Constructing Guaranteed Automatic Numerical Algorithms for Univariate Integration

Yizhi Zhang

Department of Applied Mathematics, Illinois Institute of Technology

November 29, 2013

Contents

- Problem Description
- Demo
- Future Work

Expanding from $[0, 1]$ to $[a, b]$

- It is not as straight forward as one may think of. Previously we use the input ninit as the initial number of points.
- The algorithm could fail.
- The algorithm may not be efficient.

Our Solution

initial number of point $=$ ninit

Our Solution

$$\text{initial number of point} = \text{ninit} = n_{\text{ninit,max}} \left(\frac{n_{\text{ninit,min}}}{n_{\text{ninit,max}}} \right)^{\frac{1}{1+b-a}}$$

Assumptions

The node set and the linear spline algorithm using n function values are defined for $n \in \mathcal{I} := \{2, 3, \ldots\}$ as follows:

$$x_i = a + \frac{i-1}{n-1}(b-a), \qquad i = 1, \ldots, n,$$

$$A_n(f)(x) := \frac{n-1}{b-a}\left[f(x_i)(x_{i+1}-x) + f(x_{i+1})(x-x_i)\right]$$

for $x_i \le x \le x_{i+1}$.

Assumptions, continued

The problem to be solved is univariate integration on the unit interval, $S(f) := \mathrm{INT}(f) := \int_a^b f(x)\,\mathrm{d}x \in \mathcal{G} := \mathbb{R}$. The fixed cost building blocks to construct the adaptive integration algorithm are the composite trapezoidal rules based on $n-1$ trapezoids:

$$T_n(f) := \int_a^b A_n(f)\,\mathrm{d}x = \frac{b-a}{2n-2}[f(x_1)+2f(x_2)+\cdots+2f(x_{n-1})+f(x_n)].$$

Assumptions, continued

The space of input functions is $\mathcal{V}$, 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 stronger semi-norm is $|f|_{\mathcal{F}} = \mathrm{Var}(f')$, while the weaker semi-norm is

$$|f|_{\widetilde{\mathcal{F}}} := \left\|f' - A_2(f)'\right\|_1 = \left\|f' - \frac{f(b)-f(a)}{b-a}\right\|_1 = \mathrm{Var}(f - A_2(f)),$$

The cone of the integrand is defined as

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

For simplicity, I will denote $\tau_{a,b}$ as τ for the rest of the context.

Multi-step Automatic Algorithms

Algorithm (Adaptive Univariate Integration)

Let the sequence of algorithms $\{T_n\}_{n\in\mathcal{I}}$, $\{\widetilde{F}_n\}_{n\in\mathcal{I}}$, and $\{F_n\}_{n\in\mathcal{I}}$ be as described above. Choose integer $n_{\text{lo}}, n_{\text{hi}}$, such that $n_{\text{lo}} \le n_{\text{hi}}$. Set $i = 1$. Let $n_1 = \max\left\{\lceil n_{\text{hi}}\left(\frac{n_{\text{lo}}}{n_{\text{hi}}}\right)^{\frac{1}{1+b-a}}\rceil, 3\right\}$. Let $\tau_{a,b} = 2n_1 - 3$. For any error tolerance ε and input function f, do the following:

Stage 1. Estimate $\left\|f' - \frac{f(b)-f(a)}{b-a}\right\|_1$ and bound $\mathrm{Var}(f')$. Compute $\widetilde{F}_{n_i}(f)$ and $F_{n_i}(f)$

Multi-step Automatic Algorithms, continuoued

Algorithm

Stage 2. Check the necessary condition for $f \in \mathcal{C}_{\tau_{a,b}}$. Compute

$$\tau_{\min,n_i} = \frac{F_{n_i}(f)}{\widetilde{F}_{n_i}(f) + (b-a)F_{n_i}(f)/(2n_i - 2)}.$$

If $\tau_{a,b} \geq \tau_{\min,n_i}$, then go to stage 3. Otherwise, set $\tau_{a,b} = 2\tau_{\min,n_i}$. If $n_i \geq (\tau+1)/2$, then go to stage 3. Otherwise, choose

$$n_{i+1} = 1 + (n_i - 1)\left\lceil \frac{\tau_{a,b}+1}{2n_i - 2} \right\rceil.$$

Go to Stage 1.

Multi-step Automatic Algorithms, continued

Algorithm

Stage 3. Check for convergence. Check whether n_i is large enough to satisfy the error tolerance, i.e.

$$\widetilde{F}_{n_i}(f) \leq \frac{4\varepsilon(n_i-1)(2n_i-2-\tau_{a,b}(b-a))}{\tau_{a,b}(b-a)^2}.$$

If this is true, then return $T_{n_i}(f)$ and terminate the algorithm. If this is not true, choose

$$n_{i+1} = 1+(n_i-1)\max\left\{2, \left\lceil \frac{1}{(n_i-1)}\sqrt{\frac{\tau_{a,b}(b-a)\widetilde{F}_{n_i}(f)}{8\varepsilon}} \right\rceil \right\}.$$

Go to Stage 1.

Demo

Demo

Future Work

- Verify my theoretic proof of this case.
- Double check the MATLAB code.
- Polishing and debugging.

References 1

Clancy N, Ding Y, Hamilton C, Hickernell FJ, Zhang Y (2013) The complexity of guaranteed automatic algorithms: Cones, not balls. DOI 10.1016/j.jco.2013.09.002., arXiv.org:1303.2412 [math.NA]