Journal of Complexity 40 (2017) 17–33

Contents lists available at ScienceDirect

Journal of Complexity

journal homepage: www.elsevier.com/locate/jco

Local adaption for approximation and minimization of univariate functions☆

Sou-Cheng T. Choi [a], Yuhan Ding [a], Fred J. Hickernell [a,*], Xin Tong [b]

[a] *Department of Applied Mathematics, Illinois Institute of Technology, RE 208, 10 West 32nd Street, Chicago, IL, 60616, USA*

[b] *Department of Mathematics, Statistics, and Computer Science, University of Illinois at Chicago, Room 322 SEO, 851 S. Morgan Street, Chicago, IL, 60607, USA*

ARTICLE INFO

Article history:
Received 31 May 2016
Accepted 9 November 2016
Available online 1 December 2016

Keywords:
Adaption
Automatic
Computational complexity
Function approximation
Function recovery
Global minimization

ABSTRACT

Most commonly used *adaptive* algorithms for univariate real-valued function approximation and global minimization lack theoretical guarantees. Our new locally adaptive algorithms are guaranteed to provide answers that satisfy a user-specified absolute error tolerance for a cone, $\mathcal{C}$, of non-spiky input functions in the Sobolev space $W^{2,\infty}[a, b]$. Our algorithms automatically determine where to sample the function—sampling more densely where the second derivative is larger. The computational cost of our algorithm for approximating a univariate function f on a bounded interval with L^∞-error no greater than ε is $\mathcal{O}\left(\sqrt{\|f''\|_{\frac{1}{2}}/\varepsilon}\right)$ as $\varepsilon \to 0$. This is the same order as that of the best function approximation algorithm for functions in $\mathcal{C}$. The computational cost of our global minimization algorithm is of the same order and the cost can be substantially less if f significantly exceeds its minimum over much of the domain. Our Guaranteed Automatic Integration Library (GAIL) contains these new algorithms. We provide numerical experiments to illustrate their superior performance.

☆ Communicated by E. Novak.

* Corresponding author.
E-mail address: hickernell@iit.edu (F.J. Hickernell).

http://dx.doi.org/10.1016/j.jco.2016.11.005
0885-064X/

1. Introduction

Our goal is to reliably solve univariate function approximation and global minimization problems by adaptive algorithms. We prescribe a suitable set, $\mathcal{C}$, of continuously differentiable, real-valued functions defined on a finite interval $[a, b]$. Then, we construct algorithms $A : (\mathcal{C}, (0, \infty)) \to L^\infty[a, b]$ and $M : (\mathcal{C}, (0, \infty)) \to \mathbb{R}$ such that for any $f \in \mathcal{C}$ and any error tolerance $\varepsilon > 0$,

$$\|f - A(f, \varepsilon)\| \le \varepsilon, \tag{APP}$$

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

Here, $\|\cdot\|$ denotes the L^∞-norm on $[a, b]$, i.e., $\|f\| = \sup_{x \in [a,b]} |f(x)|$. Algorithms A and M depend only on function values.

Our algorithms proceed iteratively until their data-dependent stopping criteria are satisfied. The input functions are sampled nonuniformly over $[a, b]$, with the sampling density determined by the function data. We call our algorithms *locally adaptive*, to distinguish them from globally adaptive algorithms that have a fixed sampling pattern and only the sample size determined adaptively.

1.1. Key ideas in our algorithms

Our Algorithms A and M are based on a *linear spline*, $S(f, x_{0:n})$ defined on $[a, b]$. Let $0 : n$ be shorthand for $\{0, \dots, n\}$, and let $x_{0:n}$ be any ordered sequence of $n + 1$ points that includes the endpoints of the interval, i.e., $a =: x_0 < x_1 < \cdots < x_{n-1} < x_n := b$. We call such a sequence a *partition*. Then given any $x_{0:n}$ and any $i \in 1 : n$, the linear spline is defined for $x \in [x_{i-1}, x_i]$ by

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

The error of the linear spline is bounded in terms of the second derivative of the input function as follows [2, Theorem 3.3]:

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

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 $W^{2,\infty} := W^{2,\infty}[a, b] := \{f \in C^1[a, b] : \left\|f''\right\| < \infty\}$.

Algorithms A and M require upper bounds on $\left\|f''\right\|_{[x_{i-1}, x_i]}$, $i \in 1 : n$, to make use of (2). A nonadaptive algorithm might assume that $\left\|f''\right\| \le \sigma$, for some known σ, and proceed to choose $n = \left\lceil (b - a)\sqrt{\sigma/(8\varepsilon)} \right\rceil$, $x_i = a + i(b - a)/n$, $i \in 0 : n$. Providing an upper bound on $\left\|f''\right\|$ is often impractical, and so we propose adaptive algorithms that do not require such information.

However, one must have some a priori information about $f \in W^{2,\infty}$ to construct successful algorithms for (APP) or (MIN). Suppose that Algorithm A satisfies (APP) for the zero function $f = 0$, and $A(0, \varepsilon)$ uses the data sites $x_{0:n} \subset [a, b]$. Then one can construct a *nonzero* function $g \in W^{2,\infty}$ satisfying $g(x_i) = 0, i \in 0 : n$ but with $\|g - A(g, \varepsilon)\| = \|g - A(0, \varepsilon)\| > \varepsilon$.

Our set $\mathcal{C} \subset W^{2,\infty}$ for which A and M succeed includes only those functions whose second derivatives do not change dramatically over a short distance. The precise definition of $\mathcal{C}$ is given in Section 2. This allows us to use second-order divided differences to construct rigorous upper bounds on the linear spline error in (2). These data-driven error bounds inform the stopping criteria for Algorithm A in Section 3.1 and Algorithm M in Section 4.1.

The computational cost of Algorithm A is analyzed in Section 3.2 and is shown to be $\mathcal{O}\left(\sqrt{\|f''\|_{\frac{1}{2}}/\varepsilon}\right)$ as $\varepsilon \to 0$. Here, $\|\cdot\|_{\frac{1}{2}}$ denotes the $L^{\frac{1}{2}}$-quasi-norm, a special case of the L^p-quasi-norm, $\|f\|_p := \left(\int_a^b |f|^p \, dx\right)^{1/p}$, $0 < p < 1$. Since $\left\|f''\right\|_{\frac{1}{2}}$ can be much smaller than $\left\|f''\right\|$, locally adaptive

algorithms can be more efficient than globally adaptive algorithms, whose computational costs are proportional to $\sqrt{\|f''\|/\varepsilon}$. The computational complexity of (APP) is determined in Section 3.3 to be $\mathcal{O}\left(\sqrt{\|f''\|_{\frac{1}{2}}/\varepsilon}\right)$ as well.

The computational cost of our optimization Algorithm M is analyzed in Section 4.2. A lower bound on the computational complexity of (MIN) is a subject for future investigation.

Our algorithms are implemented in our MATLAB [22] Guaranteed Automatic Integration Library (GAIL) [4]. Section 5 provides numerical examples of our algorithms and compares their performances with MATLAB's and Chebfun's algorithms. We note cases where our algorithms are successful in meeting the error tolerance, and other algorithms are not.

1.2. Related work on adaptive algorithms

Adaptive algorithms relieve the user of having to specify the number of samples required. Only the desired error tolerance is needed. Existing adaptive numerical algorithms for function approximation, such as the MATLAB toolbox Chebfun [10], succeed for some functions, but fail for others. No theory explains for which f Chebfun succeeds. A corresponding situation exists for minimization algorithms, such as `min` in Chebfun or MATLAB's built-in `fminbnd` [1,9].

Our theoretically justified Algorithms A and M build upon the ideas used to construct the adaptive algorithms in [7,8,11,12,14,15,23]. In all those cases, a cone, $\mathcal{C}$, of input functions is identified for which the adaptive algorithms succeed, just as is done here. However, unlike the algorithms in [7,8,11,23], the definition of $\mathcal{C}$ here does not depend on a weaker norm. Also, unlike the globally adaptive approximation and optimization algorithms in [7,23], the algorithms proposed here are locally adaptive, sampling the interval $[a, b]$ nonuniformly.

Novak [17] summarizes the settings under which adaption may provide an advantage over nonadaption. For linear problems, such as (APP), adaption has no advantage if the set of functions being considered is symmetric and convex [17, Theorem 1], [24, Chapter 4, Theorem 5.2.1], [25]. The cone $\mathcal{C}$ defined for our approximation problem (APP) is symmetric, but *not* convex. Plaskota et al. [19] have developed adaptive algorithms for functions with singularities. Our algorithms are not designed for such functions. Rather they are designed to be efficient when the second derivative is large in a small part of the domain.

Plaskota [18] has developed an adaptive Simpson's algorithm for approximating $\int_a^b f(x)\,\mathrm{d}x$ assuming that the fourth derivative $f^{(4)}(x) \geq 0$ for all $x \in [a, b]$. His algorithm relies on divided differences, like ours do. His error is asymptotically proportional to $\|f^{(4)}\|_{\frac{1}{4}}$, which is analogous to the $\|f''\|_{\frac{1}{2}}$ that appears in our analysis. Horn [13] has developed an optimization algorithm for Lipschitz continuous functions that does not require knowledge of the Lipschitz constant.

There is a significant literature on theoretically justified algorithms based on interval arithmetic [16,20], which are implemented in INTLAB [21]. This approach assumes that functions have interval inputs and outputs. We focus on the more common situation where functions have point inputs and outputs.

2. The cone, $\mathcal{C}$, of functions of interest

Linear splines (1) are the foundation for adaptive Algorithms A and M. To bound the error of the linear spline in (2), our algorithms construct data-based upper bounds on $\|f''\|_{[\alpha,\beta]}$ in terms of divided differences. For these bounds to hold, we must assume that $f''(x)$ does not change drastically with respect to a small change in x. These assumptions define our cone of functions, $\mathcal{C}$, for which our algorithms ultimately apply.

Let p denote the quadratic Lagrange interpolating polynomial at the nodes $\{\alpha, (\alpha+\beta)/2, \beta\}$, which may be written as

$$p(x) := f(\alpha) + \frac{(x-\alpha)[f(\beta)-f(\alpha)]}{\beta-\alpha} + (x-\alpha)(x-\beta)D(f,\alpha,\beta),$$
$$D(f,\alpha,\beta) := \frac{2f(\beta) - 4f((\alpha+\beta)/2) + 2f(\alpha)}{(\beta-\alpha)^2}. \tag{3}$$

For any $f \in W^{2,\infty}$, the function $f-p$ has at least three distinct zeros on $[\alpha,\beta]$, so $f'-p'$ has at least two distinct zeros on (α,β). Specifically, there exist $\xi_\pm$ with $\alpha < \xi_- < (\alpha+\beta)/2 < \xi_+ < \beta$ with $f'(\xi_\pm) - p'(\xi_\pm) = 0$. Thus,

$$\begin{aligned}\left\|f''\right\|_{-\infty,[\alpha,\beta]} &:= \inf_{\alpha\le\eta<\zeta\le\beta}\left|\frac{f'(\zeta)-f'(\eta)}{\zeta-\eta}\right| \\ &\le \left|\frac{f'(\xi_+)-f'(\xi_-)}{\xi_+-\xi_-}\right| = \left|\frac{p'(\xi_+)-p'(\xi_-)}{\xi_+-\xi_-}\right| = 2\left|D(f,\alpha,\beta)\right| \\ &\le \sup_{\alpha\le\eta<\zeta\le\beta}\left|\frac{f'(\zeta)-f'(\eta)}{\zeta-\eta}\right| =: \left\|f''\right\|_{[\alpha,\beta]}.\end{aligned} \tag{4}$$

This inequality tells us that twice the divided difference, $2|D(f,\alpha,\beta)|$, is a *lower* bound for $\left\|f''\right\|_{[\alpha,\beta]}$, which by itself is not helpful. But $2|D(f,\alpha,\beta)|$ is an *upper* bound for $\left\|f''\right\|_{-\infty,[\alpha,\beta]}$. The cone of interesting functions, $\mathcal{C}$, will contain those f for which $\left\|f''\right\|_{[\alpha,\beta]}$ is not drastically greater than the maximum of $\left\|f''\right\|_{-\infty,[\beta-h_-,\alpha]}$ and $\left\|f''\right\|_{-\infty,[\beta,\alpha+h_+]}$, where $h_\pm > \beta-\alpha$.

The cone $\mathcal{C}$ is defined in terms of two numbers: an integer $n_{\text{ninit}} \ge 5$ and a number $\mathfrak{C}_0 \ge 1$. Let

$$\mathfrak{h} := \frac{3(b-a)}{n_{\text{ninit}}-1}, \qquad \mathfrak{C}(h) := \frac{\mathfrak{C}_0\mathfrak{h}}{\mathfrak{h}-h} \quad \text{for } 0 < h < \mathfrak{h}. \tag{5}$$

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

$$B(f'',\alpha,\beta,h_-,h_+) := \begin{cases} \max\left(\mathfrak{C}(h_-)\left\|f''\right\|_{-\infty,[\beta-h_-,\alpha]}, \mathfrak{C}(h_+)\left\|f''\right\|_{-\infty,[\beta,\alpha+h_+]}\right), \\ \qquad a \le \beta-h_- < \alpha+h_+ \le b, \\ \mathfrak{C}(h_-)\left\|f''\right\|_{-\infty,[\beta-h_-,\alpha]}, \quad a \le \beta-h_- < b < \alpha+h_+, \\ \mathfrak{C}(h_+)\left\|f''\right\|_{-\infty,[\beta,\alpha+h_+]}, \quad \beta-h_- < a < \alpha+h_+ \le b. \end{cases} \tag{6}$$

$$\begin{aligned}\mathcal{C} := \Big\{ & f \in W^{2,\infty} : \left\|f''\right\|_{[\alpha,\beta]} \le B(f'',\alpha,\beta,h_-,h_+) \text{ for all } [\alpha,\beta] \subset [a,b] \\ & \text{and } h_\pm \in (\beta-\alpha,\mathfrak{h})\Big\}.\end{aligned} \tag{7}$$

The set $\mathcal{C}$ is a cone because $f \in \mathcal{C} \Longrightarrow cf \in \mathcal{C}$ for all real c. The integer n_{ninit} is the initial number of subintervals in Algorithms A and M. The parameter $\mathfrak{C}_0$ is some number no less than one for which

$$\lim_{h\to0}\left\|f''\right\|_{[x-h,x+h]} \le \mathfrak{C}_0 \lim_{h\to0}\left\|f''\right\|_{-\infty,[x-h,x+h]}, \quad \forall x \in (a,b),\ f \in \mathcal{C}.$$

Increasing either n_{ninit} or $\mathfrak{C}_0$ expands the cone to include more functions.

Fig. 1 depicts the second derivative of a typical function in $W^{2,\infty}$. In this figure $\left\|f''\right\|_{-\infty,[\beta-h_-,\alpha]} = \left|f''(\beta-h_-)\right| = 0$, which means that the behavior of f'' to the left of $[\alpha,\beta]$ cannot help provide an upper bound on $\left\|f''\right\|_{[\alpha,\beta]}$. However, $\left\|f''\right\|_{-\infty,[\beta,\alpha+h_+]} = \left|f''(\alpha+h_+)\right| > 0$, so this f may lie in the cone $\mathcal{C}$ provided that $\mathfrak{C}(h_+)$ is large enough. The possibility of points in $[a,b]$ where f'' vanishes motivates the definition of $B(f'',\alpha,\beta,h_-,h_+)$ to depend on the behavior of f'' to both the left and right of $[\alpha,\beta]$. One may note that if f'' vanishes at two points that are close to each other or at a point that is close to either a or b, then f will lie outside $\mathcal{C}$. The definition of "close" depends on $(b-a)/n_{\text{ninit}}$.

We give an example of a family of functions whose members lie inside $\mathcal{C}$ if they are not too spiky. Consider the following hump-shaped function defined on $[-1,1]$, whose second derivative has jump

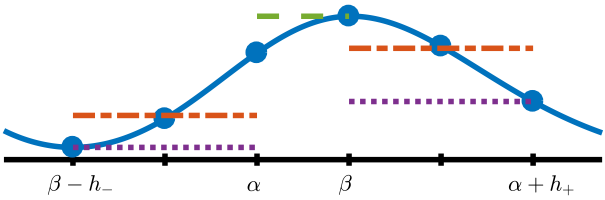

Fig. 1. For some sample f, a plot of $|f''(x)|$ (solid), $\|f''\|_{[\alpha,\beta]}$ (dashed), $\|f''\|_{-\infty,[\beta-h_-,\alpha]}$ and $\|f''\|_{-\infty,[\beta,\alpha+h_+]}$ (dotted), and $2\,|D(f,\beta-h_-,\alpha)|$ and $2\,|D(f,\beta,\alpha+h_+)|$ (dot-dashed). All figures in this paper are reproducible by `LocallyAdaptivePaperFigs.m` in GAIL [4].

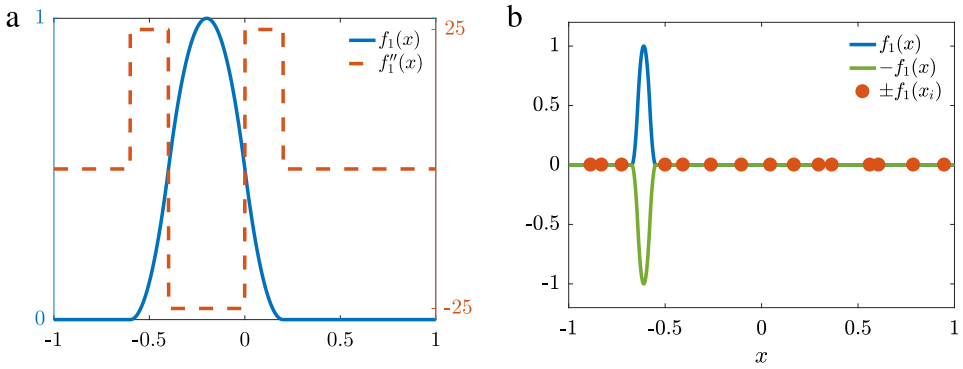

Fig. 2. (a) The example f_1 with $-c = \delta = 0.2$ and its piecewise constant second derivative. (b) The fooling functions $\pm f_1$ used to prove (19) (with different choices of c and δ). The case $n = 15$ is shown.

discontinuities:

$$f_1(x) = \begin{cases} \dfrac{1}{2\delta^2}\Big[4\delta^2 + (x-c)^2 + (x-c-\delta)\,|x-c-\delta| \\ \quad -(x-c+\delta)\,|x-c+\delta|\Big], & |x-c| \le 2\delta, \\ 0, & \text{otherwise}, \end{cases} \tag{8}$$

$$f_1''(x) = \begin{cases} \dfrac{1}{\delta^2}[1 + \operatorname{sign}(x-c-\delta) - \operatorname{sign}(x-c+\delta)], & |x-c| \le 2\delta, \\ 0, & \text{otherwise}. \end{cases}$$

Here c and δ are parameters satisfying $-1 \le c - 2\delta < c + 2\delta \le 1$. This function and its second derivative are shown in Fig. 2(a) for $-c = \delta = 0.2$.

If the hump is wide enough, i.e., $\delta \ge 2\mathfrak{h}$, then $f_1 \in \mathcal{C}$ for any choice of $\mathfrak{C}_0 \ge 1$. For any $[\alpha,\beta] \subseteq [-1,1]$ and $h_\pm$ satisfying the conditions in the definition of $\mathcal{C}$ in (7), it follows that

$$\|f_1''\|_{[\alpha,\beta]} = \frac{1}{\delta^2} = \begin{cases} \|f_1''\|_{-\infty,[\beta,\alpha+h_+]} & \text{if } \alpha \text{ or } \beta \in [c-2\delta, c-1.5\delta] \\ & \cup [c-\delta, c-0.5\delta] \cup [c+\delta, c+1.5\delta], \\ \|f_1''\|_{-\infty,[\beta-h_-,\alpha]} & \text{if } \alpha \text{ or } \beta \in [c-1.5\delta, c-\delta] \\ & \cup [c-0.5\delta, c+\delta] \cup [c+1.5\delta, c+2\delta]. \end{cases}$$

Thus, $B(f'',\alpha,\beta,h_-,h_+) \ge \|f_1''\|_{[\alpha,\beta]}$ for $\beta \ge c - 2\delta$ or $\alpha \le c + 2\delta$. For $[\alpha,\beta] \subset [-1, c-2\delta) \cup (c+2\delta, 1]$, it follows that $\|f_1''\|_{[\alpha,\beta]} = 0$, so $B(f'',\alpha,\beta,h_-,h_+) \ge \|f_1''\|_{[\alpha,\beta]}$ automatically. Thus, the definition of the cone is satisfied.

However, if the hump is too narrow, i.e., $\delta < 2\mathfrak{h}$, the function f_1 is too spiky to lie in the cone $\mathcal{C}$ regardless of how $\mathfrak{C}_0$ is defined. For α, β, and h satisfying

$$0 < c - 1.5\delta - \alpha = \beta - c + 1.5\delta < 0.5\delta < c - 1.5\delta - \beta + h < \mathfrak{h},$$

it follows that

$$\beta - h < c - 2\delta < \alpha < c - 1.5\alpha < \beta < c - \delta < \alpha + h,$$
$$\left\|f_1''\right\|_{-\infty,[\beta-h,\alpha]} = \left\|f_1''\right\|_{-\infty,[\beta,\alpha+h]} = 0 < \delta^{-2} = \left\|f_1''\right\|_{[\alpha,\beta]}.$$

This violates the definition of $\mathcal{C}$. This example illustrates how the choice of n_{ninit}, or equivalently $\mathfrak{h}$, influences the width of a spiky function and determines whether it lies in $\mathcal{C}$.

3. The function approximation algorithm, *A*

3.1. Algorithm A

The idea of Algorithm A is to use divided differences to provide upper bounds on $\|f\|_{-\infty,[\beta-h_-,\alpha]}$ and $\|f\|_{-\infty[\beta,\alpha+h_+]}$ via (4), which then provide an upper bound on $\|f\|_{[\alpha,\beta]}$ via the definition of the cone, $\mathcal{C}$, in (7). This in turn yields an upper bound on the spline error via (2). After stating the algorithm, its effectiveness is proven.

Algorithm A. For some finite interval $[a, b]$, integer $n_{\text{ninit}} \geq 5$, and constant $\mathfrak{C}_0 \geq 1$, let $\mathfrak{h}$ and $\mathfrak{C}(h)$ be defined as in (5). Let $f : [a, b] \to \mathbb{R}$ and $\varepsilon > 0$ be user inputs. Define the number of subintervals, $n = n_{\text{ninit}}$, and the iteration number, $l = 0$. Define the initial partition of equally spaced points, $x_{0:n}$, and an index set of subintervals:

$$h_0 = \frac{b-a}{n}, \qquad x_i = a + ih_0, \quad i \in 0:n, \qquad \mathcal{I} = 1:(n-1).$$

Step 1. *Check for convergence*. For all $i \in \mathcal{I}$ compute

$$\overline{\text{err}}_i = \frac{1}{8}\mathfrak{C}(3h_l)\left|f(x_{i+1}) - 2f(x_i) + f(x_{i-1})\right|. \tag{9}$$

Let $\widetilde{\mathcal{I}} = \{i \in \mathcal{I} : \overline{\text{err}}_i > \varepsilon\}$ be the index set for those $\overline{\text{err}}_i$ that are too large. If $\widetilde{\mathcal{I}} = \emptyset$, return the linear spline $A(f, \varepsilon) = S(f, x_{0:n})$ and terminate the algorithm. Otherwise, continue to the next step.

Step 2. *Split the subintervals as needed*. Update the present partition, $x_{0:n}$, to include the subinterval midpoints

$$\frac{x_{i-2}+x_{i-1}}{2}, \qquad \frac{x_{i-1}+x_i}{2}, \qquad \frac{x_i+x_{i+1}}{2}, \qquad \frac{x_{i+1}+x_{i+2}}{2}, \quad i \in \widetilde{\mathcal{I}}.$$

(The leftmost midpoint is only needed for $i \geq 2$, and the rightmost midpoint is only needed for $i \leq n-2$.) Update the set $\mathcal{I}$ to consist of the new indices corresponding to the old points

$$x_{i-1}, \qquad \frac{x_{i-1}+x_i}{2}, \qquad \frac{x_i+x_{i+1}}{2}, \qquad x_{i+1}, \quad i \in \widetilde{\mathcal{I}}.$$

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

Theorem 1. *Algorithm A defined above satisfies (APP) for functions in the cone $\mathcal{C}$ defined in (7).*

Proof. For every iteration l and every $i \in \mathcal{I}$, the definitions in this algorithm imply that $x_i - x_{i-1} = x_{i+1} - x_i = h_l = 2^{-l}h_0$, and

$$\overline{\text{err}}_i = \frac{1}{4}\mathfrak{C}(3h_l)h_l^2\left|D(f, x_{i-1}, x_{i+1})\right| \quad \text{by (3)} \tag{10}$$
$$\geq \frac{1}{8}\mathfrak{C}(3h_l)h_l^2\left\|f''\right\|_{-\infty,[x_{i-1},x_{i+1}]} \quad \text{by (4)}. \tag{11}$$

We show that when all $\overline{\text{err}}_i$ get small enough, Algorithm A terminates successfully.

For all $x \in [a, b]$, let $I_{x,l}$ be the closed interval with width h_l containing x that *might* arise at some stage in Algorithm A as $[x_{i_l-1}, x_{i_l}]$ for some $i_l \in 1 : n$. (The dependence of n on l is suppressed.) Specifically this interval is defined for all $x \in [a, b]$ and $l \in \mathbb{N}_0$ as

$$I_{x,l} := [a + jh_l, a + (j+1)h_l], \quad j = \min\left(\left\lfloor \frac{(x-a)}{h_l} \right\rfloor, 2^l n_{\text{ninit}} - 1\right). \tag{12}$$

Let $\ell(x)$ be defined such that $I_{x,\ell(x)}$ is the final subinterval in Algorithm A that contains x when the algorithm terminates. We need to establish that $\|f - S(f)\|_{I_{x,\ell(x)}} \le \varepsilon$ for every $x \in [a, b]$.

Fix $x \in [a + \mathfrak{h}, b - \mathfrak{h}]$. The proof for $x \in [a, a + \mathfrak{h}) \cup (b - \mathfrak{h}, b]$ is similar. By (11) there exists some $l_- \le \ell(x)$ for which $I_{x,l_-} = [x_{i_{l_-}-1}, x_{i_{l_-}}]$ and

$$\frac{1}{8}\mathfrak{C}(3h_{l_-})h_{l_-}^2 \, \|f\|_{-\infty,[x_{i_{l_-}-3}, x_{i_{l_-}-1}]} \le \overline{\text{err}}_{i_{l_-}-2} \le \varepsilon. \tag{13a}$$

There also exists an $l_+ \le \ell(x)$ such that $I_{x,l_+} = [x_{i_{l_+}-1}, x_{i_{l_+}}]$ and

$$\frac{1}{8}\mathfrak{C}(3h_{l_+})h_{l_+}^2 \, \|f\|_{-\infty,[x_{i_{l_+}}, x_{i_{l_+}+2}]} \le \overline{\text{err}}_{i_{l_+}+1} \le \varepsilon. \tag{13b}$$

Noting that $x_{i_{l_\pm}-1} \le x_{i_{\ell(x)-1}} < x_{i_{\ell(x)}} \le x_{i_{l_\pm}}$, we may conclude that

$$\begin{aligned}
\|f - S(f)\|_{I_{x,\ell(x)}} &\le \frac{1}{8}h_{\ell(x)}^2 \left\|f''\right\|_{I_{x,\ell(x)}} \quad \text{by (2)} \\
&\le \frac{1}{8}h_{\ell(x)}^2 B(f, x_{i_{\ell(x)-1}}, x_{i_{\ell(x)}}, x_{i_{\ell(x)}} - x_{i_{l_-}-3}, x_{i_{l_+}+2} - x_{i_{\ell(x)-1}}) \quad \text{by (7)} \\
&\le \frac{1}{8}h_{\ell(x)}^2 \max\Big(\mathfrak{C}(x_{i_{\ell(x)}} - x_{i_{l_-}-3}) \left\|f''\right\|_{-\infty,[x_{i_{l_-}-3}, x_{i_{\ell(x)-1}}]}, \\
&\quad \mathfrak{C}(x_{i_{l_+}+2} - x_{i_{\ell(x)-1}}) \left\|f''\right\|_{-\infty,[x_{i_{\ell(x)}}, x_{i_{l_+}+2}]}\Big) \quad \text{by the definition of } B \text{ in (6)} \\
&\le \max\Big(\frac{h_{l_-}^2}{8}\mathfrak{C}(3h_{l_-}) \left\|f''\right\|_{-\infty,[x_{i_{l_-}-3}, x_{i_{l_-}-1}]}, \frac{h_{l_+}^2}{8}\mathfrak{C}(3h_{l_+}) \left\|f''\right\|_{-\infty,[x_{i_{l_+}}, x_{i_{l_+}+2}]}\Big) \\
&\quad \text{because } h_{\ell(x)} \le h_{l_\pm} \text{ and } \mathfrak{C} \text{ is non-decreasing} \\
&\le \varepsilon \quad \text{by (13).}
\end{aligned}$$

This concludes the proof. □

Fig. 3(a) displays the function $-f_1$ defined in (8) for a certain choice of parameters, along with the data used to compute the linear spline approximation $A(-f_1, 0.02)$ by the algorithm described above. Note that $-f_1$ is sampled less densely where it is flat.

3.2. The computational cost of A

In this section, we investigate the computational cost of our locally adaptive algorithm. Recall the definitions of h_l, $I_{x,l}$, and $\ell(x)$ from the previous subsection. Let $\bar{I}_{x,l}$ be a similar interval with generally five times the width of $I_{x,l}$:

$$\bar{I}_{x,l} = \left[a + \max(0, j-3)h_l, a + \min(j+2, 2^l n_{\text{ninit}})h_l\right] \supset I_{x,l}, \tag{14}$$

with the same j as in (12) above. Let

$$L(x) = \min\left\{l \in \mathbb{N}_0 : \frac{1}{8}\mathfrak{C}\left(3h_l\right) h_l^2 \left\|f''\right\|_{\bar{I}_{x,l}} \le \varepsilon\right\}. \tag{15}$$

Note that $L(x)$ does depend on f and ε, although this dependence is suppressed in the notation.

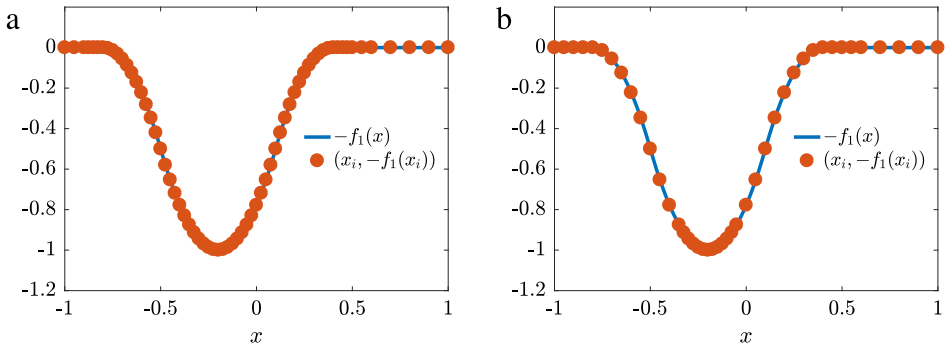

Fig. 3. (a) The nonuniform sampling density of Algorithm A for input function $-f_1$ defined by $\delta = 0.3$ and $c = -0.2$. A total of 3 iterations and 65 points are used to meet the error tolerance of 0.02. We have chosen $n_{\text{ninit}} = 20$ and $\mathfrak{C}_0 = 10$. (b) The same situation as in (a), but now with Algorithm M. Still 3 iterations but only 43 nonuniform sampling points are needed to obtain the minimum of $-f_1$.

We now show that $\ell(x) \le L(x)$. At each iteration of Algorithm A, x lies in $I_{x,l}$ for some l, and by the time Algorithm A terminates, all values of $l = 0, \ldots, \ell(x)$ are realized. If $\ell(x) > L(x)$, then at iteration $L(x)$, the interval $I_{x,L(x)}$ must be split in Step 2 of A. So, $I_{x,L(x)}$ has width $h_{L(x)}$ and corresponds to $[x_{i-1}, x_i]$ for some i. We assume that $i \in 3 : n-2$; the other cases have a similar proof. According to Step 2 of Algorithm A, the only way for $[x_{i-1}, x_i]$ to be split is if $\overline{\text{err}}_{i-2}$, $\overline{\text{err}}_{i-1}$, $\overline{\text{err}}_i$, or $\overline{\text{err}}_{i+1}$ is larger than ε. However, in the proof of Theorem 1 it is noted that for $k \in \{-2, -1, 0, 1\}$,

$$\begin{aligned}
\overline{\text{err}}_{i+k} &= \frac{1}{4}\mathfrak{C}(3h_{L(x)})h_{L(x)}^2\,|D(f, x_{i-1+k}, x_{i+1+k})| \quad \text{by (10)} \\
&\le \frac{1}{8}\mathfrak{C}(3h_{L(x)})h_{L(x)}^2\,\|f''\|_{[x_{i-1+k}, x_{i+1+k}]} \quad \text{by (4)} \\
&\le \frac{1}{8}\mathfrak{C}(3h_{L(x)})h_{L(x)}^2\,\|f''\|_{\bar{I}_{x,L(x)}} \le \varepsilon \quad \text{by (14) and (15).}
\end{aligned} \tag{16}$$

This is a contradiction, so in fact, $\ell(x) \le L(x)$, which is used to prove an upper bound on the computational cost of Algorithm A.

Theorem 2. *Let* $\text{cost}(A, f, \varepsilon)$ *denote the number of functional evaluations required by* $A(f, \varepsilon)$. *This computational cost has the following upper bound:*

$$\text{cost}(A, f, \varepsilon) \le \frac{1}{h_0}\int_a^b 2^{L(x)}\,\mathrm{d}x + 1 = \int_a^b \frac{1}{h_{L(x)}}\,\mathrm{d}x + 1,$$

where $L(x)$ *is defined in* (15).

Proof. Let $x_{0:n}$ be the final partition when $A(f, \varepsilon)$ successfully terminates. Note that $2^{\ell(x)}$ is constant for $x \in I_{x_{i-1},\ell(x_{i-1})} = [x_{i-1}, x_i]$ for $i \in 1 : n$. Furthermore $\int_{x_{i-1}}^{x_i} 2^{\ell(x)}\,\mathrm{d}x = h_0$. Then the number of function values required is

$$n + 1 = 1 + \sum_{i=1}^{n} 1 = 1 + \sum_{i=1}^{n} \frac{1}{h_0}\int_{x_{i-1}}^{x_i} 2^{\ell(x)}\,\mathrm{d}x = 1 + \frac{1}{h_0}\int_a^b 2^{\ell(x)}\,\mathrm{d}x.$$

Noting that $\ell(x) \le L(x)$ establishes the formula for $\text{cost}(A, f, \varepsilon)$. □

From the definition of $L(x)$ in (15), we know that

$$\frac{1}{h_{L(x)}} = \frac{2}{h_{L(x)-1}} < 2\sqrt{\frac{\mathfrak{C}\left(3h_{L(x)-1}\right)\|f''\|_{\bar{I}_{x,L(x)-1}}}{8\varepsilon}} = \sqrt{\frac{\mathfrak{C}\left(6h_{L(x)}\right)\|f''\|_{\bar{I}_{x,L(x)-1}}}{2\varepsilon}}.$$

As $\varepsilon \to 0$, $L(x) \to \infty$, $h_{L(x)} \to 0$, and $\left\|f''\right\|_{\bar{I}_{x,L(x)-1}}$ approaches $|f''(x)|$. Thus, the small ε asymptotic upper bound on computational cost is

$$\begin{aligned}\operatorname{cost}(A,f,\varepsilon) &\lesssim \int_a^b \sqrt{\frac{\mathfrak{C}\,(0)\,|f''(x)|}{2\varepsilon}}\,\mathrm{d}x + 1 = \sqrt{\frac{\mathfrak{C}_0\,\|f''\|_{\frac{1}{2}}}{2\varepsilon}} + 1\\ &\le (b-a)\sqrt{\frac{\mathfrak{C}_0\,\|f''\|}{2\varepsilon}} + 1 \quad \text{by (17a) below.}\end{aligned}$$

For functions in the cone $\mathcal{C}$, the (quasi-)seminorms $\left\|f''\right\|$ and $\left\|f''\right\|_{\frac{1}{2}}$ are equivalent, but for functions in $W^{2,\infty}$ they are not, as shown in the following proposition.

Proposition 3. *The quantities $\left\|f''\right\|$ and $\left\|f''\right\|_{\frac{1}{2}}$ bound each other as follows:*

$$(b-a)^2\left\|f''\right\|_{-\infty,[\alpha,\beta]} \le \left\|f''\right\|_{\frac{1}{2},[\alpha,\beta]} \le (b-a)^2\left\|f''\right\|_{[\alpha,\beta]} \quad \forall f \in W^{2,\infty}, \tag{17a}$$

$$\frac{4\mathfrak{h}^2}{27\mathfrak{C}_0}\left\|f''\right\| \le \left\|f''\right\|_{\frac{1}{2}} \quad \forall f \in \mathcal{C}, \tag{17b}$$

$$\sup_{f\in W^{2,\infty}:\left\|f''\right\|_{\frac{1}{2}}\le 1} \left\|f''\right\| = \infty. \tag{17c}$$

Proof. The first inequality follows from the definitions of the (quasi-)norms:

$$\begin{aligned}(\beta-\alpha)^2\left\|f''\right\|_{-\infty,[\alpha,\beta]} &= \left\{\sqrt{\|f''\|_{-\infty,[\alpha,\beta]}}\int_\alpha^\beta \mathrm{d}x\right\}^2 \le \left\{\int_\alpha^\beta \sqrt{|f''(x)|}\,\mathrm{d}x\right\}^2\\ = \left\|f''\right\|_{\frac{1}{2},[\alpha,\beta]} &\le \left\{\sqrt{\|f''\|_{[\alpha,\beta]}}\int_\alpha^\beta \mathrm{d}x\right\}^2 \le (\beta-\alpha)^2\left\|f''\right\|_{[\alpha,\beta]}.\end{aligned} \tag{18}$$

The second inequality comes from the cone definition. Since $\left\|f''\right\| = \left\|f''\right\|_{[\alpha,\beta]}$ for some interval $[\alpha,\beta]$ whose width can be made arbitrarily small, we have

$$\begin{aligned}\left\|f''\right\|_{[\alpha,\beta]} &\le \inf\left\{B(f,\alpha,\beta,h,h) : h \in (\beta-\alpha,\mathfrak{h})\right\} \quad \text{by (7)}\\ &\le \inf\left\{\mathfrak{C}(h)\max\left(\left\|f''\right\|_{-\infty,[\beta-h,\alpha]}, \left\|f''\right\|_{-\infty,[\beta,\alpha+h]}\right) : h \in (\beta-\alpha,\mathfrak{h})\right\}\\ &\le \inf_{\beta-\alpha<h<\mathfrak{h}} \frac{\mathfrak{C}(h)}{(h-\beta+\alpha)^2}\left\|f''\right\|_{\frac{1}{2}}\\ &\quad \text{by (18), and since } \left\|f''\right\|_{\frac{1}{2},[\alpha,\beta]} \le \left\|f''\right\|_{\frac{1}{2}}\ \forall[\alpha,\beta]\subseteq[a,b]\\ &\le \inf_{0<h<\mathfrak{h}} \frac{\mathfrak{C}(h)}{h^2}\left\|f''\right\|_{\frac{1}{2}} \quad \text{since } \beta-\alpha \text{ may be made arbitrarily small}\\ &= \frac{27\mathfrak{C}_0}{4\mathfrak{h}^2}\left\|f''\right\|_{\frac{1}{2}} \quad \text{by (5).}\end{aligned}$$

When $\mathfrak{h}$ is small, it is possible for $\left\|f''\right\|_{\frac{1}{2}}$ to be quite small in comparison to $\left\|f''\right\|$. This occurs when f'' is rather spiky.

The hump function f_1 in (8) satisfies $\left\|f_1''\right\| / \left\|f_1''\right\|_{\frac{1}{2}} = \delta^{-2}/16$. By making δ small enough, we may make this ratio arbitrarily large, thus proving (17c). However, since $f_1 \notin \mathcal{C}$ for $\delta < 2\mathfrak{h}$, this does not violate (17b). □

3.3. Lower complexity bound

The upper bound on the computational cost of Algorithm A provides an upper bound on the complexity of problem (APP). We now construct lower bounds on the complexity of the problem, i.e., the computational cost of the best algorithm. We then observe that these lower bounds have the same asymptotic behavior as the computational cost of Algorithm A. Our lower complexity bounds are derived for subsets of functions in the balls, $\mathcal{B}_\sigma^{2,p} = \{f \in W^{1,\infty} : \|f''\|_p \le \sigma\}$, for $p = 1/2, \infty$.

Theorem 4. *Let σ be any positive number, and $\mathcal{C}$ be defined as in (7).*

i. *If A^* solves (APP) for all $f \in \mathcal{B}_\sigma^{2,\frac{1}{2}}$ and all $0 < \varepsilon < \sigma/16$, then*

$$\mathrm{cost}(A^*, f, \varepsilon) = \infty. \tag{19a}$$

ii. *If A^* solves (APP) for all $f \in \mathcal{B}_\sigma^{2,\infty}$ and all $\varepsilon > 0$, then*

$$\mathrm{cost}(A^*, f, \varepsilon) \ge \frac{(b-a)}{4}\sqrt{\frac{\sigma}{\varepsilon}} - 1. \tag{19b}$$

iii. *If A^* satisfies (APP) for all $f \in \mathcal{C} \cap \mathcal{B}_\sigma^{2,\frac{1}{2}}$ and all $\varepsilon > 0$, then*

$$\mathrm{cost}(A^*, f, \varepsilon) \ge \sqrt{\frac{(\mathfrak{C}_0 - 1)\sigma}{16(\mathfrak{C}_0 + 1)\varepsilon}} - 1. \tag{20a}$$

iv. *If A^* satisfies (APP) for all $f \in \mathcal{C} \cap \mathcal{B}_\sigma^{2,\infty}$ and all $\varepsilon > 0$, then*

$$\mathrm{cost}(A^*, f, \varepsilon) \ge (b-a)\sqrt{\frac{(\mathfrak{C}_0 - 1)\sigma}{16(\mathfrak{C}_0 + 1)\varepsilon}} - 1. \tag{20b}$$

Note by comparing (19a) and (20a) that the lower complexity bound is significantly altered by restricting the set of input functions from the whole ball of $\mathcal{B}_\sigma^{2,\frac{1}{2}}$ to the intersection of that ball with the cone $\mathcal{C}$. Also note that the lower bounds above assume that the radius of the ball, σ, is known a priori, whereas for our Algorithm A, *no* bound on a norm of f'' is provided as input. However, the computational cost of Algorithm A is asymptotically the same as the computational cost of the best possible algorithm, A^* in (20), as $\varepsilon \to 0$.

Proof. The lower bounds are proved by constructing fooling functions for which Algorithm A succeeds, and then showing that at least a certain number of samples must be used. The proofs of (19) are simpler, so we start with them.

Let A^* be a successful algorithm for all $f \in W^{2,\infty}$, and consider the partition $x_{0:n+1}$, where $x_{1:n}$ are the data sites used to compute $A^*(0, \varepsilon)$. We now allow the possibility of $a = x_0 = x_1$ and $x_n = x_{n+1} = b$. Choose any $j = 1, \ldots, n+1$ with $x_j - x_{j-1} \ge (b-a)/(n+1)$. Let f_1 be defined as in (8) with $c = (x_j + x_{j-1})/2$, and $\delta = (b-a)/[4(n+1)]$.

For any real γ, it follows that $\gamma f_1(x_i) = 0$ for $i = 0, \ldots, n+1$. Fig. 2(b) illustrates this situation. Since 0 and $\pm\gamma f_1$ share the same values at the data sites, then they must share the same approximation: $A^*(\pm\gamma f_1, \varepsilon) = A^*(0, \varepsilon)$. Moreover, $\mathrm{cost}(A^*, 0, \varepsilon) = \mathrm{cost}(A^*, \pm\gamma f_1, \varepsilon) = n$. Since the approximations of 0, $-\gamma f_1$, and γf_1 are identical, this implies that γ must be no greater than ε:

$$\begin{aligned}
\varepsilon &\ge \max\left(\left\|\gamma f_1 - A^*(\gamma f_1, \varepsilon)\right\|, \left\|-\gamma f_1 - A^*(-\gamma f_1, \varepsilon)\right\|\right) \\
&= \max\left(\left\|\gamma f_1 - A^*(0, \varepsilon)\right\|, \left\|-\gamma f_1 - A^*(0, \varepsilon)\right\|\right) \\
&\ge \frac{1}{2}\left[\left\|\gamma f_1 - A^*(0, \varepsilon)\right\| + \left\|-\gamma f_1 - A^*(0, \varepsilon)\right\|\right] \\
&\ge \frac{1}{2}\left\|\gamma f_1 - (-\gamma f_1)\right\| = \left\|\gamma f_1\right\| = \gamma = \begin{cases} \left\|\gamma f_1''\right\|_{\frac{1}{2}}/16, \\ \delta^2\left\|\gamma f_1''\right\| = \dfrac{(b-a)^2\left\|\gamma f_1''\right\|}{16(n+1)^2}, \end{cases}
\end{aligned}$$

since $\|f_1''\| = \delta^{-2}$, and $\|f_1''\|_{\frac{1}{2}} = 16$. The top inequality cannot be satisfied unless $\sigma = \|\gamma f_1''\|_{\frac{1}{2}}$ is small enough, which establishes (19a). Solving the bottom inequality for n in terms of $\sigma = \|\gamma f_1''\|$ establishes (19b).

Now, we prove the lower complexity bounds (20), assuming that A^* is a successful algorithm for all $f \in \mathcal{C}$. Let f_0 be defined as follows

$$f_0(x) = \frac{x^2}{2}, \qquad f_0''(x) = 1, \quad x \in [a,b]; \qquad \|f_0''\|_{\frac{1}{2}} = (b-a)^2, \qquad \|f_0''\| = 1.$$

Since f_0'' is constant, it follows that $f_0 \in \mathcal{C}$, and A^* successfully approximates γf_0 for any $\gamma \geq 0$.

Consider the partition $x_{0:n+1}$, where $x_{1:n}$ are the data sites used to compute $A^*(\gamma f_0, \varepsilon)$, and we again allow the possibility of $a = x_0 = x_1$ and $x_n = x_{n+1} = b$. Again choose any $j = 1, \ldots, n+1$ with $x_j - x_{j-1} \geq (b-a)/(n+1)$, and let f_1 be defined as in (8) with $c = (x_j + x_{j-1})/2$, and $\delta = (b-a)/[4(n+1)]$. We construct two fooling functions:

$$f_\pm = f_0 \pm \widetilde{\gamma} f_1, \qquad \widetilde{\gamma} = \frac{\mathfrak{C}_0 - 1}{\mathfrak{C}_0 + 1}\delta^2, \qquad \|f_\pm''\| = 1 + \frac{\widetilde{\gamma}}{\delta^2} = \frac{2\mathfrak{C}_0}{\mathfrak{C}_0 + 1},$$

$$\|f_\pm''\|_{-\infty,[\alpha,\beta]} \geq 1 - \frac{\widetilde{\gamma}}{\delta^2} = \frac{2}{\mathfrak{C}_0 + 1} = \frac{\|f_\pm''\|}{\mathfrak{C}_0} \quad \forall [\alpha,\beta] \subseteq [a,b].$$

The above calculations show that $\gamma f_\pm \in \mathcal{C}$ for all real γ. Moreover, the definition of $f_\pm$ ensures that $A^*(\gamma f_0) = A^*(\gamma f_\pm)$, and $\operatorname{cost}(A^*, \gamma f_0) = \operatorname{cost}(A^*, \gamma f_\pm) = n$.

Analogously to the argument above, we show that $\gamma\widetilde{\gamma}$ must be no larger than ε:

$$\begin{aligned}
\varepsilon &\geq \max\left(\|\gamma f_+ - A(\gamma f_+, \varepsilon)\|, \|\gamma f_- - A(\gamma f_-, \varepsilon)\|\right) \\
&\geq \frac{1}{2}\left[\|\gamma f_+ - A(\gamma f_+, \varepsilon)\| + \|\gamma f_- - A(\gamma f_-, \varepsilon)\|\right] \\
&= \frac{1}{2}\left[\|\gamma f_+ - A(\gamma f_0, \varepsilon)\| + \|\gamma f_- - A(\gamma f_0, \varepsilon)\|\right] \\
&\geq \frac{1}{2}\|\gamma f_+ - \gamma f_-\| = \|\gamma\widetilde{\gamma} f_1\| = \gamma\widetilde{\gamma} \\
&= \left\{\begin{array}{c} \|\gamma f_0''\|_{\frac{1}{2}} \\ \hline (b-a)^2 \\ \hline \|\gamma f_0''\| \end{array}\right\} \cdot \frac{\mathfrak{C}_0 - 1}{\mathfrak{C}_0 + 1}\delta^2 = \left\{\begin{array}{c} \|\gamma f_0''\|_{\frac{1}{2}} \\ \hline (b-a)^2 \|\gamma f_0''\| \end{array}\right\} \cdot \frac{\mathfrak{C}_0 - 1}{16(\mathfrak{C}_0 + 1)(n+1)^2}.
\end{aligned}$$

Substituting $\|\gamma f_0''\|_{\frac{1}{2}} = \sigma$ in the top inequality and $\|\gamma f_0''\| = \sigma$ in the bottom inequality, and then solving for n yield the two bounds in (20). □

4. The minimization algorithm, *M*

4.1. *Algorithm M*

Our minimization Algorithm M relies on the derivations in the previous sections. The main departure from Algorithm A is the stopping criterion. It is unnecessary to approximate f accurately everywhere, only where f is small.

Algorithm M. For some finite interval $[a,b]$, integer $n_{\text{ninit}} \geq 5$, and constant $\mathfrak{C}_0 \geq 1$, let $\mathfrak{h}$ and $\mathfrak{C}(h)$ be defined as in (5). 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:

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

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

Step 1. *Check for convergence*. Compute $\overline{\mathrm{err}}_i$ for all $i \in \mathcal{I}_\pm$ according to (9). Let $\widetilde{\mathcal{I}}_s = \{i \in \mathcal{I}_s : \overline{\mathrm{err}}_i > \varepsilon\}$. Next compute

$$\widehat{\mathrm{err}}_{i,s} := \overline{\mathrm{err}}_i + \widehat{M} - \min\bigl(f(x_{i-s2}), f(x_{i-s1})\bigr) \quad \forall i \in \widetilde{\mathcal{I}}_s,$$
$$\widehat{\mathcal{I}}_s = \Bigl\{i \in \widetilde{\mathcal{I}}_s : \widehat{\mathrm{err}}_{i,s} > \varepsilon \text{ or } \bigl(i - s3 \in \widetilde{\mathcal{I}}_{-s} \,\&\, \widehat{\mathrm{err}}_{i-s3,-s} > \varepsilon\bigr)\Bigr\}.$$

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.

Step 2. *Split the subintervals as needed*. Update the present partition, $x_{0:n}$, to include the subinterval midpoints

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

(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

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

(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.

Theorem 5. *Algorithm* M *defined above satisfies* (MIN) *for functions in the cone* $\mathcal{C}$ *defined in* (7).

Proof. The proof of success of Algorithm M is similar to that for Algorithm A. Here we give the highlights. We use the notation of $I_{x,l}$ introduced in (12) and analogously define $\tilde{\ell}(x)$ such that $I_{x,\tilde{\ell}(x)}$ is the final subinterval in Algorithm M containing x when the algorithm terminates. For a fixed $x \in [a, b]$ we argue as in the proof of Theorem 1 that there exist $l_\pm \leq l_* \leq \ell(x)$ such that $I_{x,l_*} = [x_{i_{l_*}-1}, x_{i_{l_*}}]$, $x_{i_{l_\pm-1}} \leq x_{i_{l_*}-1} \leq x_{i_{l_*}} \leq x_{i_{l_\pm}}$, and

$$\frac{1}{8}\mathfrak{C}(3h_{l_-})h_{l_-}^2 \,\|f\|_{-\infty,[x_{i_{l_--3}},x_{i_{l_--1}}]} + \widehat{M}_{l_*} - \min\bigl(f(x_{i_{l_*}-1}), f(x_{i_{l_*}})\bigr) \leq \varepsilon,$$
$$\frac{1}{8}\mathfrak{C}(3h_{l_+})h_{l_+}^2 \,\|f\|_{-\infty,[x_{i_{l_+}},x_{i_{l_+}+2}]} + \widehat{M}_{l_*} - \min\bigl(f(x_{i_{l_*}-1}), f(x_{i_{l_*}})\bigr) \leq \varepsilon,$$

where $\widehat{M}_l$ denotes the value of $\widehat{M}$ at iteration $l \in \mathbb{N}_0$. By the definition of $\mathcal{C}$ in (7), this then implies that

$$\widehat{M}_{l_*} - \min_{x_{i_{l_*}-1} \leq x \leq x_{i_{l_*}}} f(x) \leq \widehat{M}_{l_*} - \min(f(x_{i-1}), f(x_i)) + \frac{1}{8}h_{l_*}^2 \,\|f\|_{[x_{i_{l_*}-1},x_{i_{l_*}}]} \leq \varepsilon. \tag{21}$$

Further iterations of the algorithm can only make $\widehat{M}_l$ possibly closer to the solution, $\min_{a\leq x\leq b} f(x)$. □

Fig. 3(b) displays the same function $-f_1$ as in Fig. 3(a), but this time with the sampling points used for minimization. Here $M(-f_1, 0.02)$ uses only 43 points, whereas $A(-f_1, 0.02)$ uses 65 points. This is because $-f_1$ does not need to be approximated accurately when its value is far from the minimum.

4.2. The computational cost of M

The derivation of an upper bound on the cost of Algorithm M proceeds in a similar manner as that for Algorithm A. There are essentially two reasons that a subinterval $[x_{i-1}, x_i]$ need not be split further. The first reason is the same as that for Algorithm A: the function being minimized is approximated on $[x_{i-1}, x_i]$ with an error no more than the tolerance ε. This is reflected in the definition of $\widetilde{\mathcal{I}}_\pm$ in Step 1 of Algorithm M. The second reason is that, although the spline approximation error on $[x_{i-1}, x_i]$ is

larger than ε, the function values on that subinterval are significantly larger than the minimum of the function over $[a, b]$. This is reflected in the definition of $\widehat{\mathcal{I}}_{\pm}$ in Step 1 of Algorithm M.

Our definition of $\widetilde{L}(x)$ reflects these two reasons. Let x_* be some place where the minimum of f is obtained, i.e., $f(x_*) = \min_{a\le x\le b} f(x)$. Let

$$\widetilde{L}(x) = \min\bigl(L(x), \widehat{L}(x)\bigr), \quad x \in [a, b], \tag{22}$$

where $L(x)$ is defined above in (15),

$$\widehat{L}(x) = \min\Biggl\{l \in \mathbb{N}_0 : \biggl\{\biggl[\frac{1}{8}\mathfrak{C}(3h_l) + 2\biggr]\bigl\|f''\bigr\|_{\tilde{I}_{x,l}} + \frac{1}{8}\bigl\|f''\bigr\|_{I_{x_*,l}}\biggr\}h_l^2 + 2\bigl|f'(x)\bigr|h_l + [f(x_*) - f(x)] \le 0\Biggr\}, \tag{23}$$

and $\tilde{I}_{x,l}$ is similar to $I_{x,l}$, but with generally seven times the width:

$$\tilde{I}_{x,l} = \bigl[a + \max(0, j-4)h_l,\, a + \min(j+3, 2^l n_{\text{ninit}})h_l\bigr] \supset I_{x,l},$$

with the same j as in (12) above.

Note that $\widehat{L}(x)$ does not depend on ε, whereas $L(x)$ does. As is the case with $L(x)$, both $\widehat{L}(x)$ and $\widetilde{L}(x)$ depend on f, although this dependence is suppressed in the notation.

Theorem 6. *Denote by* $\text{cost}(M, f, \varepsilon)$ *the number of functional evaluations required by* $M(f, \varepsilon)$. *This computational cost is bounded as follows:*

$$\text{cost}(M, f, \varepsilon) \le \frac{1}{h_0}\int_a^b 2^{\widetilde{L}(x)}\,\mathrm{d}x + 1,$$

where $\widetilde{L}(x)$ *is defined in* (22).

Proof. Using the same argument as for Theorem 2, we only need to show that $\tilde{\ell}(x) \le \widetilde{L}(x)$ for all $x \in [a, b]$. At each iteration of Algorithm M, the index sets $\mathcal{I}_{\pm}$ are both subsets of $\mathcal{I}$ for the corresponding iteration of Algorithm A. Thus $\tilde{\ell}(x) \le L(x)$ by the same argument as used to prove Theorem 2. We only need to show that $\tilde{\ell}(x) \le \widehat{L}(x)$.

We will show that $\widehat{L}(x) < \tilde{\ell}(x) \le L(x)$ for any fixed x leads to a contradiction. If $\widehat{L}(x) < \tilde{\ell}(x)$, then at the $\widehat{L}(x)$th iteration, $I_{x,\widehat{L}(x)} = [x_{i-1}, x_i]$ for some i must be split in Step 2 of M, where $x_i - x_{i-1} = h_{\widehat{L}(x)} = h_0 2^{-\widehat{L}(x)}$. This means that one or more of the following must exceed ε:

$$\widehat{\overline{\text{err}}}_{i+2,+}, \quad \widehat{\overline{\text{err}}}_{i+1,+}, \quad \widehat{\overline{\text{err}}}_{i,+}, \quad \widehat{\overline{\text{err}}}_{i-1,-}, \quad \widehat{\overline{\text{err}}}_{i-2,-}, \quad \widehat{\overline{\text{err}}}_{i-3,-}.$$

We prove that $\widehat{\overline{\text{err}}}_{i+2,+} > \varepsilon$ is impossible. The arguments for the other cases are similar.

If $[x_{i-1}, x_i]$ must be split because $\widehat{\overline{\text{err}}}_{i+2,+} > \varepsilon$, then it is also the case that $i - 1 \in \widehat{\mathcal{I}}_-$, and so $\overline{\text{err}}_{i-1} > \varepsilon$. In this case

$$x_j - x_{j-1} = h_{\widehat{L}(x)} \quad \text{for } j = (i-1) : (i+3).$$

This means that $[x_{i-2}, x_{i+3}] \in \tilde{I}_{x,l}$. By the same argument used in (16) it can be shown that

$$\overline{\text{err}}_{i+2} \le \frac{1}{8}\mathfrak{C}(3h_{\widehat{L}(x)})h_{\widehat{L}(x)}^2\bigl\|f''\bigr\|_{\tilde{I}_{x,\widehat{L}(x)}}. \tag{24}$$

The quantity $\overline{\text{err}}_{i+2}$ is the first term in the definition of $\widehat{\overline{\text{err}}}_{i+2,+}$ in Step 1 of Algorithm M.

Next, we bound $\min\bigl(f(x_i), f(x_{i+1})\bigr)$, which also appears in the definition of $\widehat{\overline{\text{err}}}_{i+2,+}$. As was argued earlier, $[x_{i-1}, x_{i+1}] \in \tilde{I}_{x,\widehat{L}(x)}$. Then a Taylor expansion about the arbitrary $x \in [x_{i-1}, x_i]$ under consideration establishes that

$$\min\bigl(f(x_i), f(x_{i+1})\bigr) \ge f(x) - 2h_{\widehat{L}(x)}\bigl|f'(x)\bigr| - h_{\widehat{L}(x)}^2\bigl\|f''\bigr\|_{\tilde{I}_{x,\widehat{L}(x)}} \tag{25}$$

since $|x_i - x| \le |x_{i+1} - x| \le 2h_{\widehat{L}(x)}$.

Finally, we bound $\widehat{M}_{\widehat{L}(x)}$. Let x_* be a point where f attains its minimum, and let $I_{x_*,l_*} = [x_{i_*-1}, x_{i_*}]$ be the subinterval in the present partition containing x_*, where $l_* \le \widehat{L}(x)$. By (2) it follows that

$$f(x_*) \ge \min(f(x_{i_*-1}), f(x_{i_*})) - \frac{1}{8} h_{l_*}^2 \left\| f'' \right\|_{I_{x_*,l_*}}. \tag{26}$$

There are two possibilities regarding l_*. If $l_* < \widehat{L}(x)$, then by the argument in (21) used to prove Theorem 5,

$$\begin{aligned} \widehat{M}_{\widehat{L}(x)} &\le \widehat{M}_{l_*} \le \min(f(x_{i_*-1}), f(x_{i_*})) - \frac{1}{8} h_{l_*}^2 \left\| f'' \right\|_{[x_{i_*-1}, x_{i_*}]} + \varepsilon \\ &\le f(x_*) + \varepsilon \quad \text{by (26).} \end{aligned}$$

Otherwise, if $l_* = \widehat{L}(x)$, then

$$\widehat{M}_{\widehat{L}(x)} \le \min(f(x_{i_*-1}), f(x_{i_*})) \le f(x_*) + \frac{1}{8} h_{\widehat{L}(x)}^2 \left\| f'' \right\|_{I_{x_*,\widehat{L}(x)}} \quad \text{by (26).}$$

Thus, in either case we have

$$\widehat{M}_{\widehat{L}(x)} \le f(x_*) + \frac{1}{8} h_{\widehat{L}(x)}^2 \left\| f'' \right\|_{I_{x_*,\widehat{L}(x)}} + \varepsilon. \tag{27}$$

Combining the three inequalities (24), (25), and (27) yields the inequality that allows us to contradict the assumption that $\widetilde{\ell}(x) > \widehat{L}(x)$:

$$\begin{aligned} \varepsilon &< \widehat{\overline{\mathrm{err}}}_{i+2,+} \quad \text{by assumption} \\ &= \overline{\mathrm{err}}_{i+2} + \widehat{M}_{\widehat{L}(x)} - \min\bigl(f(x_i), f(x_{i+1})\bigr) \quad \text{by Step 1 of Algorithm M} \\ &= \frac{1}{8} \mathfrak{C}(3h_{\widehat{L}(x)}) h_{\widehat{L}(x)}^2 \left\| f'' \right\|_{\tilde{I}_{x,\widehat{L}(x)}} + f(x_*) + \frac{1}{8} h_{\widehat{L}(x)}^2 \left\| f'' \right\|_{I_{x_*,\widehat{L}(x)}} + \varepsilon \\ &\quad - f(x) + 2h_{\widehat{L}(x)} \left| f'(x) \right| + 2h_{\widehat{L}(x)}^2 \left\| f'' \right\|_{\tilde{I}_{x,\widehat{L}(x)}} \quad \text{by (24), (25), and (27)} \\ &\le \varepsilon + \left\{ \left[\frac{1}{8} \mathfrak{C}\left(3h_{\widehat{L}(x)}\right) + 2 \right] \left\| f'' \right\|_{\tilde{I}_{x,\widehat{L}(x)}} + \frac{1}{8} \left\| f'' \right\|_{I_{x_*,\widehat{L}(x)}} \right\} h_{\widehat{L}(x)}^2 + 2 \left| f'(x) \right| h_{\widehat{L}(x)} + [f(x_*) - f(x)] \\ &\le \varepsilon \quad \text{by (23).} \end{aligned}$$

This gives a contradiction and completes the proof. □

If $f(x)$ is close to the minimum function value, $f(x_*)$, for x in much of $[a, b]$, then $\widehat{L}(x)$ may be quite large, and $L(x)$ determines the computational cost of Algorithm M. In this case, the computational cost for minimization is similar to that for function approximation. However, if f attains its minimum at only a finite number of points, then for vanishing ε, $\widetilde{L}(x) = \widehat{L}(x)$ for nearly all x, and the computational cost for minimization is significantly smaller than that for function approximation.

The minimization problem (MIN) for functions in the whole Sobolev space $W^{2,\infty}$ has a similar lower complexity bound as (19) for the function approximation problem by a similar proof. However, for functions only in the cone $\mathcal{C}$, we have not yet derived a lower bound on the complexity of the minimization problem (MIN) for functions in $\mathcal{C}$.

5. Numerical examples

Together with our collaborators, we have developed the Guaranteed Automatic Integration Library (GAIL) [4]. This MATLAB software library implements algorithms that provide answers to univariate and multivariate integration problems, as well as (APP) and (MIN), by automatically determining the sampling needed to satisfy a user-provided error tolerance. GAIL is under active development. It implements our best adaptive algorithms and upholds the principles of reproducible and reliable computational science as elucidated in Choi et al. [3,6]. We have adopted practices including input

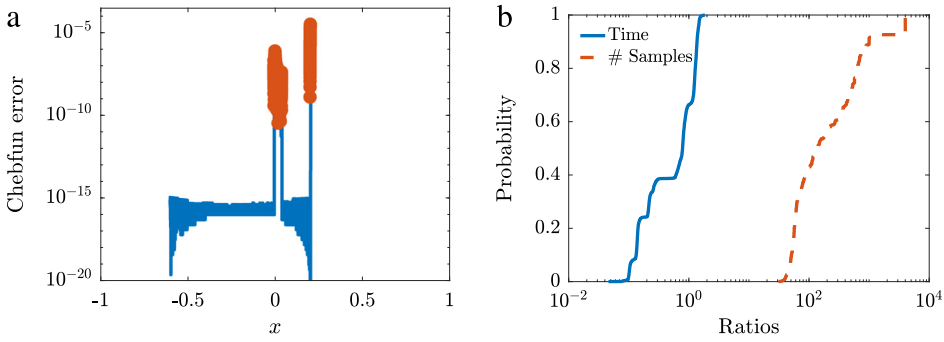

Fig. 4. (a) The approximation errors for $f_1(x)$, $x \in [-1, 1]$, with $-c = 0.2 = \delta$ using Chebfun with an error tolerance of 10^{-12}. (b) An empirical distribution function of performance ratios based on 1000 simulations for each test function in (28): `funappx_g` time/Chebfun time (solid), `funappx_g` # of samples/Chebfun # of samples (dashed). The data for this figure is conditionally reproducible by `funappx_g_test.m` and `LocallyAdaptivePaperFigs.m` in GAIL.

parsing, extensive testing, code comments, a user guide [5], and case studies. Algorithms A and M described here are implemented as GAIL functions `funappx_g` and `funmin_g`, respectively in GAIL version 2.2. The following examples showcase the merits and drawbacks of our algorithms. We compare them to the performance of algorithms in MATLAB and the Chebfun toolbox.

Chebfun [10] is a MATLAB toolbox that approximates functions in terms of a Chebyshev polynomial basis, in principle to machine precision ($\approx 10^{-15}$) by default. In this example, we show that it fails to reach its intended error tolerance for the function f_1 defined in (8) with $-c = 0.2 = \delta$. Fig. 4(a) shows the absolute errors of Chebfun's approximation to f_1 with an input error tolerance 10^{-12}, and the "splitting" option turned on to allow Chebfun to construct a piecewise polynomial interpolant if derivative discontinuities are detected. However, Chebfun produces some pointwise errors computed at a partition of $[-1, 1]$ with even subinterval length 10^{-5} to be greater than 10^{-5}.

In contrast, the pointwise errors of the piecewise linear interpolant produced by `funappx_g` are uniformly below the error tolerance. Unfortunately, the time taken by `funappx_g` is about 30 times as long as the time required by Chebfun.

Next, we compare our adaptive algorithms with Chebfun for random samples from the following families of test functions defined on $[-1, 1]$:

$$f_1(x) \quad \text{defined in (8)}, \qquad \delta = 0.2, \qquad c \sim \mathcal{U}[0, 0.6], \tag{28a}$$

$$f_2(x) = x^4 \sin(d/x), \quad d \sim \mathcal{U}[0, 2], \tag{28b}$$

$$f_3(x) = 10x^2 + f_2(x), \tag{28c}$$

where $\mathcal{U}[a, b]$ represents a uniform distribution over $[a, b]$. We set $n_{\text{ninit}} = 250$, $\mathfrak{C}(h) = 10\mathfrak{h}/(\mathfrak{h} - h)$, and $\varepsilon = 10^{-6}$. Our new algorithm `funappx_g` and Chebfun are used to approximate 1000 random test functions from each family. For Chebfun we override the default tolerance to 10^{-6}, and switch on the splitting feature to allow piecewise Chebyshev polynomials for approximation. Success is determined by whether a discrete approximation to the L^∞ error is no greater than the error tolerance.

We see in Table 1 that `funappx_g` obtains the correct answer in all cases, even for f_2, which is outside the cone $\mathcal{C}$. Since it is a higher order algorithm, Chebfun generally uses substantially fewer samples than `funappx_g`, but its run time is longer than `funappx_g` for a significant proportion of the cases; see Fig. 4(b). Moreover, Chebfun rarely approximates the test functions satisfactorily.

Similar simulation tests have been run to compare our `funmin_g`, MATLAB's `fminbnd`, and Chebfun's `min`, but this time $n_{\text{ninit}} = 20$ for `funmin_g`. The results are summarized in the lower half of Table 1. Our `funmin_g` achieves 100% success for all families of test functions with substantially fewer sampling points and run time than `funappx_g`. This is because `funmin_g` does not sample densely where the function is not close to its minimum value. Although MATLAB's `fminbnd` uses far fewer function values than `funmin_g`, it cannot locate the global minimum (at the left boundary) for

Table 1
Comparison of number of sample points, computational time, and success rates of `funappx_g` and Chebfun in upper table; `funmin_g`, `fminbnd`, and Chebfun's `min` in lower table. This table is conditionally reproducible by `funappx_g_test.m` and `funmin_g_test.m` in GAIL.

	Mean # samples			Mean time used			Success (%)		
	`funappx_g`	Chebfun		`funappx_g`	Chebfun		`funappx_g`	Chebfun	
f_1	6557	116		0.0029	0.0205		100	0	
f_2	5017	43		0.0031	0.0051		100	3	
f_3	15698	22		0.0049	0.0036		100	3	
	`funmin_g`	`fminbnd`	`min`	`funmin_g`	`fminbnd`	`min`	`funmin_g`	`fminbnd`	`min`
$-f_1$	111	8	116	0.0029	0.0006	0.0256	100	100	14
f_2	48	22	43	0.0028	0.0007	0.0063	100	27	60
f_3	108	9	22	0.0028	0.0007	0.0037	100	100	35

about 70% of the f_2 test cases. Chebfun's `min` uses fewer points than `funmin_g`, but Chebfun is slower and less accurate than `funmin_g` for these tests.

6. Discussion

Adaptive and automatic algorithms are popular because they require only a (black-box) function and an error tolerance. Such algorithms exist in a variety of software packages. We have highlighted those found in MATLAB and Chebfun because they are among the best. However, as we have shown by numerical examples, these algorithms may fail. Moreover, there is no theory to provide necessary conditions for failure, or equivalently, sufficient conditions for success.

Our Algorithms A (`funappx_g`) and M (`funmin_g`) are locally adaptive and have sufficient conditions for success. Although it may be difficult to verify those conditions in practice, the theory behind these algorithms provides several advantages:

- The cone, $\mathcal{C}$, is intuitively explained as a set of functions whose second derivatives do not change drastically over a small interval. This intuition can guide the user in setting the parameters defining $\mathcal{C}$, if desired.
- The norms of f and its derivatives appearing in the upper bounds of computational cost in Theorems 2 and 6 may be unknown, but these theorems explain how the norms influence the time required by our algorithms.
- Our Algorithm A has been shown to be asymptotically optimal for the complexity of the function approximation problem (APP).

The minimum horizontal scale of functions in $\mathcal{C}$ is roughly $1/n_{\text{ninit}}$. The computational cost of our algorithms is at least n_{ninit}, but n_{ninit} is not a multiplicative factor. Increasing n_{ninit} makes our new algorithms more robust, and it may increase the minimum number of sample points and computational cost, if any, only mildly.

As mentioned in the introduction, there are general theorems providing sufficient conditions under which adaption provides no advantage. Our setting fails to satisfy those conditions because $\mathcal{C}$ is not convex. One may average two mildly spiky functions in $\mathcal{C}$ – whose spikes have opposite signs and partially overlap – to obtain a very spiky function outside $\mathcal{C}$.

Nonadaptive algorithms are unable to solve (APP) or (MIN) using a finite number of function values if the set of interesting functions, $\mathcal{C}$, is a cone, unless there exist nonadaptive algorithms that solve these problems exactly. Suppose that some nonadaptive, Algorithm A satisfies (APP) for some cone $\mathcal{C}$, and that for an error tolerance ε, this Algorithm A requires n function values. For any positive c, define $A^*(f,\varepsilon) = A(cf,\varepsilon)/c$ for all $f \in \mathcal{C}$. Then $\|f - A^*(f,\varepsilon)\| = \|cf - A(cf,\varepsilon)\|/c \le \varepsilon/c$ for all $f \in \mathcal{C}$ since cf is also in $\mathcal{C}$. Thus, A^* satisfies (APP) for error tolerance ε/c, using the same number of function values as A. Making c arbitrarily large establishes the existence of a nonadaptive algorithm that solves (APP) exactly.

Our algorithms do not take advantage of higher orders of smoothness that the input function may have. We view the present work as a stepping stone to developing higher order algorithms. Nonlinear splines or higher degree polynomials, such as those used in Chebfun, are potential candidates.

Acknowledgments

We dedicate this article to the memory of our colleague Joseph F. Traub, who passed away on August 24, 2015. He was a polymath and an influential figure in computer science and applied mathematics. He was the founding Editor-in-Chief of the Journal of Complexity and we are grateful to his tremendous impact and lifelong service to our research community.

We thank Greg Fasshauer, Erich Novak, the GAIL team, and two anonymous referees for their valuable comments and suggestions. This research was supported in part by grants NSF-DMS-1522687 and NSF-DMS-0923111.

References

[1] R.P. Brent, Algorithms for Minimization Without Derivatives, Dover Publications, Inc., Mineola, NY, 2013, Republication of the 1973 edition by Prentice-Hall, Inc..
[2] R.L. Burden, J.D. Faires, A.M. Burden, Numerical Analysis, tenth ed., Brooks/Cole, 2016.
[3] S.-C.T. Choi, MINRES-QLP Pack and reliable reproducible research via staunch scientific software, J. Open Res. Softw. 2 (2014) 1–7. http://dx.doi.org/10.5334/jors.bb.
[4] S.-C.T. Choi, Y. Ding, F.J. Hickernell, L. Jiang, Ll.A. Jiménez Rugama, X. Tong, Y. Zhang, X. Zhou, GAIL: Guaranteed Automatic Integration Library (versions 1.0–2.1), MATLAB software, 2013–2015. URL: http://gailgithub.github.io/GAIL_Dev/.
[5] S.-C.T. Choi, Y. Ding, F.J. Hickernell, L. Jiang, Ll.A. Jiménez Rugama, X. Tong, Y. Zhang, X. Zhou, GAIL—Guaranteed Automatic Integration Library in MATLAB: Documentation for version 2.1, arXiv:1503.06544 (2015).
[6] S.-C.T. Choi, F.J. Hickernell, IIT MATH-573 Reliable Mathematical Software, Illinois Institute of Technology, 2013. Course slides at [4].
[7] N. Clancy, Y. Ding, C. Hamilton, F.J. Hickernell, Y. Zhang, The cost of deterministic, adaptive, automatic algorithms: Cones, not balls, J. Complexity 30 (2014) 21–45. http://dx.doi.org/10.1016/j.jco.2013.09.002.
[8] Y. Ding, Guaranteed Adaptive Univariate Function Approximation (Ph.D. thesis), Illinois Institute of Technology, 2015.
[9] G. Forsythe, M. Malcolm, C. Moler, Computer Methods for Mathematical Computations, Prentice-Hall, 1976.
[10] N. Hale, L.N. Trefethen, T.A. Driscoll, Chebfun Version 5.4, 2016.
[11] F.J. Hickernell, L. Jiang, Y. Liu, A.B. Owen, Guaranteed conservative fixed width confidence intervals via Monte Carlo sampling, in: J. Dick, F.Y. Kuo, G.W. Peters, I.H. Sloan (Eds.), Monte Carlo and Quasi-Monte Carlo Methods 2012, in: Springer Proceedings in Mathematics and Statistics, vol. 65, Springer-Verlag, Berlin, 2013, pp. 105–128.
[12] F.J. Hickernell, Ll.A. Jiménez Rugama, Reliable adaptive cubature using digital sequences, in: R. Cools, D. Nuyens (Eds.), Monte Carlo and Quasi-Monte Carlo Methods: MCQMC, Leuven, Belgium, April 2014, in: Springer Proceedings in Mathematics and Statistics, vol 163, Springer-Verlag, Berlin, 2016, pp. 367–383. arXiv:1410.8615 [math.NA].
[13] M. Horn, Optimal algorithms for global optimization in case of unknown Lipschitz constant, J. Complexity 22 (2006) 50–70.
[14] L. Jiang, Guaranteed Adaptive Monte Carlo Methods for Estimating Means of Random Variables (Ph.D. thesis), Illinois Institute of Technology, 2016.
[15] Ll.A. Jiménez Rugama, F.J. Hickernell, Adaptive multidimensional integration based on rank-1 lattices, in: R. Cools, D. Nuyens (Eds.), Monte Carlo and Quasi-Monte Carlo Methods: MCQMC, Leuven, Belgium, April 2014, in: Springer Proceedings in Mathematics and Statistics, vol 163, Springer-Verlag, Berlin, 2016, pp. 407–422. arXiv:1411.1966.
[16] R. Moore, R. Kearfott, M. Cloud, Introduction To Interval Analysis, Cambridge University Press, Cambridge, 2009.
[17] E. Novak, On the power of adaption, J. Complexity 12 (1996) 199–237.
[18] L. Plaskota, Automatic integration using asymptotically optimal adaptive Simpson quadrature, Numer. Math. 131 (2015) 173–198.
[19] L. Plaskota, G.W. Wasilkowski, Y. Zhao, The power of adaption for approximating functions with singularities, Math. Comp. 77 (2008) 2309–2338.
[20] S.M. Rump, Verification methods: Rigorous results using floating-point arithmetic, Acta Numer. 19 (2010) 287–449.
[21] S.M. Rump, INTLAB - INTerval LABoratory, in: T. Csendes (Ed.), Developments in Reliable Computing, Kluwer Academic Publishers, Dordrecht, 1999, pp. 77–104. URL: http://www.ti3.tuhh.de/rump/.
[22] The MathWorks, Inc., MATLAB 9.1, Natick, MA, 2016.
[23] X. Tong, A Guaranteed, Adaptive, Automatic Algorithm for Univariate Function Minimization (Master's thesis), Illinois Institute of Technology, 2014.
[24] J.F. Traub, G.W. Wasilkowski, H. Woźniakowski, Information-Based Complexity, Academic Press, Boston, 1988.
[25] H. Woźniakowski, A survey of information-based complexity, J. Complexity 1 (1985) 11–84.