Constructing Guaranteed Automatic Numerical Algorithms for Univariate Integration

Yizhi Zhang, Joint work with Prof. Fred Hickernell

Department of Applied Mathematics, Illinois Institute of Technology

February 24, 2017

Contents

- Introduction.
- Automatic Numerical Integration, Theory and Practice.
- GAIL
- Future Work

Motivation

- Non-adaptive methods provide guarantees.
- Adaptive methods provide no guarantee.

Non-adaptive, guaranteed

The user provides a function, the error tolerance, and some conditions.

$$x \mapsto f(x), \varepsilon = \text{tolerance}, \sigma \text{ such that } \|f''\|_1 \leq \sigma.$$

For example, using the trapezoidal rule to compute $\int_a^b f(x)dx$:

$$\text{cost } = n + 1 = \left\lceil \sqrt{\frac{\sigma}{8\varepsilon}} \right\rceil + 1.$$

We can have an estimation of $T_n(f)$ such that

$$\left| \int_a^b f(x)dx - T_n(f) \right| \leq \varepsilon.$$

Guaranteed!

Adaptive, but not guaranteed

The user provides a function, and the error tolerance.

$$x \mapsto f(x), \varepsilon = \text{tolerance.}$$

For example, using MATLAB's integral to compute $\int_a^b f(x)dx$, the cost depends on how hard the problem is. But, there is no guarantee to achieve an estimation of $Q_n(f)$ such that

$$\left| \int_a^b f(x)dx - Q_n(f) \right| \leq \varepsilon,$$

What do we think of *automatic*

- By "automatic", it is meant that the user provides a function f and an error tolerance, ε, and the algorithm attempts to provide an approximate solution that is within a distance of ε of the true solution. The algorithm will adaptively decide how many and which pieces of function data are needed.
- We want to establish a framework for providing rigorous guarantees for automatic algorithms.

Trapezoidal Rule

The problem to be solved is univariate integration on interval $[a, b]$, $\mathrm{INT}(f) := \int_a^b f(x)\,\mathrm{d}x \in \mathcal{G} := \mathbb{R}$. The space of input functions is $\mathcal{V}^3$, the space of functions whose first derivatives have finite variation:

$$\mathcal{V}^1[a, b] = \{f \in C^1[a, b] : \mathrm{Var}(f') < \infty\},$$

The space of outputs is the real space $\mathbb{R}$.

The cone of the integrand is defined as

$$\mathcal{C}_{\tau_{a,b}} := \left\{ f \in \mathcal{V}^1 : \mathrm{Var}(f') \le \frac{\tau_{a,b}}{b-a} \left\| f' - \frac{f(b) - f(a)}{b-a} \right\|_1 \right\}.$$

Simpson's Rule

Now we use Simpson's rule based on an even number of $3n$ intervals. The space of input functions is $\mathcal{V}^3$, the space of functions with continuous first and second derivatives and third derivatives having finite variation:

$$\mathcal{V}^3[a,b] = \{f \in C^1[a,b] : \mathrm{Var}(f''') < \infty\},$$

The cone of the integrand is defined as:

$$\mathcal{C} := \Big\{ f \in \mathcal{V}^3, \mathrm{Var}(f''') \le \mathfrak{C}(\text{size}(\{x_j\}_{j=0}^{n+1}))\widehat{V}(f''', \{x_j\}_{j=0}^{n+1}), \text{ for all choices of } n \in \mathbb{N}, \text{and } \{x_j\}_{j=0}^{n+1} \text{with size}(\{x_j\}_{j=0}^{n+1}) < \mathfrak{h} \Big\},$$

where $\mathfrak{C}(\text{size}(\{x_j\}_{j=0}^{n+1}))$ is the inflation factor. The cut-off value $\mathfrak{h}$ and the inflation factor $\mathfrak{C}$ define the cone. The choice of $\mathfrak{C}$ is flexible. But it must be non-decreasing. One choice could be $\mathfrak{C}(h) = \mathfrak{C}(0)\frac{\mathfrak{h}}{\mathfrak{h}-h}, \mathfrak{C}(0) \ge 1$.

Simpson's Rule, continued

From reference, the error bound of Simpson's rule is related to the variation of the third derivatives of the function to be integrated:

$$\mathrm{err}(f,n) \le \overline{\mathrm{err}}(f,n) := \frac{(b-a)^4 \operatorname{Var}(f''')}{5832n^4}. \tag{1}$$

We do not have the variation of the third derivative of the function. In order to find the error bound, $\widehat{V}$ is introduced: Given any partition $\{x_j\}_{j=0}^{n+1}$, where $a = x_0 \le x_1 \le \cdots \le x_n \le x_{n+1} = b$, define an approximation to $\operatorname{Var}(f''')$ as:

$$\widehat{V}(f''', \{x_j\}_{j=0}^{n+1}) = \sum_{j=1}^{n-1} |f'''(x_{j+1}) - f'''(x_j)|.$$

By definition, the approximation is actually a lower bound of $\operatorname{Var}(f''')$:

$$\widehat{V}(f''', \{x_j\}_{j=0}^{n+1}) \le \operatorname{Var}(f'''), \quad \forall f \in \mathcal{C}, \quad \{x_j\}_{j=0}^{n+1}, \quad n \in \mathbb{N}. \tag{2}$$

Simpson's Rule, continued

We cannot use $\widehat{V}(f''', \{x_j\}_{j=0}^{n+1})$ to approximate $\operatorname{Var}(f''')$ because it depends on values of f''', not values of f. However, $\widehat{V}(f''', \{x_j\}_{j=0}^{n+1})$ is closely related to the following approximation to $\operatorname{Var}(f''')$:

$$\widetilde{V}_n(f) = \frac{27n^3}{(b-a)^3} \sum_{j=1}^{n-1} |f(t_{3j+3}) - 3f(t_{3j+2}) + 3f(t_{3j+1}) \\ -2f(t_{3j}) + 3f(t_{3j-1}) - 3f(t_{3j-2}) + f(t_{3j-3})| \,, \quad (3)$$

where the t_i's are uniformly distributed between $[a, b]$

$$t_i = a + \frac{i(b-a)}{3n}, \qquad i = 0, \ldots, 3n, \qquad n \in \mathbb{N}. \quad (4)$$

Simpson's Rule, continued

We use divided differences to explain (3). Let $h = t_{i+1} - t_i = (b-a)/3n$ and

$$f[t_i, t_{i-1}, t_{i-2}, t_{i-3}] = \frac{f(t_i) - 3f(t_{i-1}) + 3f(t_{i-2}) - f(t_{i-3})}{6h^3}, \text{ for } i = 3, \cdots, 3n.$$

According to Mean Value Theorem for divided differences,

$$\begin{aligned} f'''(x_j) &= \frac{f(t_{3j}) - 3f(t_{3j-1}) + 3f(t_{3j-2}) - f(t_{3j-3})}{h^3}, \\ &= \frac{27n^3}{(b-a)^3}[f(t_{3j}) - 3f(t_{3j-1}) + 3f(t_{3j-2}) - f(t_{3j-3})]. \end{aligned} \quad (5)$$

Simpson's Rule, continued

If we combine (3) and (5) together, we obtain

$$\widetilde{V}_n(f) = \sum_{j=1}^{n-1} |f'''(x_{j+1}) - f'''(x_j)| = \widehat{V}(f''', \{x_j\}_{j=0}^{n+1}). \tag{6}$$

Then we can use $\widetilde{V}_n(f)$ to approximate $\mathrm{Var}(f''')$ by just using function values.

Simpson's Rule, continued

Lemma

For all $f \in \mathcal{C}$, *it follows that* $\widetilde{V}_n(f) \le \mathrm{Var}(f''') \le \mathfrak{C}(2(b-a)/n)\widetilde{V}_n(f)$, *for* $n > 2(b-a)/\mathfrak{h}$.

Proof.

By (2) and (6), $\widetilde{V}_n(f) \le \mathrm{Var}(f''')$.
Moreover, for any $x_j \in (t_{3j-3}, t_{3j})$ and $x_{j+1} \in (t_{3j}, t_{3j+3})$, it follows that $x_{j+1} - x_j \le t_{3j+3} - t_{3j-3} = 6h = 2(b-a)/n$. So by (**??**), (6), the assumption that $\mathfrak{C}$ is non-decreasing, and the fact that $2(b-a)/n \ge \mathrm{size}(\{x_j\}_{j=0}^{n+1})$,

$$\widetilde{V}_n(f) = \widehat{V}(f''', \{x_j\}_{j=0}^{n+1}) \ge \frac{\mathrm{Var}(f''')}{\mathfrak{C}(\mathrm{size}\{x_j\}_{j=0}^{n+1})} \ge \frac{\mathrm{Var}(f''')}{\mathfrak{C}(2(b-a)/n)}.$$

□

Simpson's Rule, continued

Then data-based upper bound on $\mathrm{Var}(f''')$ in this lemma can be combined with the error bound in (1) to provide the following data-based error bound:

$$\begin{aligned}\mathsf{err}(f,n) \le \overline{\mathsf{err}}(f,n) &= \frac{(b-a)^4\,\mathrm{Var}(f''')}{5832n^4}\\ &\le \frac{(b-a)^4\mathfrak{C}(2(b-a)/n)\widetilde{V}_n(f)}{5832n^4} =: \widetilde{\mathsf{err}}(f,n), \forall n > 2(b-a)/\mathfrak{h}. \qquad (7)\end{aligned}$$

Simpson's Rule, continued

Therefore, we can use Simpson's rule using $3n$ intervals, where $3n$ is an even number, to approximate integrals such that the error bound is guaranteed in (7):

$$
\begin{aligned}
&S_n(f) := \int_a^b A_n(f)\,\mathrm{d}t \\
&= \frac{(b-a)}{18n}[f(t_0) + 4f(t_1) + 2f(t_2) + 4f(t_3) + 2f(t_4) \cdots + 4f(t_{3n-1}) + f(t_{3n})].
\end{aligned} \tag{8}
$$

where t_i's are defined in (4) and $n/2 \in \mathbb{N}$.

Multi-step Automatic Algorithms

Algorithm (Adaptive Univariate Integration)

Given an interval $[a, b]$, an inflation function, $\mathfrak{C}$, a positive key mesh size $\mathfrak{h}$, a positive error tolerance, ε, and a routine for generating values of the integrand, f, set $l = 1$, and $n_1 = 2(\lfloor (b-a)/\mathfrak{h} \rfloor + 1)$.

Stage 1 Compute the error estimate $\widetilde{\text{err}}(f, n_l)$ according to (7).

Stage 2 If $\widetilde{\text{err}}(f, n_l) \le \varepsilon$, then return the Simpson's rule approximation $S_{n_l}(f)$ as the answer.

Stage 3 Otherwise let $n_{l+1} = \max(2, m)\eta_l$, where
$$m = \min\{r \in \mathbb{N} : \eta(rn_l)\widetilde{V}_{nl(f)} \le \varepsilon\},$$
$$\text{with } \eta(n) := \frac{(b-a)^4 \mathfrak{C}(2(b-a)/n)}{5832n^4}.$$
increase l by one, and go to 1.

Upper Bound of Computational Cost

Theorem

Let $N(f,\varepsilon)$ *denote the final number of* n_l *in Stage 2 when the algorithm terminates. Then this number is bounded below and above in terms of the true, yet unknown,* $\mathrm{Var}(f''')$.

$$\max\left(\left\lfloor\frac{2(b-a)}{\mathfrak{h}}\right\rfloor+1,\left\lceil(b-a)\left(\frac{\mathrm{Var}(f''')}{5832\varepsilon}\right)^{1/4}\right\rceil\right)\le N(f,\varepsilon)$$
$$\le 2\min\left\{n\in\mathbb{N}:n\ge 2\left(\left\lfloor\frac{(b-a)}{\mathfrak{h}}\right\rfloor+1\right),\eta(n)\,\mathrm{Var}(f''')\le\varepsilon\right\}$$
$$\le 2\min_{0<\alpha\le 1}\max\left(2\left(\left\lfloor\frac{(b-a)}{\alpha\mathfrak{h}}\right\rfloor+1\right),(b-a)\left(\frac{\mathfrak{C}(\alpha\mathfrak{h})\,\mathrm{Var}(f''')}{5832\varepsilon}\right)^{1/4}+1\right). \quad (9)$$

The number of function values required by the algorithm is $3N(f,\varepsilon)+1$.

Upper Bound of Computational Cost

Proof.

No matter what inputs f and ε are provided, $N(f,\varepsilon) \geq n_1 = 2(\lfloor (b-a)/\mathfrak{h} \rfloor + 1)$. Then the number of intervals increases until $\widetilde{\mathrm{err}}(f,n) \leq \varepsilon$, which by (7) implies that $\overline{\mathrm{err}}(f,n) \leq \varepsilon$. This implies the lower bound on $N(f,\varepsilon)$.

Let L be the value of l for which Algorithm 1 terminates. Since n_1 satisfies the upper bound, we may assume that $L \geq 2$. Let m be the integer found in Step 3, and let $m^* = \max(2,m)$. Note that $\eta((m^*-1)n_{L-1}) \operatorname{Var}(f''') > \varepsilon$. For $m^* = 2$, this is true because $\eta(n_{L-1}) \operatorname{Var}(f''') \geq \eta(n_{L-1}) \widetilde{V}_{n_{L-1}}(f) = \widetilde{\mathrm{err}}(f, n_{L-1}) > \varepsilon$. For $m^* = m > 2$ it is true because of the definition of m. Since η is a decreasing function, it follows that

$$(m^*-1)n_{L-1} < n^* := \min\left\{ n \in \mathbb{N} : n \geq \left\lfloor \frac{2(b-a)}{n} \right\rfloor + 1, \eta(n) \operatorname{Var}(f''') \leq \varepsilon \right\}.$$

□

Upper Bound of Computational Cost

Proof.

Therefore $n_L = m^* n_{L-1} < m^* \frac{n^*}{m^*-1} = \frac{m^*}{m^*-1} n^* \leq 2n^*$. To prove the latter part of the upper bound, we need to prove that

$$n^* \leq \max\left(\left\lfloor \frac{2(b-a)}{\alpha \mathfrak{h}} \right\rfloor + 1, (b-a) \left(\frac{\mathfrak{C}(\alpha \mathfrak{h}) \operatorname{Var}(f''')}{5832 \varepsilon} \right)^{1/4} + 1 \right), \quad 0 < \alpha < 1.$$

$\square$

Upper Bound of Computational Cost

Proof.

For fixed $\alpha \in (0,1]$, we only need to consider that case where $n^* > \lfloor 2(b-a)/(\alpha\mathfrak{h}) \rfloor + 1$. This implies that $n^* - 1 > \lfloor 2(b-a)/(\alpha\mathfrak{h}) \rfloor \geq 2(b-a)/(\alpha\mathfrak{h})$ thus $\alpha\mathfrak{h} \geq 2(b-a)/(n^*-1)$. Also by the definition of n^*, η, and $\mathfrak{C}$ is non-decreasing:

$$\begin{aligned}
&\eta(n^*-1)\operatorname{Var}(f''') > \varepsilon, \\
\Rightarrow\ & 1 < \left(\frac{\eta(n^*-1)\operatorname{Var}(f''')}{\varepsilon}\right)^{1/4}, \\
\Rightarrow\ & n^*-1 < n^*-1\left(\frac{\eta(n^*-1)\operatorname{Var}(f''')}{\varepsilon}\right)^{1/4}, \\
&= n^*-1\left(\frac{(b-a)^4\mathfrak{C}(2(b-a)/(n^*-1))\operatorname{Var}(f''')}{5832(n^*-1)^4\varepsilon}\right)^{1/4}, \\
&\leq (b-a)\left(\frac{\mathfrak{C}(\alpha\mathfrak{h})\operatorname{Var}(f''')}{5832\varepsilon}\right)^{1/4}.
\end{aligned}$$

This completes the prove of latter part of the upper bound. □

Lower Bound of Computational Cost

Building fooling function:

$$\mathsf{bump}(x;t,h) := \begin{cases} (x-t)^3/6, & t \le x < t+h, \\ [-3(x-t)^3 + 12h(x-t)^2 - 12h^2(x-t) + 4h^3]/6, & t+h \le x < t+2h \\ [3(x-t)^3 - 24h(x-t)^2 + 60h^2(x-t) - 44h^3]/6, & t+2h \le x < t+3 \\ (t+4h-x)^3/6, & t+3h \le x < t+4 \\ 0, & \text{otherwise}, \end{cases} \tag{10a}$$

$$\operatorname{Var}(\mathsf{bump}'''(\cdot;t,h)) \le 16 \text{ with equality if } a < t < t+4h < b, \tag{10b}$$

$$\int_a^b \mathsf{peak}(x;t,h)dx = h^4. \tag{10c}$$

Lower Bound of Computational Cost

Theorem

Let int be any (possibly adaptive) algorithm that succeeds for all integrands in $\mathcal{C}$, and only uses function values. For any error tolerance $\varepsilon > 0$ and any arbitrary value of $\mathrm{Var}(f''')$, there will be some $f \in \mathcal{C}$ for which int must use at least

$$-\frac{5}{4} + \frac{b-a-5\mathfrak{h}}{8}\left[\frac{[\mathfrak{C}(0)-1]\,\mathrm{Var}(f''')}{\varepsilon}\right]^{1/4} \tag{11}$$

function values. As $\mathrm{Var}(f''')/\varepsilon \to \infty$ the asymptotic rate of increase is the same as the computational cost of `integral`.

Numerical Example

Consider the family of bump test functions defined by

$$f(x) = \begin{cases} \beta[4\alpha^2 + (x-z)^2 - (x-z-\alpha)|x-z-\alpha| \\ \qquad -(x-z+\alpha)|x-z+\alpha|], & z-2\alpha \le x \le z+2\alpha, \\ 0, & \text{otherwise.} \end{cases} \tag{12}$$

with $\log_{10}(\alpha) \sim \mathcal{U}[-4,-1]$, $z \sim \mathcal{U}[2\alpha, 1-2\alpha]$, and $\beta = 1/(4\alpha^3)$ chosen to make $\int_0^1 f(x)\,\mathrm{d}x = 1$. It follows that $|f|_{\widetilde{\mathcal{F}}} = 1/\alpha$ and $\mathrm{Var}(f') = 2/\alpha^2$. The probability that $f \in \mathcal{C}_\tau$ is $\min\left(1, \max(0, \left(\log_{10}(\tau/2) - 1\right)/3)\right)$.

Experiment Setup

As an experiment, we chose 10000 random test functions and applied Algorithm **??** with an error tolerance of $\varepsilon = 10^{-8}$ and initial τ values of $10, 100, 1000$. The algorithm is considered successful for a particular f if the exact and approximate integrals agree to within ε. The success and failure rates are given in Table 1. Our algorithm imposes a cost budget of $N_{\max} = 10^7$. The probability that f initially lies in $\mathcal{C}_\tau$ is the smaller number in the third column of Table 1, while the larger number is the empirical probability that f eventually lies in $\mathcal{C}_\tau$ after possible increases in τ made by Stage 2 of Algorithm **??**. For this experiment Algorithm **??** was successful for all f that finally lie inside $\mathcal{C}_\tau$, for which there was no warning. It was also successful for a small percentage of functions lying outside the cone.

Results

	τ	$\mathrm{Prob}(f \in \mathcal{C}_\tau)$	Success No Warning	Success Warning	Failure No Warning
Algorithm **??**	10	$0\% \to 25\%$	25%	$< 1\%$	75%
	100	$23\% \to 58\%$	56%	2%	42%
	1000	$57\% \to 88\%$	68%	20%	12%
`quad`			8%		92%
`integral`			19%		81%
`chebfun`			29%		71%

Table: The probability of the test function lying in the cone for the original and eventual values of τ and the empirical success rate of Algorithm **??** plus the success rates of other common quadrature algorithms.

Guaranteed Automatic Integration Library (GAIL)

The ideas presented here are being implemented in MATLAB code (code.google.com\p\gail), which also include:

- Automatic univariate function recovery via linear splines.
- Guaranteed automatic Monte Carlo algorithm for multidimensional integration.
- Guaranteed automatic quasi-Monte Carlo algorithm for multidimensional integration.
- And more.

Future Work

- Guaranteed automatic algorithms with higher order convergence rate.
- Locally adaptive algorithms.
- Relative Error.

References 1

Clancy N, Ding Y, Hamilton C, Hickernell FJ, Zhang Y (2013) The complexity of guaranteed automatic algorithms: Cones, not balls. Submitted for publication, arXiv.org:1303.2412 [math.NA]

Lower Bound of Computational Cost

Now we compute the lower bound by constructing fooling functions. We choose the triangle shaped function $f_0 : x \mapsto 1/2 - |1/2 - x|$. Then

$$|f_0|_{\widetilde{\mathcal{F}}} = \|f_0' - f_0(1) + f_0(0)\|_1 = \int_0^1 |\mathrm{sign}(1/2 - x)| \, \mathrm{d}x = 1,$$
$$|f_0|_{\mathcal{F}} = \mathrm{Var}(f_0') = 2 = \tau_{\min}.$$

Lower Bound of Computational Cost, continued

For any $n \in \mathcal{J} := \mathbb{N}_0$, suppose that the one has the data $L_i(f) = f(\xi_i)$, $i = 1, \ldots, n$ for arbitrary ξ_i, where $0 = \xi_0 \le \xi_1 < \cdots < \xi_n \le \xi_{n+1} = 1$. There must be some $j = 0, \ldots, n$ such that $\xi_{j+1} - \xi_j \ge 1/(n+1)$. The function f_1 is defined as a triangle function on the interval $[\xi_j, \xi_{j+1}]$:

$$f_1(x) := \begin{cases} \dfrac{\xi_{j+1} - \xi_j - |\xi_{j+1} + \xi_j - 2x|}{8} & \xi_j \le x \le \xi_{j+1}, \\ 0 & \text{otherwise.} \end{cases}$$

Lower Bound of Computational Cost, continued

This is a piecewise linear function whose derivative changes from 0 to $1/4$ to $-1/4$ to 0 provided $0 < \xi_j < \xi_{j+1} < 1$, and so $|f_1|_{\mathcal{F}} = \mathrm{Var}(f_1') \le 1$. Moreover,

$$\mathrm{INT}(f) = \int_0^1 f_1(x)\,\mathrm{d}x = \frac{(\xi_{j+1} - \xi_j)^2}{16} \ge \frac{1}{16(n+1)^2} =: g(n),$$

$$g^{-1}(\varepsilon) = \left\lceil \sqrt{\frac{1}{16\varepsilon}} \right\rceil - 1.$$

Using these choices of f_0 and f_1, along with the corresponding g above, we can have that the complexity of the integration problem over the cone of functions $\mathcal{C}_\tau$ is bounded below as

$$\mathrm{comp}(\varepsilon, \mathcal{A}(\mathcal{C}_\tau, \mathbb{R}, \mathrm{INT}, \Lambda^{\mathrm{std}}), \mathcal{B}_s) \ge \left\lceil \sqrt{\frac{(\tau - 2)s}{32\tau\varepsilon}} \right\rceil - 1.$$