A Locally Adaptive Algorithm for Global Minimization of Univariate Functions

Xin Tong

Department of Mathematics, Statistics and Computer Science
University of Illinois at Chicago (UIC)

SIAM Annual Meeting 2018

July 9, 2018

Outline

- Introduction
- The Algorithm
- Numerical Examples
- Improvements

Introduction

Collaborators on this project:

- Professor Fred Hickernell and Professor Sou-Cheng Choi at Illinois Institute of Technology (IIT)
- Yuhan Ding and the GAIL team

Motivation:

- **fminbnd** in Matlab[1]: may report a local minimum
- Linear spline is used to construct L_∞ approximation of univariate functions[2].
- We constructed a globally adaptive algorithm for univariate function minimization[3].

[1] R. P. (Richard Peirce) Brent. *Algorithms for minimization without derivatives*. Englewood Cliffs, N.J. : Prentice-Hall, 1973.

[2] N. Clancy et al. "The Cost of Deterministic, Adaptive, Automatic Algorithms: Cones, Not Balls". In: *Journal of Complexity* 30 (2014), pp. 21–45.

[3] X. Tong. "A Guaranteed, Adaptive, Automatic Algorithm for Univariate Function Minimization". MA thesis. Illinois Institute of Technology, 2014.

Problem Description

Locally adaptive algorithms for global minimization problems:

For some suitable set, 'cone' $\mathcal{C}$, real-valued functions defined on a finite interval $[a, b]$, we construct algorithm $M : (\mathcal{C}, (0, \infty)) \to \mathbb{R}$ such that for any $f \in \mathcal{C}$ and any error tolerance $\varepsilon > 0$,

$$0 \le M(f, \varepsilon) - \min_{a \le x \le b} f(x) \le \varepsilon.$$

Linear Spline

- The Algorithm is based on the *linear spline* for $x \in [x_{i-1}, x_i]$ by

$$S(f, x_{0:n})(x) = \frac{x - x_i}{x_{i-1} - x_i} f(x_{i-1}) + \frac{x - x_{i-1}}{x_i - x_{i-1}} f(x_i), \quad i \in 1{:}n.$$

  $x_{0:n}$ is an ordered sequence of $n+1$ points including the endpoints of the interval, i.e., $a =: x_0 < x_1 < \cdots < x_{n-1} < x_n := b$.
- The error of the linear spline is bounded in terms of the second derivative of the input function as follows

$$\|f - S(f, x_{0:n})\|_{[x_{i-1},x_i]} \le \frac{(x_i - x_{i-1})^2 \, \|f''\|_{[x_{i-1},x_i]}}{8}, \quad i \in 1{:}n,$$

  where $\|f\|_{[\alpha,\beta]}$ denotes the L^∞-norm of f restricted to the interval $[\alpha, \beta] \subseteq [a, b]$.
- This error bound leads us to focus on input functions in the Sobolev space $W^{2,\infty} := W^{2,\infty}[a, b] := \{f \in C^1[a, b] : \|f''\| < \infty\}$.

An Upper Bound

$$|\min f(x) - S(f, x_{0:n})| \leq \sup\{|f(x) - S(f, x_{0:n})|\}, \quad x \in [x_{i-1}, x_i]$$

Take the minimum on each interval $[x_{i-1}, x_i]$:

$$0 \leq \min(f(x_{i-1}), f(x_i)) - \min f(x) \leq \frac{(x_i - x_{i-1})^2 \|f''\|_{[x_{i-1}, x_i]}}{8}.$$

- For each subinterval, we take the minimum value of function, $\min(f(x_{i-1}), f(x_i))$, as the approximation, then the error in that subinterval has an upper bound.
- Next, if we take $\min\limits_{i \in 0:n} f(x_i)$ as a candidate for $\min_{a \leq x \leq b} f(x)$, then we demand that each upper bound of the subinterval is less than the tolerance ε, i.e. $\frac{(x_i - x_{i-1})^2 \|f''\|_{[x_{i-1}, x_i]}}{8} \leq \varepsilon$.
- Question: What is a proper bound/approximation on $\|f''\|_{[x_{i-1}, x_i]}$? Remember that we want the second derivatives f'' do not change dramatically over a short distance. We will define a set of such functions.

Idea of defining the Cone of function set

For any subinterval $[\alpha, \beta]$, we use quadratic Newton's Interpolation polynomial at nodes $\{\alpha, (\alpha+\beta)/2, \beta\}$ to compute

$$\|f''\|_{-\infty,[\alpha,\beta]} := \inf_{\alpha \le \eta < \zeta \le \beta} \left| \frac{f'(\zeta) - f'(\eta)}{\zeta - \eta} \right| \le 2\,|D(f,\alpha,\beta)| \le \sup_{\alpha \le \eta < \zeta \le \beta} \left| \frac{f'(\zeta) - f'(\eta)}{\zeta - \eta} \right| = \|f''\|_{[\alpha,\beta]},$$

with the divided difference $D(f,\alpha,\beta) := \frac{2f(\beta) - 4f((\alpha+\beta)/2)) + 2f(\alpha)}{(\beta-\alpha)^2}$.

- $2\,|D(f,\alpha,\beta)|$ is an *upper* bound for $\|f''\|_{-\infty,[\alpha,\beta]}$.
- $2\,|D(f,\alpha,\beta)|$ is a *lower* bound for $\|f''\|_{[\alpha,\beta]}$.
- We define the *Cone* of interesting functions, $\mathcal{C}$, containing f for which $\|f''\|_{[\alpha,\beta]}$ is not drastically greater than the maximum of $\|f''\|_{-\infty,[\beta-h_-,\alpha]}$ and $\|f''\|_{-\infty,[\beta,\alpha+h_+]}$ with $h_\pm > \beta - \alpha$.

Cone: definition

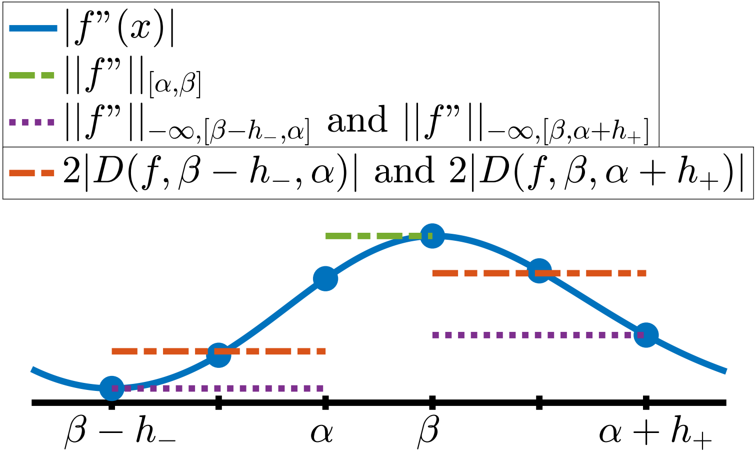
For any $[\alpha,\beta]\subset[a,b]$ and any $h_\pm$ satisfying $0<\beta-\alpha<h_\pm<\mathfrak{h}$, define

$$B(f'',\alpha,\beta,h_-,h_+):=$$
$$\begin{cases}\max\left(\mathfrak{C}(h_-)\left\|f''\right\|_{-\infty,[\beta-h_-,\alpha]},\mathfrak{C}(h_+)\left\|f''\right\|_{-\infty,[\beta,\alpha+h_+]}\right), \\ \qquad a\le\beta-h_-<\alpha+h_+\le b, \\ \mathfrak{C}(h_-)\left\|f''\right\|_{-\infty,[\beta-h_-,\alpha]}, \quad a\le\beta-h_-<b<\alpha+h_+, & \text{left end} \\ \mathfrak{C}(h_+)\left\|f''\right\|_{-\infty,[\beta,\alpha+h_+]}, \quad \beta-h_-<a<\alpha+h_+\le b. & \text{right end}\end{cases}$$

The Cone is defined as

$$\mathcal{C}:=\Big\{f\in W^{2,\infty}:\left\|f''\right\|_{[\alpha,\beta]}\le B(f'',\alpha,\beta,h_-,h_+)\text{ for all }[\alpha,\beta]\subset[a,b]$$
$$\text{and } h_\pm\in(\beta-\alpha,\mathfrak{h})\Big\}.$$

Cone: An Example



Algorithm: Motivation

On $[x_{i-1}, x_i]$:

$$\min(f(x_{i-1}), f(x_i)) - \min f(x) \le \frac{(x_i - x_{i-1})^2 \|f''\|_{[x_{i-1}, x_i]}}{8}$$

1. We estimate the right-hand-side and we want

$$\overline{\text{err}}_i := \frac{1}{8}\mathfrak{C}(3h_l)\left|f(x_{i+1}) - 2f(x_i) + f(x_{i-1})\right| \le \varepsilon, \quad \forall i.$$

2. Take $\widehat{M} = \min_{i \in 0:n} f(x_i)$ as the approximation to $\min_{[a,b]} f(x)$.
3. Rewrite $\min(f(x_{i-1}), f(x_i)) - \min f(x) \le \overline{\text{err}}_i$:

$$\text{Ture error} = \widehat{M} - \min f(x) \le \overline{\text{err}}_i + \widehat{M} - \min(f(x_{i-1}), f(x_i)).$$

We will focus on the intervals with

$$\overline{\text{err}}_i > \varepsilon$$

$$\overline{\text{err}}_i + \widehat{M} - \min(f(x_{i-1}), f(x_i)) > \epsilon$$

Algorithm[4]

For finite interval $[a, b]$, integer $n_{\text{ninit}} \geq 5$, and constant $\mathfrak{C}_0 \geq 1$. Let

$$\mathfrak{h} := \frac{3(b-a)}{n_{\text{ninit}} - 1}, \qquad \mathfrak{C}(h) := \frac{\mathfrak{C}_0 \mathfrak{h}}{\mathfrak{h} - h} \text{ for } 0 < h < \mathfrak{h}.$$

Let $f : [a, b] \to \mathbb{R}$ and $\varepsilon > 0$ be user inputs. Let $n = n_{\text{ninit}}$, and define the initial partition of equally spaced points, $x_{0:n}$, and certain index sets of subintervals:

$$x_i = a + i\frac{b-a}{n},\ i \in 0\!:\!n, \qquad \mathcal{I}_+ = 2\!:\!(n-1), \quad \mathcal{I}_- = 1\!:\!(n-2).$$

Compute $\widehat{M} = \min_{i \in 0:n} f(x_i)$. For $s \in \{+, -\}$ do the following.

[4] S.-C. T. Choi et al. "Local adaption for approximation and minimization of univariate functions". In: *Journal of Complexity* 40 (2017), pp. 17 –33.

Algorithm

Step 1. Check for convergence.
Compute $\overline{\mathrm{err}}_i = \frac{1}{8}\mathfrak{C}(3h_l)\left|f(x_{i+1}) - 2f(x_i) + f(x_{i-1})\right|$ for all $i \in \mathcal{I}_\pm$. Let $\widetilde{\mathcal{I}}_s = \{i \in \mathcal{I}_s : \overline{\mathrm{err}}_i > \varepsilon\}$.
Next compute

$$\widehat{\mathrm{err}}_{i,s} = \overline{\mathrm{err}}_i + \widehat{M} - \min\big(f(x_{i-s2}), f(x_{i-s1})\big) \quad \forall i \in \widetilde{\mathcal{I}}_s,$$

$$\widehat{\mathcal{I}}_s = \Big\{ i \in \widetilde{\mathcal{I}}_s : \widehat{\mathrm{err}}_{i,s} > \varepsilon \text{ or } \big(i - s3 \in \widetilde{\mathcal{I}}_{-s} \ \& \ \widehat{\mathrm{err}}_{i-s3,-s} > \varepsilon\big) \Big\}.$$

If $\widehat{\mathcal{I}}_+ \cup \widehat{\mathcal{I}}_- = \emptyset$, return $M(f, \varepsilon) = \widehat{M}$ and terminate the algorithm. Otherwise, continue to the next step.

Algorithm

Step 2. *Split the subintervals as needed.*

Update the present partition, $x_{0:n}$, to include the subinterval midpoints

$$\frac{x_{i-s2}+x_{i-s1}}{2}, \quad \frac{x_{i-s1}+x_i}{2} \quad \forall i \in \widehat{\mathcal{I}}_s.$$

(The point $(x_{i-2}+x_{i-1})/2$ is only included for $i \geq 2$, and the point $(x_{i+1}+x_{i+2})/2$ is only included for $i \leq n-2$.) Update the sets $\mathcal{I}_{\pm}$ to consist of the new indices corresponding to the old points

$$x_{i-s1}, \quad \frac{x_{i-s1}+x_i}{2} \text{ for } i \in \widehat{\mathcal{I}}_s.$$

(The point x_{i-1} is only included for $i \geq 2$, and the point x_{i+1} is only included for $i \leq n-2$.) Return to Step 1.

Numerical Examples: GAIL

Together with our collaborators, we have developed the Guaranteed Automatic Integration library (GAIL)[5]. This algorithm is implemented as GAIL function **funmin_g**.

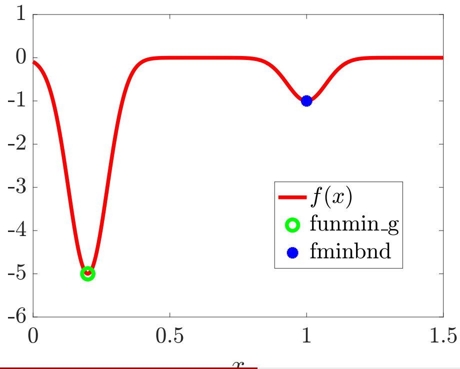
[5] S.-C. T. Choi et al. *GAIL: Guaranteed Automatic Integration Library (Versions 1.0–2.2)*. MATLAB software. 2013–2017. URL: http://gailgithub.github.io/GAIL_Dev/.

Function with two local minima

Consider the function

$$f(x) = -5\exp(-[10(x-0.2)]^2) - \exp(-100(x-1)^2) \qquad 0 \le x \le 1.5,$$

It has two local minimum points at 0.2 and 1. It attains its minimum at $x = 0.2$.



Our **funmin_g** caught the global minimum but MATLAB's **fminbnd** retured the local minimum.

Test Functions

Next, we compare our adaptive algorithms with MATLAB's **fminbnd** and Chebfun' min for random samples from the following families of test functions defined on $[-1, 1]$:

$$f_1(x) = \begin{cases} -12.5\Big[0.16 + (x-c)^2 + (x-c-0.2)\,|x-c-0.2| \\ -(x-c+0.2)\,|x-c+0.2|\Big], & |x-c| \le 0.4, \\ 0, & \text{otherwise}, \end{cases}$$

$c \sim \mathcal{U}[0, 0.6]$ Bump functions

$f_2(x) = x^4 \sin(d/x),\quad d \sim \mathcal{U}[0, 2],$ Outside the cone $\mathcal{C}$

$f_3(x) = 10x^2 + f_2(x),$ Almost quadratic

where $\mathcal{U}[a, b]$ represents a uniform distribution over $[a, b]$.

Results of Comparison

	Mean # Samples			**Success (%)**		
	fminbnd	min	funmin_g	fminbnd	min	funmin_g
f_1	8	116	111	100	14	100
f_2	22	43	48	27	60	100
f_3	9	22	108	100	35	100

- MATLAB's **fminbnd** uses far fewer function values than **funmin_g**, but it cannot locate the global minimum (at the left boundary) for about 70% of the f_2 test cases.
- Chebfun's **min**[6] uses fewer points than **funmin_g**, but Chebfun is slower and less accurate than **funmin_g** for these tests.

[6] N. Hale T. A. Driscoll and L. N. Trefethen. *Chebfun Guide*. Pafnuty Publications, Oxford, 2014.

Improvements

- Output intervals containing minima
- Lower bound of computational cost
- Higher order splines as a basis
- Interval extension: $[a, b] \to [a, \infty)$ or $(-\infty, b]$ or $(-\infty, \infty)$

Comments

Thank you!

Any Comments and Questions?