Journal of Complexity 30 (2014) 21–45

Contents lists available at ScienceDirect

Journal of Complexity

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

The cost of deterministic, adaptive, automatic algorithms: Cones, not balls

Nicholas Clancy, Yuhan Ding, Caleb Hamilton, Fred J. Hickernell *, Yizhi Zhang

Room E1-208, Department of Applied Mathematics, Illinois Institute of Technology, 10 W. 32nd St., Chicago, IL 60616, United States

ARTICLE INFO

Article history:
Received 10 March 2013
Accepted 19 September 2013
Available online 8 October 2013

Keywords:
Function recovery
Guarantee
Integration
Quadrature

ABSTRACT

Automatic numerical algorithms attempt to provide approximate solutions that differ from exact solutions by no more than a user-specified error tolerance. The computational cost is often determined *adaptively* by the algorithm based on the function values sampled. While adaptive, automatic algorithms are widely used in practice, most lack *guarantees*, i.e., conditions on input functions that ensure that the error tolerance is met.

This article establishes a framework for guaranteed, adaptive, automatic algorithms. Sufficient conditions for success and two-sided bounds on the computational cost are provided in Theorems 2 and 3. Lower bounds on the complexity of the problem are given in Theorem 6, and conditions under which the proposed algorithms have optimal order are given in Corollary 1. These general theorems are illustrated for univariate numerical integration and function recovery via adaptive algorithms based on linear splines.

The key to these adaptive algorithms is performing the analysis for *cones* of input functions rather than balls. Cones provide a setting where adaption may be beneficial.

1. Introduction

Automatic algorithms conveniently determine the computational effort required to obtain an approximate answer that differs from the true answer by no more than an error tolerance, ε. The

* Corresponding author.
E-mail addresses: nclancy@hawk.iit.edu (N. Clancy), yding2@hawk.iit.edu (Y. Ding), chamilt2@hawk.iit.edu (C. Hamilton), hickernell@iit.edu (F.J. Hickernell), yzhang97@hawk.iit.edu (Y. Zhang).

0885-064X/$ – see front matter
http://dx.doi.org/10.1016/j.jco.2013.09.002

required inputs are both ε and a black-box routine that provides function values. Unfortunately, most commonly used adaptive, automatic algorithms are not guaranteed to provide answers satisfying the error tolerance. On the other hand, most existing guaranteed automatic algorithms are not adaptive, i.e., they do not adjust their effort based on information about the function obtained through sampling. The goal here is to construct adaptive, automatic algorithms that are guaranteed to satisfy the error tolerance.

1.1. Non-adaptive, automatic algorithms for balls of input functions

Let $\mathcal{F}$ be a *linear space* of input functions defined on $\mathcal{X}$ with *semi-norm* $|\cdot|_{\mathcal{F}}$, let $\mathcal{G}$ be a linear space of outputs with *norm* $\|\cdot\|_{\mathcal{G}}$, and let $S : \mathcal{F} \to \mathcal{G}$ be a *solution operator*. Suppose that one has a sequence of fixed-cost algorithms, $\{A_n\}_{n\in\mathcal{I}}$, indexed by their computational cost, n, with $\mathcal{I} \subseteq \mathbb{N}_0$. Furthermore, suppose that there is some known error bound of the form

$$\|S(f) - A_n(f)\|_{\mathcal{G}} \le h(n)|f|_{\mathcal{F}}, \tag{1a}$$

where $h : \mathcal{I} \to [0, \infty)$ is non-negative valued and non-increasing. Note that A_n must be exact for input functions with vanishing semi-norms, i.e., $S(f) = A_n(f)$ if $|f|_{\mathcal{F}} = 0$. Furthermore, h is assumed to have zero infimum, which makes it possible to define h^{-1} for all positive numbers:

$$\inf_{n\in\mathcal{I}} h(n) = 0, \qquad h^{-1}(\varepsilon) = \min\{n \in \mathcal{I} : h(n) \le \varepsilon\}, \quad \varepsilon > 0. \tag{1b}$$

Error bound (1) allows one to construct an automatic, yet non-adaptive, algorithm that is guaranteed for input functions in a prescribed $\mathcal{F}$-ball.

Algorithm 1 (*Non-Adaptive, Automatic*)**.** Let $\{A_n\}_{n\in\mathcal{I}}$ be defined as above, and let σ be a fixed positive number. For any input function $f \in \mathcal{B}_\sigma := \{f \in \mathcal{F} : |f|_{\mathcal{F}} \le \sigma\}$ and any positive error tolerance ε, find the computational cost needed to satisfy the error tolerance, $n = h^{-1}(\varepsilon/\sigma)$. Return $A_n(f)$ as the answer.

Theorem 1. *For $\mathcal{F}$, $|\cdot|_{\mathcal{F}}$, $\mathcal{G}$, $\|\cdot\|_{\mathcal{G}}$, S as described above, and under the assumptions of Algorithm 1, if f lies in the ball $\mathcal{B}_\sigma$, then the answer provided by Algorithm 1 must satisfy the error tolerance, i.e., $\|S(f) - A_n(f)\|_{\mathcal{G}} \le \varepsilon$.*

Algorithm 1, Theorem 1, and the other theoretical results in this article related to Algorithm 1 are essentially known. They serve as a benchmark to which we may compare our new adaptive algorithms.

Algorithm 1 has drawbacks. If it works for $f \in \mathcal{F}$, it may not work for $cf \in \mathcal{F}$, where $c > 1$, because cf may fall outside the ball $\mathcal{B}_\sigma$. Moreover, although error bound (1a) depends on $|f|_{\mathcal{F}}$, the computational cost of Algorithm 1 does not depend on $|f|_{\mathcal{F}}$. The cost is the same whether $|f|_{\mathcal{F}} = \sigma$ or $|f|_{\mathcal{F}}$ is much smaller than σ. This is because Algorithm 1 is not adaptive.

1.2. Adaptive, automatic algorithms for cones of input functions

Adaptive, automatic algorithms are common in numerical software packages. Examples include MATLAB's `quad` and `integral` [18], the quadrature algorithms in the NAG Library [19], and the MATLAB Chebfun toolbox [6]. While these adaptive algorithms work well for many cases, they have no rigorous justification. The methods used to determine the computational cost are either heuristics or asymptotic error estimates that do not hold for finite sample sizes.

In this article we derive guaranteed adaptive, automatic algorithms. These adaptive algorithms use $\{A_n\}_{n\in\mathcal{I}}$ with known h as described in (1) and satisfying some additional technical conditions in (6). Rather than assuming an upper bound on $|f|_{\mathcal{F}}$, our adaptive algorithms use function data to construct *rigorous* upper bounds on $|f|_{\mathcal{F}}$. We highlight the requirements here.

The key idea is to identify a suitable semi-norm on $\mathcal{F}$, $|\cdot|_{\tilde{\mathcal{F}}}$, that is weaker than $|\cdot|_{\mathcal{F}}$, i.e., there exists a positive constant $\tau_{\min}$ for which

$$\tau_{\min}|f|_{\tilde{\mathcal{F}}} \le |f|_{\mathcal{F}} \quad \forall f \in \mathcal{F}. \tag{2}$$