt evidence for the presence of rounding error. Extrapolation improves the accuracy of the original quadratures in all cases.

Table 15 is based on data collected using our simple implementation of the ε-algorithm applied to quadratures generated by the compound rectangle rule, where the specified transformations have been applied to the integrands. The quadrature sequences are the same as those used to generate the data in Table 9, where the presence of rounding error was evident in a number of cases.

In Table 15, accumulating rounding error is evident in all but one example. There is only a single case where an extrapolated result provides more accuracy than does some member of the corresponding quadrature sequence. In this limited note, it is not possible to delineate what proportions of the observed errors can be attributed to the increased complexity of the error expansions on the one hand, and the representational limitations of 64-bit IEEE arithmetic on the other. However, some insight can be gained by returning to the quadrature of Example 9, created to illustrate the theoretical prediction that this integral paired with the Sidi6 has the potential to return accurate quadratures.

Clearly from Table 16, although Sidi6 delivers the most accurate results of the transformations used, the best results are achieved with extrapolation when no transformation is involved. In addition, there is evident rounding error in all cases where a periodising transformation has been applied.

Table 17 contains data extrapolated from quadrature sequences that were calculated using arithmetic carrying 32 significant figures of accuracy. Except in the case of the Tanh transformation (as mentioned earlier, not a transformation

Table 16
Example 9. $\int_0^1 (1-x)^{-6/7}\,\mathrm{d}x = 7$. Extrapolated relative error using the ε-algorithm and 64-bit IEEE arithmetic. $R_N(g;a,b)$ used with the specified transformations and $T_N(g;a,b)$ with no transformation

N	None	Sidi6	Sidi8	Sidi10	Tanh
	Rel. error	Rel. error	Rel. error	Rel. error	Rel. error
1	8.57E − 01	1.00E + 00	1.00E + 00	1.00E + 00	1.00E + 00
4	1.60E + 00	3.53E − 01	4.07E − 01	4.01E − 01	4.82E − 01
16	1.81E − 01	1.65E − 03	2.37E − 04	5.85E − 04	6.01E + 00
64	2.97E − 03	3.87E − 07	5.93E − 06	5.71E − 04	6.24E − 03
256	5.46E − 06	3.86E − 07	2.33E − 04	1.09E − 02	1.43E − 02
1024	1.24E − 09	2.80E − 05	1.04E − 02	1.01E − 02	1.09E − 02

Table 17
Example 9. $\int_0^1 (1-x)^{-6/7}\,\mathrm{d}x = 7$. Extrapolated relative error using the ε-algorithm and 32-figure accuracy. $R_N(g;a,b)$ used with the specified transformations and $T_N(g;a,b)$ with no transformation

N	None	Sidi6	Sidi8	Sidi10	Tanh
	Rel. error	Rel. error	Rel. error	Rel. error	Rel. error
1	8.57E − 01	1.00E + 00	1.00E + 00	1.00E + 00	1.00E + 00
4	1.60E + 00	3.53E − 01	4.07E − 01	4.01E − 01	4.82E − 01
16	1.81E − 01	1.65E − 03	2.37E − 04	5.85E − 04	6.01E + 00
64	2.97E − 03	3.51E − 07	1.99E − 06	1.97E − 06	7.62E − 05
256	5.46E − 06	5.36E − 20	1.40E − 09	6.36E − 09	1.07E − 04
1024	1.24E − 09	5.83E − 28	3.68E − 14	1.32E − 14	5.01E − 05

likely to produce good extrapolated results), there is no evidence of accumulating rounding error amongst those quadratures where a transformation was employed. Moreover, the predicted advantage of the Sidi6 transformation with this particular integrand is now abundantly evident in the data. The theoretical prediction is correct, but 64-bit IEEE arithmetic does not retain sufficient accuracy over the course of performing the quadratures to reveal the advantages of this integrand-transformation pairing.

5. Conclusion

It is commonly known that the representational limitations of 64-bit IEEE arithmetic significantly affect the accuracy of quadratures for integrands on [0,1] with a singularity at 1. This paper has looked in some detail at two methods used to improve the accuracy of such quadratures: periodising transformations and extrapolation.

Periodising transformations act before quadrature. Extrapolation acts after quadrature. In the middle is 64-bit IEEE arithmetic. Transformations are generally not effective in circumventing the 64-bit IEEE representation problems near 1 and 64-bit IEEE arithmetic introduces a sufficient number of errors (for these types of integrals) into the quadratures that the effectiveness of extrapolation is compromised.

As illustrated in the numerical examples throughout the paper, the choice seems to be between

(a) no transformation + extrapolation, or
(b) transformation + no extrapolation.

In practical situations, the heuristic developed in Section 3.7 is accurate, efficient and effective. It can be used to detect when the user-requested relative error is unlikely to be attained. This heuristic, coupled with a test (which most algorithms have) for the existence of accumulating rounding error, can save execution time by halting the algorithm before large numbers of integrand evaluations have been performed.

Several examples have demonstrated that where possible, the best strategy is to symbolically alter the integral so that any singularity in the domain occurs closest to the point of highest representational density. The success of this strategy will depend on exactly how the automatic algorithm handles the internal mappings from one domain to another.

For integrals over [0, 1] that have a singularity at 1, symbolic alteration of the integral prior to use of the quadrature algorithm remains the preferred option.

Acknowledgements

We wish to thank the referees for their valuable suggestions and contributions.

References

[1] P.J. Davis, P. Rabinowitz, Methods of Numerical Integration, Academic Press, London, 1984.
[2] G. Evans, Practical Numerical Integration, Wiley, Chichester, 1993.
[3] M. Hill, I. Robinson, A comparison of strategies for the automatic computation of two-dimensional integrals over infinite domains, Numer. Algorithms 34 (2003) 325–338.
[4] IEEE Standard 754-1985, IEEE Standard for Binary Floating-Point Arithmetic, IEEE Computer Society, 1985.
[5] N.M. Korobov, Number-Theoretic Methods of Approximate Analysis, Fizmatgiz, Moscow, 1963.
[6] A. Krommer, C. Ueberhuber, Numerical Integration on Advanced Computer Systems, Springer, Berlin, 1994.
[7] D. Laurie, Periodising transformations for numerical integration, J. Comput. Appl. Math. 66 (1996) 337–344.
[8] J. Lyness, L.M. Delves, On the implementation of a modified Sag–Szekeres quadrature method, J. Comput. Appl. Math. 112 (1999) 189–200.
[9] T. Sag, G. Szekeres, Numerical evaluation of high-dimensional integrals, Math. Comput. 18 (1964) 245–253.
[10] A. Sidi, A new variable transformation for numerical integration, Internat. Ser. Numer. Math. 112 (1993) 359–373.