RELIABLE AUTOMATIC QUADRATURE FOR CONES OF INTEGRANDS

FRED J. HICKERNELL, MARTHA RAZO, AND SUNNY YUN

ABSTRACT. Hi

1. INTRODUCTION

Automatic numerical algorithms for univariate integration appear in many widely used software packages, such as MATLAB (2013), Mathematica (2012), NAG (2012), and R (2012). Automatic algorithms determine adaptively where and how much to sample the integrand by estimating the algorithm error from the sampled function values. Unfortunately, such error estimation methods are unreliable, as pointed out by James Lyness (1983).

Here we show how to overcome Lyness's objections by employing a new paradigm for error bounds, which is based on cones, rather than balls, of integrands rather than balls. This article illustrates with greater detail the more general setting in Clancy et al. (2013). We argue that the traditional way of teaching error estimation for numerical integration should be changed to reflect this new paradigm. Moreover, this paradigm can, and should, be applied to other problems requiring automatic algorithms.

1.1. **Why Numerical Integration Is Needed?** Soon after a calculus student first learns about integration, she learns that the integrals of many elementary functions are *not* themselves elementary functions. An important example is the standard normal probability density function, $\phi : x \mapsto e^{-x^2}/\sqrt{2\pi}$. Since the antiderivative of ϕ cannot be expressed in terms of elementary functions, values of the standard normal distribution function $\Phi : x \mapsto \int_{-\infty}^{x} \phi(t)\, dt$ must be provided in tables in textbooks and special functions in calculators. By contrast, the derivatives of elementary functions are elementary functions.

1.2. **The Trapezoidal Rule.** Students need to know that no analytic answer does not mean no answer, so calculus students are often introduced to numerical integration (quadrature) methods such as the trapezoidal rule, T_n:

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

One may ask "How large should the number of trapezoids, n, be so that the absolute error of $T_n(f)$ does not exceed a tolerance ε?" Theoretical and rigorous error bounds are proportional to $1/n^2$, e.g., Brass and Petras (2011, Sect. 7.2, (7.15)):

$$(2) \qquad \mathrm{err}(f, n) := \left|\int_0^1 f(x)\, dx - T_n(f)\right| \leq \frac{\mathrm{Var}(f')}{8n^2}, \qquad n \in \mathbb{N} = \{1, 2, \ldots\}.$$

The error bound is also proportional to the total variation of the first derivative of the integrand. The total variation, $\mathrm{Var}(\cdot)$, is defined as

$$(3) \qquad \mathrm{Var}(f) := \sup_{\substack{n \in \mathbb{N} \\ 0 = x_0 < x_1 < \cdots < x_n = 1}} \sum_{i=1}^{n} |f(x_i) - f(x_{i-1})| .$$

The trapezoidal rule approximates the integrand by a piecewise linear spline and thus gives the exact answer for such integrands. Error bound (2) reflects this since $\mathrm{Var}(f')$ vanishes when $f(x) = ax + b$.

The total variation is closely related to the $\mathcal{L}_1$-norm:

$$\mathrm{Var}(f) = \|f'\|_1 \text{ if } \|f'\|_1 < \infty,$$

$$\text{where} \qquad \|f\|_p := \begin{cases} \left[\int_0^1 |f(x)|^p \, \mathrm{d}x \right]^{1/p}, & 1 \le p < \infty, \\ \sup_{0 \le x \le 1} |f'(x)|, & p = \infty. \end{cases}$$

Error bounds resembling (2) exist in terms of $\|f''\|_p$ for $1 \le p \le \infty$ with leading constants depending on p.

If f has a simple form, one might be able to derive an upper bound on $\mathrm{Var}(f')$ and turn error bound (2) into an automatic algorithm. The algorithm below is guaranteed to produce $\mathrm{err}(f, n) \le \varepsilon$ provided that $\mathrm{Var}(f') \le M_1$.

Algorithm 1. Given an error tolerance, ε, set

$$(4) \qquad n = \left\lceil \sqrt{M_1/(8\varepsilon)} \right\rceil ,$$

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

For example, for ϕ defined above, $\mathrm{Var}(f') = 1/\sqrt{2\pi e}$, and so one may choose any $n \ge 1/\sqrt{8\sqrt{2\pi e}\,\varepsilon}$. However, not all integrands have first derivatives whose variations are so easy to bound.

2. The Flaws in the Prevalent Data-Based Error Estimate

Automatic quadrature algorithms do not require the user to specify M_1, but bound or estimate the error based solely on the function values, $f(x_i)$, and then determine the sample size accordingly. Here we focus on this error estimation process. As we demonstrate in this section, the popular error estimate for the trapezoidal rule is flawed.

Numerical methods 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 $T_n(f)$ in terms of its from $T_{n/2}(f)$, specifically,

$$(5) \qquad \widehat{\mathrm{err}}(f, n) = \frac{|T_n(f) - T_{n/2}(f)|}{3}, \qquad \frac{n}{2} \in \mathbb{N}.$$

This algorithm leads to the following automatic and adaptive quadrature algorithm.

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

> **Stage 1:** Compute the error estimate $\widehat{\mathrm{err}}(f, n_j)$ according to (5).
> **Stage 2:** If $\widehat{\mathrm{err}}(f, n_j) \le \varepsilon$, then return the trapezoidal rule approximation $T_{n_j}(f)$ as the answer.
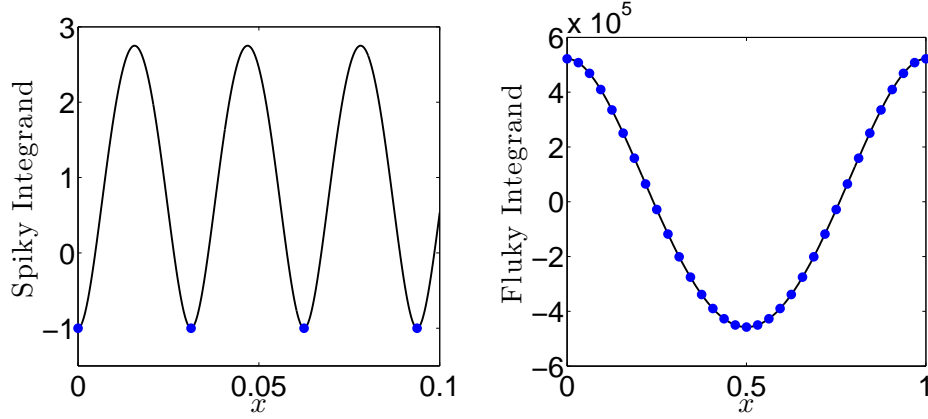> **Stage 3:** Otherwise let $n_{j+1} = 2n_j$, increase j by one, and go to Stage 1.

FIGURE 1. Examples of a spiky integrand (left) and a fluky integrand (right), which satisfy the failure conditions (6) for $n = 32$.

By doubling the number of trapezoids for each iteration, the data from the previous iterations can be reused. Automatic quadrature algorithms in commonly used numerical libraries are more sophisticated than Algorithm 2, but their stopping criteria are similar in spirit and have the same intrinsic flaws.

While, Algorithm 2 requires no a priori bound on the variation of the first derivative of the integrand, there is no guarantee that $\mathrm{err}(f, n) \leq \varepsilon$. In fact, one may easily to construct integrands, f, such that this error estimate fails completely, e.g.,

$$(6) \qquad \int_0^1 f(x)\, \mathrm{d}x = 1, \quad T_n(f) = T_{n/2}(f) = -1, \quad \mathrm{err}(f, n) = 2 \neq 0 = \widehat{\mathrm{err}}(f, n).$$

Figure 1 shows examples of two integrands satisfying these conditions. These two integrands illustrate two 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.

2.1. **Spiky Integrands.** *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.

Figure 1 (left) plots the following spiky integrand that is defined in terms of the quartic Bernoulli polynomial:

$$(7) \qquad f_{\mathrm{spiky}}(x; n) = 1 + 60B_4(\{nx\}), \qquad n \in \mathbb{N}, \quad \{x\} := x \bmod 1.$$

The Bernoulli polynomials are described by Abramowitz and Stegun (1964, Chap. 23, http://dlmf.nist.gov/24). The first six are defined as follows

$$\text{(8a)} \qquad B_0(x) = 1, \qquad B_1(x) = x - 1/2, \qquad B_2(x) = 1/6 - x(1-x),$$

$$\text{(8b)} \qquad B_3(x) = \frac{1}{2}x(1-x)(1-2x), \qquad B_4(x) = -1/30 + [x(1-x)]^2,$$

$$\text{(8c)} \qquad B_5(x) = \frac{1}{6}x(1-x)(1-2x)(1+3x-3x^2).$$

The integrand $f_{\text{spiky}}(\cdot; n)$ satisfies (6) for even n and has

- $n+1$ 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 problem of integrating spiky functions never goes away, but it should be quantifiable. An algorithm ought to integrate functions accurately, provided that they have spikes no thinner than say, δ. Moreover, the algorithm should be tunable so that the user can specifiy δ. We describe such an algorithm in Section 3, along with a precise definition of spike width.

2.2. **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 1 (right) plots a fluky integrand that is defined in terms of the quadratic and quartic Bernoulli polynomials:

$$\text{(9)} \qquad f_{\text{fluky}}(x; n) = 1 - 15n^2[B_2(x) + n^2 B_4(x)], \qquad n \in \mathbb{N}.$$

The integrand $f_{\text{fluky}}(\cdot; n)$ satisfies (6) 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 $x = (1 \pm \sqrt{1 - 2/n^2})/2$.

Integrands like this one are termed fluky by us because their construction is rather delicate. They satisfy conditions (6) without a large number of local optima. To illustrate this delicacy, note that removing either the B_2 or the B_4 term does not affect the value of the integral but does cause $\widehat{\text{err}}(f_{\text{fluky}}(\cdot; n), n)$ to be significantly different from zero.

The failure of the error estimate in (5) for $f_{\text{fluky}}(\cdot; n)$ seems avoidable. For even moderate n, the sampled function values, $\left(f_{\text{fluky}}(i/n; n)\right)_{i=1}^{n}$, are spread across a large range. This should indicate that true error, $\text{err}(f_{\text{fluky}}(\cdot; n), n)$, might be substantial, and far from the error estimate, $\widehat{\text{err}}(f_{\text{fluky}}(\cdot; n), n) = 0$. In Section

3 we transform this intuition into a better method for bounding the error of the trapezoidal rule.

2.3. Necessary Conditions for Failure of the Error Estimate.
Integrands that satisfy (6) and fool the error estimate in (5) must satisfy certain conditions. First note that the error bound in (2) implies that

$$\mathrm{Var}(f') \geq 8n^2 \, \mathrm{err}(f,n) = 16n^2 \asymp n^2 \qquad \text{as } n \to \infty,$$

so the total variation of the first derivative must be increasingly large as the number of trapezoids increases. Moreover, the trapezoidal rule added to the error estimate in (5) is actually Simpson's rule:

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

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):

$$(10) \qquad |\mathrm{err}(f,n) - \widehat{\mathrm{err}}(f,n)| = \left| \int_0^1 f(x)\, \mathrm{d}x - S_n(f) \right| \leq \frac{\mathrm{Var}(f''')}{36n^4}, \qquad \frac{n}{2} \in \mathbb{N}.$$

So, integrands satisfying (6) must also satisfy

$$\mathrm{Var}(f''') \geq 36n^4 \, |\mathrm{err}(f,n) - \widehat{\mathrm{err}}(f,n)| = 36n^4 \asymp n^4 \qquad \text{as } n \to \infty.$$

Thus, the total variation of the third derivative must increase even faster than the minimum increase of $\mathrm{Var}(f')$ as the number of trapezoids increases.

The derivatives of Bernoulli polynomials are multiples of lower degree Bernoulli polynomials and the integrals of nonconstant Bernoulli polynomials vanish (see Abramowitz and Stegun (1964, Chap. 23, `http://dlmf.nist.gov/24`)):

$$B_n'(x) = nB_{n-1}(x), \quad \int_0^1 B_{n+1}(x)\, \mathrm{d}x, \qquad n \in \mathbb{N}.$$

The formula for the derivatives facilitates the computation of the total variation of the spiky and fluky examples above, confirming that the variations of their first and third derivatives satisfy the above inequalities. For the spiky integrand defined in (7),

$$(11a) \qquad \mathrm{Var}(f_{\mathrm{spiky}}(\cdot; n)) = \frac{15n}{2}, \qquad \mathrm{Var}(f'_{\mathrm{spiky}}(\cdot; n)) = \frac{80n^2}{\sqrt{3}} > 16n^2,$$

$$(11b) \qquad \mathrm{Var}(f''_{\mathrm{spiky}}(\cdot; n)) = 360n^3, \qquad \mathrm{Var}(f'''_{\mathrm{spiky}}(\cdot; n)) = 1440n^4 > 36n^4.$$

For the fluky integrand defined in (9),

$$(12a) \qquad \mathrm{Var}(f_{\mathrm{fluky}}(\cdot; n)) = \frac{15}{8}(8 - 4n^2 + n^4),$$

$$(12b) \qquad \mathrm{Var}(f'_{\mathrm{fluky}}(\cdot; n)) = \frac{10n}{3}\left[9n + 2\sqrt{3(-2 + n^2)^3} \right] > 16n^2,$$

$$(12c) \qquad \mathrm{Var}(f''_{\mathrm{fluky}}(\cdot; n)) = 90n^4, \qquad \mathrm{Var}(f'''_{\mathrm{fluky}}(\cdot; n)) = 360n^4 > 36n^4.$$

For the spiky integrand the asymptotic order of the total variation for large n increases with the order of the derivative while for the fluky integrand it stays the same. Specifically,

$$\mathrm{Var}(f_{\mathrm{spiky}}^{(j)}(\cdot;n)) \asymp n^{j+1}, \quad \mathrm{Var}(f_{\mathrm{fluky}}^{(j)}(\cdot;n)) \asymp n^4, \qquad j = 0, \ldots, 3.$$

This difference in asymptotic behaviors distinguishes whether a better error bound developed in the next section will be successful: no for the spiky integrand , but yes for the fluky integrand.

3. A GUARANTEED QUADRATURE ALGORITHM

There are two paths that one might take in constructing a rigorous *data-based* error bound for the trapezoidal rule. Since the error in the error estimate, $\mathrm{err}(f,n) - \widehat{\mathrm{err}}(f,n)$ is proportional to $\mathrm{Var}(f''')$, one might try to construct or assume a bound on $\mathrm{Var}(f''')$, but this is even more difficult than bounding the term $\mathrm{Var}(f')$ that occurs in the original error bound, $\mathrm{err}(f,n)$. Estimating $\mathrm{Var}(f''')$ from data requires knowledge about even higher order derivatives of f. The perhaps less obvious, but successful, approach is to estimate a weaker semi-norm than $f \mapsto \mathrm{Var}(f')$.

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

$$(13) \quad A_n(f)(x) = \begin{cases} f(0)(1/n - x) + f(1/n)x, & 0 \le x < 1/n, \\ f(1/n)(2/n - x) + f(2/n)(x - 1/n), & 1/n \le x < 2/n, \\ \quad\vdots & \quad\vdots \\ f(1 - 1/n)(1 - x) + f(1)(x - 1 + 1/n), & 1 - 1/n \le x \le 1. \end{cases}$$

The trapezoidal rule can be written as $T_n(f) = \int_0^1 A_n(f)(x)\,\mathrm{d}x$. Using this definition 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.

One may approximate $\mathrm{Var}(f)$ well by $\mathrm{Var}(A_n(f))$ provided that $\mathrm{Var}(f')$ is not too large. For finite $\mathrm{Var}(f')$ it follows that $\mathrm{Var}(f) = \|f'\|_1$. This facilitates an bounds on $\mathrm{Var}(f) - \mathrm{Var}(A_n(f))$:

$$\mathrm{Var}(f) - \mathrm{Var}(A_n(f)) = \int_0^1 |f'(x)|\,\mathrm{d}x - \sum_{i=1}^n |f(i/n) - f((i-1)/n)|$$

$$= \sum_{i=1}^n \int_{(i-1)/n}^{i/n} [|f'(x)| - n\,|f(i/n) - f((i-1)/n)|]\,\mathrm{d}x$$

$$\le \sum_{i=1}^n \int_{(i-1)/n}^{i/n} |f'(x) - n[f(i/n) - f((i-1)/n)]|\,\mathrm{d}x.$$

The second right hand side above must be no less than zero, so $\mathrm{Var}(f) - \mathrm{Var}(A_n(f)) \ge 0$. Using integration by parts it can be verified that

$$f'(x) - n[f(i/n) - f((i-1)/n)] = (n-1)\int_{(i-1)/n}^{i/n} w_i(t,x) f''(t)\,\mathrm{d}t,$$

where

$$w_i(t,x) = \begin{cases} t - (i-1)/n, & (i-1)/n \le t \le x \le i/n, \\ t - i/n, & (i-1)/n \le x < t \le i/n. \end{cases}$$

Switching the order of integration then leads to the upper bound

$$\text{Var}(f) - \text{Var}(A_n(f))$$

$$\leq (n-1) \sum_{i=1}^{n} \int_{(i-1)/n}^{i/n} \int_{(i-1)/n}^{i/n} |w(t,x)||f''(t)| \mathrm{d}t \, \mathrm{d}x$$

$$\leq (n-1) \sum_{i=1}^{n} \int_{(i-1)/n}^{i/n} 2[t-(i-1)/n](i/n-t)|f''(t)| \, \mathrm{d}t$$

$$\leq (n-1) \sum_{i=1}^{n} \max_{(i-1)/n \leq t \leq i/n} |2(t-x_i)(i/n-t)| \int_{(i-1)/n}^{i/n} |f''(t)| \, \mathrm{d}t$$

$$\leq \sum_{i=1}^{n} \frac{1}{2(n-1)} \int_{(i-1)/n}^{i/n} |f''(t)| \, \mathrm{d}t.$$

This implies the following upper bound on a piece of $\|f'\|_1$:

$$\int_{(i-1)/n}^{i/n} |f'(x) - A_n(f)'(x)| \, \mathrm{d}x$$

$$\leq (n-1) \int_{(i-1)/n}^{i/n} \int_{(i-1)/n}^{i/n} |w(t,x)||f''(t)| \mathrm{d}t \, \mathrm{d}x$$

$$\leq (n-1) \int_{(i-1)/n}^{i/n} 2[t-(i-1)/n](i/n-t)|f''(t)| \, \mathrm{d}t$$

$$\leq (n-1) \max_{(i-1)/n \leq t \leq i/n} |2(t-x_i)(i/n-t)| \int_{(i-1)/n}^{i/n} |f''(t)| \, \mathrm{d}t$$

$$\leq \frac{1}{2(n-1)} \int_{(i-1)/n}^{i/n} |f''(t)| \, \mathrm{d}t.$$

At this point it is worth reminding ourselves that algorithms, such as the trapezoidal rule, $T_n(f)$, Simpson's rule, and any automatic quadrature rule that one can imagine, are based on limited information about the integrand, namely, a finite number of function values.

4. Acknowledgements

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.