RELIABLE ADAPTIVE, AUTOMATIC QUADRATURE FOR CONES OF INTEGRANDS

FRED J. HICKERNELL, MARTHA RAZO, AND SUNNY YUN

Abstract. Hi

1. Introduction

Although some mathematical problems can be solved using pencil and paper, many can only be solved via numerical algorithms. These algorithms typically generate a sequence of approximations that converge to the true solution. The user wants to know at what point in the sequence the approximation differs from the true solution by no more than a specified tolerance. Stopping criteria used in practice are based either on inconvenient assumptions or on heuristics without guarantees. Here we describe a practical, adaptive trapezoidal algorithm for integration that is guaranteed to provide the correct answer. The key is to consider cones of integrands. We also suggest changing the way numerical quadrature is taught.

1.1. Why Numerical Integration Is Needed?

There are many methods that we learn for evaluating integrals, e.g., substitution, integration by parts, partial fractions, etc. Unfortunately, there are many elementary functions whose integrals *cannot* be expressed as elementary functions. An important example is the standard normal probability density, $\phi : x \mapsto \mathrm{e}^{-x^2/2}/\sqrt{2\pi}$, whose integral is the standard normal distribution, $\Phi : x \mapsto \int_{-\infty}^{x} \phi(t)\,\mathrm{d}t$. Since the Φ is not an elementary function, values of Φ must be provided via tables in statistics textbooks and special functions in calculators. By contrast, the derivatives of elementary functions are always elementary functions.

1.2. The Basic Trapezoidal Rule.

To compute integrals that are not amenable to analytic methods one must turn to numerical integration (quadrature) methods such as the trapezoidal rule, T_n, which is often introduced in calculus courses:

$$\int_0^1 f(x)\,\mathrm{d}x \approx T_n(f) := \frac{1}{2n}\left[f(0) + 2f(1/n) + \cdots + 2f(1-1/n) + f(1)\right]. \tag{1}$$

The trapezoidal rule is the integral of the piecewise linear spline approximation to the integrand. The name of the rule comes from the fact that the integral of the linear spline is a sum of areas of trapezoids (see Figure 1). Under mild smoothness conditions on f one observes that $T_n(f)$ approaches $\int_0^1 f(x)\,\mathrm{d}x$ as n increases.

For the user's convenience one would like to turn T_n into an *automatic* quadrature algorithm — one that automatically chooses n to ensure that the trapezoidal rule error is small enough. Given the user's tolerance for error, ε, an automatic

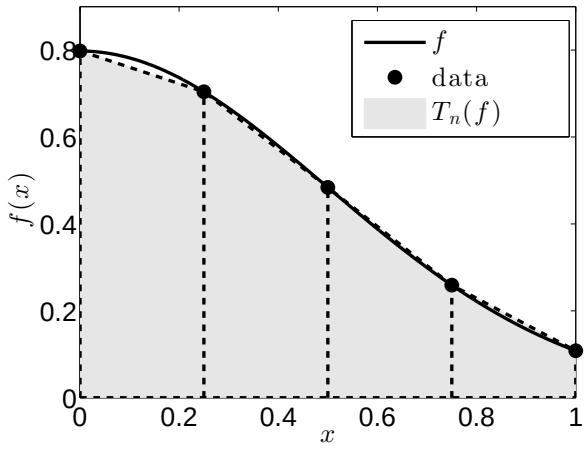

FIGURE 1. The integrand in (5) and the trapezoidal rule approximation for the integral.

trapezoidal algorithm should choose n such that

$$\text{err}(f,n) := \left| \int_0^1 f(x)\,\mathrm{d}x - T_n(f) \right| \le \varepsilon. \tag{2}$$

Although $\text{err}(f,n)$ is rarely known exactly, there exist rigorous upper bounds on the error that are proportional to $1/n^2$ and the roughness of the integrand, e.g., Brass and Petras (2011, Sect. 7.2, (7.15)):

$$\text{err}(f,n) \le \overline{\text{err}}(f,n) := \frac{\text{Var}(f')}{8n^2}, \qquad n \in \mathbb{N} := \{1,2,\ldots\}. \tag{3}$$

Here $\text{Var}(\cdot)$ represents the (total) variation of a function, which is defined as

$$\text{Var}(f) := \sup \left\{ \sum_{i=1}^{n} |f(x_i) - f(x_{i-1})| : n \in \mathbb{N},\ 0 \le x_0 < x_1 < \cdots < x_n \le 1 \right\}. \tag{4}$$

Intuitively, $\text{Var}(f)$ is the total vertical distance up and down that one travels on a roller coaster whose track is the graph of f. The function $f \mapsto \text{Var}(f')$ is a semi-norm. The trapezoidal rule gives the exact answer for linear integrands. Error bound (3) reflects this fact since if $f(x) = ax+b$, then $f'(x) = a$, and so $\text{Var}(f') = 0$.

To illustrate this error bound, consider the normal probability example mentioned above. The trapezoidal rule approximation using $n = 4$ trapezoids, the actual error, and the error bound are as follows:

$$f(x) = 2\phi(2x) = \sqrt{\frac{2}{\pi}} \mathrm{e}^{-2x^2}, \qquad \int_0^1 f(x)\,\mathrm{d}x = 0.4772, \qquad T_4(f) = 0.4750, \tag{5a}$$

$$\text{Var}(f') = 1.5038, \qquad \text{err}(f,4) = 0.0022 \le 0.0117 = \overline{\text{err}}(f,n). \tag{5b}$$

(The value of the integral to four significant digits may be found by a variety of different quadrature algorithms.) Figure 1 depicts the integrand and the trapezoidal rule approximation. Note that the error is bounded above by the error bound in (3).

Error bound (3) can help us determine how large n must be to meet the error tolerance if an upper bound on $\text{Var}(f')$ is available. Section 2 provides the details of

an automatic algorithm for such cases. In practice nothing may be known a priori about $\mathrm{Var}(f')$. Adaptive algorithms do not require an upper bound on $\mathrm{Var}(f')$ but estimate the error based on the function values sampled and then apply a stopping criterion to determine how many function values are required. Section 3 describes the stopping criterion commonly taught in elementary numerical analysis courses, but which has significant flaws, as pointed out by James Lyness (1983). An improved stopping criterion for integrands lying in cones is described in Section 4. A general framework for guaranteed automatic algorithms is presented in Clancy et al. (2013), which also describes the new adaptive trapezoidal algorithm presented here. In this article we describe the new algorithm and its ramifications in greater detail and argue why we should change how we teach error estimation for numerical integration.

2. An Automatic Algorithm for Balls of Integrands

If f has a simple form, then one may be able to derive an upper bound on $\mathrm{Var}(f')$. In this case the following automatic algorithm based on error bound (3) may be used.

Algorithm 1 (Automatic, for Balls)**.** Given a $\sigma > 0$ and an error tolerance, ε, set

$$n = \left\lceil \sqrt{\sigma/(8\varepsilon)} \right\rceil, \tag{6}$$

and return the trapezoidal rule approximation $T_n(f)$ as the answer.

Theorem 1. *If* $\mathrm{Var}(f') \le \sigma$*, then Algorithm 1 is successful, i.e.,*

$$\left| \int_0^1 f(x)\,\mathrm{d}x - T_n(f) \right| \le \varepsilon.$$

Algorithm 1 is guaranteed to work for all integrands lying in the ball $\mathcal{B}_\sigma = \{f : \mathrm{Var}(f') \le \sigma\}$. This algorithm is automatic because it expends as much effort as required by the error tolerance, ε and the radius of the ball, σ, to obtain the desired answer. It is not an adaptive algorithm because the computational cost does not depend on the particulars of f.

For f in (5), one may choose $\sigma = \mathrm{Var}(f') = 1.5038$, and then Algorithm 1 uses $n = \lceil 0.4336/\sqrt{\varepsilon}\,\rceil$ trapezoids. However, not all integrands have first derivatives whose variations are so easy to bound.

3. An Adaptive Trapezoidal Algorithm with a Flawed Stopping Criterion

Automatic quadrature algorithms do not require the user to specify σ. Instead, they bound or estimate the error using only function values and then determine the sample size accordingly. MATLAB (2013), Mathematica (2012), NAG (2012), and R (2012) all contain automatic quadrature algorithms. Unfortunately, their stopping criteria are unreliable, as pointed out by James Lyness (1983). To illustrate the flawed logic that leads to unreliable error estimation in these sophisticated algorithms, we focus on the simpler case of an adaptive trapezoidal algorithm.

3.1. **A Popular Error Estimate.** Texts such as Burden and Faires (2010, p. 223–224), Cheney and Kincaid (2013, p. 233), and Sauer (2012, p. 270), advise readers to estimate the error of $T_n(f)$ by comparing it to $T_{n/2}(f)$, specifically,

$$\widehat{\mathrm{err}}(f,n) = \frac{\bigl|T_n(f) - T_{n/2}(f)\bigr|}{3}, \qquad \frac{n}{2} \in \mathbb{N}. \tag{7}$$

This error estimate leads to the following adaptive, automatic quadrature algorithm. For each iteration the number of trapezoids is doubled so that the data from the previous iteration can be reused.

Algorithm 2. Given an error tolerance, ε, let $j = 1$ and $n_1 = 2$.

Step 1: Compute the error estimate $\widehat{\mathrm{err}}(f, n_j)$ according to (7).

Step 2: If $\widehat{\mathrm{err}}(f, n_j) \le \varepsilon$, then return the trapezoidal rule approximation $T_{n_j}(f)$ as the answer.

Step 3: Otherwise let $n_{j+1} = 2n_j$, increase j by one, and go to Step 1.

Error estimate 7 may be justified by noting that Simpson's rule may be written as

$$S_n(f) := \frac{4T_n(f) - T_{n/2}(f)}{3} = T_n(f) + \frac{T_n(f) - T_{n/2}(f)}{3} =$$
$$= \frac{1}{3n}\left[f(0) + 4f(1/n) + 2f(2/n) + 4f(3/n) + \cdots + 4f(1-1/n) + f(1)\right].$$

The error bound of Simpson's rule then serves as an bound on the error of the error estimate (Brass and Petras, 2011, Sect. 7.3, p. 231):

$$\begin{aligned} |\mathrm{err}(f,n) - \widehat{\mathrm{err}}(f,n)| &\le \left| \int_0^1 f(x)\,\mathrm{d}x - T_n(f) - \frac{T_n(f) - T_{n/2}(f)}{3} \right| \\ &= \left| \int_0^1 f(x)\,\mathrm{d}x - S_n(f) \right| \le \frac{\mathrm{Var}(f''')}{36n^4}, \qquad \frac{n}{2} \in \mathbb{N}. \end{aligned} \tag{8}$$

Comparing the orders of convergence in (8) and (3), it follows that the trapezoidal rule error estimate does an excellent job *for n large enough*, assuming finite $\mathrm{Var}(f''')$.

The success of Algorithm 2 requires the error estimate, $\widehat{\mathrm{err}}(f,n)$, to be reliable, and this requires an upper bound on $\mathrm{Var}(f''')$. However, if an upper bound on $\mathrm{Var}(f''')$ is known, then it makes more sense to apply Simpson's rule directly to obtain a higher order convergence rate than that given by Algorithm 2. Therefore, we do not formulate any theorem guaranteeing Algorithm 2.

Without any constraint on $\mathrm{Var}(f''')$ one may easily construct integrands, f, that fool the error estimate completely, e.g.,

$$\int_0^1 f(x)\,\mathrm{d}x = 1, \quad T_n(f) = T_{n/2}(f) = -1, \quad \mathrm{err}(f,n) = 2 \ne 0 = \widehat{\mathrm{err}}(f,n). \tag{9}$$

Figure 2 shows two integrands satisfying these conditions. These two integrands illustrate two different ways in which the error estimate $\widehat{\mathrm{err}}(\cdot)$, and Algorithm 2, can fail. One way is ultimately unavoidable but ought to be quantifiable. The other way is inexcusable, but is prevalent in widely used automatic numerical quadrature.

3.2. **Spiky Integrands.** Figure 2 (left) depicts the following spiky integrand:

$$f_{\mathrm{spiky}}(x;n) = -1 + 60[\{nx\}(1 - \{nx\})]^2, \qquad n \in \mathbb{N}, \quad \{x\} := x \bmod 1. \tag{10}$$

The integrand $f_{\mathrm{spiky}}(\cdot;n)$ satisfies (9) for even n and has

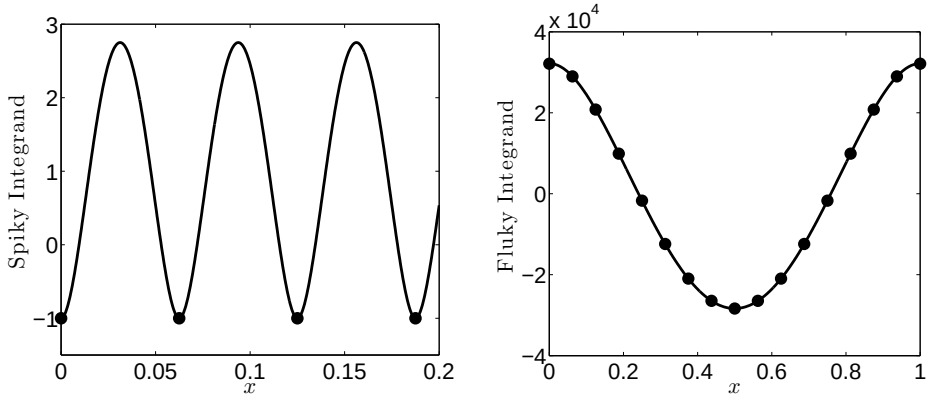

FIGURE 2. Examples of a spiky integrand (left) and a fluky integrand (right), which satisfy the failure conditions (9) for $n = 16$.

- $n + 1$ local minima of -1 at $x = 0, 1/n, \ldots, 1$, and
- n local maxima of $11/4$ at $x = 1/(2n), 3/(2n), \ldots, 1 - 1/(2n)$.

The variation of the first and third derivatives of $f(\cdot; n)$ increase with n, specifically,

$$\mathrm{Var}(f'_{\mathrm{spiky}}(\cdot; n)) = \frac{80n^2}{\sqrt{3}}, \qquad \mathrm{Var}(f'''_{\mathrm{spiky}}(\cdot; n)) = 1440n^4.$$

Thus, it is not surprising that the error of the trapezoidal rule and the error of the error estimate both stay constant even as n increases.

The problem of handling spiky integrands is not unique to Algorithm 2. *Any* quadrature algorithm that depends only on function values to estimate error will be fooled by sufficiently spiky integrands. Suppose that the quadrature algorithm samples the integrand at a seqeunce of points, $\xi_1, \xi_2, \ldots$, and suppose that $f(\xi_i)$ is the same value for all ξ_i. Here the algorithm may even be adaptive, meaning that ξ_{i+1} may depend on $(\xi_1, f(\xi_1)), \ldots, (\xi_i, f(\xi_i))$. Eventually the algorithm stops at n points. Under mild conditions on the set of integrands of interest the best estimate of the integral is that constant. However, one can construct non-constant functions that have any value of the integral.

Algorithm 1 for balls of integrands can be fooled by spiky integrands, however, Theorem 1 is valid because those spiky integrands lie outside the ball. Increasing the radius of the ball allows one to accommodate spikier integrands. Algorithm **??** in Section 4 below can also be fooled by spiky integrands, however, its guarantee in Theorem **??** is valid because those spiky integrands lie outside the cone. Increasing the cone parameter allows one to accommodate spikier integrands. Moreover, the cone parameter itself corresponds to an inverse length scale corresponding to the amount of spikiness that one wishes to tolerate.

3.3. **Fluky Integrands.** Thirty years ago James Lyness wrote a SIAM Review article, *When Not to Use an Automatic Quadrature Routine*. While recognizing the problem of spiky integrands, he claims that there is even a more serious problem with automatic quadrature methods (Lyness, 1983, p. 69):

> While prepared to take the risk of being misled by chance alignment of zeros in the integrand function, or by narrow peaks which are "missed," the user may wish to be reassured that for "reasonable" integrand functions which do not have these characteristics all will be well. It is the purpose of the rest of this section to demonstrate by example that he cannot be reassured on this point. In fact the routine is likely to be unreliable in a significant proportion of the problems it faces (say 1 to 5%) and there is no way of predicting in a straightforward way in which of any set of apparently reasonable problems this will happen.

Lyness went on to describe how to construct, what we would call a *fluky* integrand.

Figure 2 (right) plots a fluky integrand that is defined in terms of the quadratic and quartic Bernoulli polynomials:

$$f_{\text{fluky}}(x;n) = \frac{2-5n^2+n^4}{2} + 15n^2x(1-x)\{1-n^2x(1-x)\}, \qquad n \in \mathbb{N}. \tag{11}$$

The integrand $f_{\text{fluky}}(\cdot;n)$ satisfies (9) for even n and has only

- 1 local minimum of $(16+20n^2-7n^4)/16$ at $x=1/2$, and
- 2 local maxima of $(19-10n^2+2n^4)/4$ at $x=(1\pm\sqrt{1-2/n^2})/2$.

The variation of the first and third derivatives of $f(\cdot;n)$ increase with n, specifically,

$$\operatorname{Var}(f'_{\text{fluky}}(\cdot;n)) = \frac{10n}{3}\Bigl[9n+2\sqrt{3(-2+n^2)^3}\Bigr], \qquad \operatorname{Var}(f'''_{\text{fluky}}(\cdot;n)) = 360n^4.$$

We describe integrands like f_{fluky} as fluky because their construction is rather delicate. They satisfy conditions (9) with a small number of local optima.

The failure of the error estimate in (7) for f_{fluky} seems avoidable. Although the values of the trapezoidal rule are moderate, for moderate n one already observes a wide range of the sampled function values, $\bigl(f_{\text{fluky}}(i/n;n)\bigr)_{i=1}^n$, which implies that $\operatorname{Var}(f_{\text{fluky}})$ must be large. This is not the case for the sampled function values of f_{spiky}. In the next section we transform this intuition into a better method for bounding the error of the trapezoidal rule.

4. A Guaranteed, Adaptive, Automatic Trapezoidal Algorithm

We know from (3) that the error of the trapezoidal rule is bounded in terms of $\operatorname{Var}(f')$. Estimating $\operatorname{Var}(f')$ reliably requires additional smoothness assumptions on f, but if we assume more smoothness, then we ought to use a different rule, like Simpson's rule. On the other hand, finite $\operatorname{Var}(f')$ does allow us to reliably estimate, a weaker semi-norm of f, such as $\operatorname{Var}(f)$. As observed in Section 3.3 above, the fluky algorithms that fool Algorithm 2 have large $\operatorname{Var}(f)$. The approach taken in this section is to explore what can be accomplished by estimating a weaker semi-norm.

Let A_n denote the linear spline approximation using $n+1$ equally spaced points, i.e.,

$$A_n(f)(x) = \begin{cases} f(0)(1-nx) + f(1/n)nx, & 0 \le x < 1/n, \\ f(1/n)(2-nx) + f(2/n)(nx-1), & 1/n \le x < 2/n, \\ \qquad\vdots & \qquad\vdots \\ f(1-1/n)(n-nx) + f(1)(nx-n+1), & 1-1/n \le x \le 1. \end{cases} \tag{12}$$

The trapezoidal rule can be written as $T_n(f) = \int_0^1 A_n(f)(x)\,\mathrm{d}x$. Using this definition of linear spline we focus on the semi-norm defined by $f \mapsto \mathrm{Var}(f - A_1(f))$. Like $f \mapsto \mathrm{Var}(f')$ this weaker semi-norm also vanishes for functions that are linear in x. If $\mathrm{Var}(f')$ is finite, then it follows that $\mathrm{Var}(f) = \|f'\|_1 := \int_0^1 |f(x)|\,\mathrm{d}x$, so the weaker semi-norm may be written as $f \mapsto \|f' - f(1) + f(0)\|_1$. It is shown in Clancy et al. (2013, Section 5.1) that

$$2\,\|f' - f(1) + f(0)\|_1 \le \mathrm{Var}(f') \qquad \forall f \text{ with } \mathrm{Var}(f') < \infty. \tag{13}$$

The data-driven quantity

$$\|A_n(f)' - f(1) + f(0)\|_1 = \sum_{i=1}^{n} \left| f\left(\frac{i}{n}\right) - f\left(\frac{i-1}{n}\right) - \frac{f(1) - f(0)}{n} \right| \tag{14}$$

approximates $\|f' - f(1) + f(0)\|_1$ well provided that $\mathrm{Var}(f')$ is not too large. Again in Clancy et al. (2013, Section 5.1) it is shown that

$$0 \le \|f' - f(1) + f(0)\|_1 - \|A_n(f)' - f(1) + f(0)\|_1 \le \frac{\mathrm{Var}(f')}{2n}. \tag{15}$$

At this point it seems that we have reached a dead end because we have an error bound in terms of $\mathrm{Var}(f')$, just like the trapezoidal rule error bound (3). However, we make a further assumption that f lies in the cone defined as

$$\mathcal{C}_\tau := \{f : \mathrm{Var}(f') \le \tau\,\|f' - f(1) + f(0)\|_1\}. \tag{16}$$

This cone consists of functions whose stronger semi-norms are not too much greater than their weaker semi-norms. The set $\mathcal{C}_\tau$ is called a cone because $f \in \mathcal{C}_\tau$ implies that $cf \in \mathcal{C}_\tau$ for all $c \in \mathbb{R}$. Substituting this upper bound on $\mathrm{Var}(f')$ into (15) and re-arranging the terms implies a data-driven upper bound on $\mathrm{Var}(f')$:

$$\mathrm{Var}(f') \le \tau\,\|f' - f(1) + f(0)\|_1 \le \frac{\tau\,\|A_n(f)' - f(1) + f(0)\|_1}{1 - \tau/(2n)} \qquad \forall f \in \mathcal{C}_\tau, \tag{17}$$

provided that $n > \tau/2$. There is also a companion bound in the other direction based on (13) and (15):

$$\|A_n(f)' - f(1) + f(0)\|_1 \le \|f' - f(1) + f(0)\|_1 \le \frac{\mathrm{Var}(f')}{2}. \tag{18}$$

Plugging (17) into (3) yields the following guaranteed error bound for the trapezoidal rule:

$$\mathrm{err}(f,n) \le \widetilde{\mathrm{err}}(f,n) := \frac{\tau\,\|A_n(f)' - f(1) + f(0)\|_1}{4n(2n - \tau)} \qquad \forall f \in \mathcal{C}_\tau. \tag{19}$$

This new error bound can then be used to construct the following adaptive, automatic quadrature algorithm, which is a simplification of Clancy et al. (2013, Algorithm 4):

Algorithm 3 (Adaptive, Automatic, for Cones)**.** Given $\tau > 0$, and an error tolerance, ε, let $j = 1$ and $n_1 = \lceil (\tau + 1)/2 \rceil$.

Step 1: Compute the error estimate $\widetilde{\mathrm{err}}(f, n_j)$ according to (19).

Step 2: If $\widetilde{\mathrm{err}}(f, n_j) \le \varepsilon$, then return the trapezoidal rule approximation $T_{n_j}(f)$ as the answer.

Step 3: Otherwise let $n_{j+1} = 2n_j$, increase j by one, and go to Step 1.

Error bound (19) is used to prove that Algorithm 3 is successful for integrands in $\mathcal{C}_\tau$. Let $N(f,\varepsilon,\tau)$ denote the eventual number of trapezoids required by this algorithm. Because the algorithm is adaptive, the number of trapezoids does depend on the particulars of f. The two bounds (17) and (18) may be used to derive both lower and upper bounds on $N(f,\varepsilon,\tau)$:

Theorem 2. (Clancy et al., 2013, Theorem 7) *Let $N(f,\varepsilon,\tau)$ denote the final n_j in Algorithm 3. Then this algorithm is successful for all functions in $\mathcal{C}_\tau$, i.e., $\left|\int_0^1 f(x)\,\mathrm{d}x - T_{N(f,\varepsilon,\tau)}(f)\right| \le \varepsilon$. Moreover, $N(f,\varepsilon,\tau)$ is bounded below and above as follows:*

$$\begin{aligned}(20)\quad &\max\left(\left\lceil\frac{\tau+1}{2}\right\rceil, \left\lceil\sqrt{\frac{\mathrm{Var}(f')}{8\varepsilon}}\right\rceil\right)\\ &\le \max\left(\left\lceil\frac{\tau+1}{2}\right\rceil, \left\lceil\sqrt{\frac{\tau\left\|f'-f(1)+f(0)\right\|_1}{8\varepsilon}}\right\rceil\right)\\ &\le N(f,\varepsilon,\tau)\\ &\le \sqrt{\frac{\tau\left\|f'-f(1)+f(0)\right\|_1}{2\varepsilon}}+\tau+3 \le \sqrt{\frac{\tau\,\mathrm{Var}(f')}{4\varepsilon}}+\tau+3.\end{aligned}$$

Note that the cost of the algorithm depends on $\mathrm{Var}(f')$, but without $\mathrm{Var}(f')$ needing to be known in advance. For an integrand like (5) where $\mathrm{Var}(f')$ is known, the number of trapezoids required by Algorithm 3 is no greater than $\sqrt{\tau\,\mathrm{Var}(f')/(4\varepsilon)}+\tau+3$, whereas for Algorithm 1 with $\sigma=\mathrm{Var}(f')$ the number of trapezoids is no more than $\left\lceil\sqrt{\mathrm{Var}(f')/(8\varepsilon)}\,\right\rceil$. The extra cost for Algorithm 3 comes in the form of the factor $\sqrt{2\tau}$ and the additional additive term of $\tau+1$. On the other hand, if $\mathrm{Var}(f')$ is not known in advance, then the disadvantage of Algorithm 1 is that if $\mathrm{Var}(f')$ is not known in advance, and σ is chosen conservatively, then Algorithm 3 could cost arbitrarily more than $\sqrt{\mathrm{Var}(f')/(8\varepsilon)}+1$ in the limit of ,

5. Acknowledgements

The authors are grateful for discussions with a number of colleagues. This research is supported in part by grant NSF-DMS-1115392.

References

Abramowitz, M. and I. A. Stegun (eds.) 1964. *Handbook of mathematical functions with formulas, graphs and mathematical tables*, U. S. Government Printing Office, Washington, DC.

Brass, H. and K. Petras. 2011. *Quadrature theory: the theory of numerical integration on a compact interval*, First, American Mathematical Society, Rhode Island.

Burden, R. L. and J. D. Faires. 2010. *Numerical analysis*, Ninth, Cengage Brooks/Cole, Belmont, CA.

Cheney, W. and D. Kincaid. 2013. *Numerical mathematics and computing*, Seventh, Brooks/Cole, Boston.

Clancy, N., Y. Ding, C. Hamilton, F. J. Hickernell, and Y. Zhang. 2013. *The complexity of guaranteed automatic algorithms: Cones, not balls.* submitted for publication, arXiv.org:1303.2412 [math.NA].

Lyness, J. N. 1983. *When not to use an automatic quadrature routine*, SIAM Rev. **25**, 63–87.

R Development Core Team. 2012. *The R Project for Statistical Computing.*

Sauer, T. 2012. *Numerical analysis*, Pearson.

The MathWorks, Inc. 2013. *MATLAB 8.1*, Natick, MA.
The Numerical Algorithms Group. 2012. *The NAG library*, Mark 23, Oxford.
Wolfram Research Inc. 2012. *Mathematica 9*, Champaign, IL.