IMA Journal of Applied Mathematics (2021) **00**, 1–26
https://doi.org/hxad035

Analyzing inexact hypergradients for bilevel learning

MATTHIAS J. EHRHARDT
Department of Mathematical Sciences, University of Bath, Bath, BA2 7AY, UK

AND

LINDON ROBERTS*
School of Mathematics and Statistics, University of Sydney, Camperdown, 2006, NSW, Australia
*Corresponding author: lindon.roberts@sydney.edu.au

[Received on Date Month Year; revised on Date Month Year; accepted on Date Month Year]

Estimating hyperparameters has been a long-standing problem in machine learning. We consider the case where the task at hand is modeled as the solution to an optimization problem. Here the exact gradient with respect to the hyperparameters cannot be feasibly computed and approximate strategies are required. We introduce a unified framework for computing hypergradients that generalizes existing methods based on the implicit function theorem and automatic differentiation/backpropagation, showing that these two seemingly disparate approaches are actually tightly connected. Our framework is extremely flexible, allowing its subproblems to be solved with any suitable method, to any degree of accuracy. We derive a priori and computable a posteriori error bounds for all our methods, and numerically show that our a posteriori bounds are usually more accurate. Our numerical results also show that, surprisingly, for efficient bilevel optimization, the choice of hypergradient algorithm is at least as important as the choice of lower-level solver.

Keywords: Hyperparameter Optimization; Bilevel Optimization; Automatic Differentiation.

1. Introduction

In this work we consider the hyperparameter tuning problem framed as a bilevel optimization problem [1, 2, 3] where we aim to solve

$$\min_{\theta\in\mathbb{R}^n} F(\theta) := \frac{1}{m}\sum_{i=1}^{m} f_i(x_i^*(\theta)) + r(\theta) \tag{1.1a}$$

$$\text{s.t.} \quad x_i^*(\theta) := \arg\min_{x\in\mathbb{R}^d} g_i(x,\theta), \qquad \forall i = 1,\dots,m, \tag{1.1b}$$

where the lower-level functions g_i are smooth in x and θ and strongly convex in x, and the upper-level functions f_i and r are smooth but possibly nonconvex. Our main motivation for studying (1.1) is the problem of supervised bilevel learning, where we may have $f_i(x) = \|x - x_i^\dagger\|^2$ for example, where $x_i^\dagger$ is the desired outcome of the lower-level problem (1.1b). Problems of the form (1.1) are ubiquitous in

every aspect of science and tasks such as clustering, time series analysis and image reconstruction can be modeled as such. As a simple example with $n = 1$, we could think of $\theta \geq 0$ as a choice of regularization weight suitable for a family of regularized regression problems g_i, i.e. $g_i(x, \theta) = \frac{1}{2}\|Ax - y_i\|^2 + \theta R(x)$ for $i = 1, \ldots, m$. Similarly one can select a regularizer [4] or noise model [5]. Other models use many more parameters like an input-convex neural network as a regularizer [6, 7] or the sampling of the forward operator for image compression [8, 9] or MRI [10].

When the number of parameters n is small, the problem (1.1) can be efficiently solved by search methods (e.g., [11, 12]) or derivative-free approaches, see, e.g., [13, 14] for general hyperparameter search and [15] for bilevel learning.

Motivated by bilevel learning, we are interested in algorithms which can scale to millions of parameters (or more), and so consider (1.1) with first-order methods used for solving the lower-level problem (1.1b) and upper-level problem (1.1a). If the functions f_i and g_i in (1.1a) and (1.1b) respectively are sufficiently smooth, it is well-known (see, e.g., [16]) that the gradients of the upper-level objective (1.1a) (also called hypergradients) can be computed via

$$\nabla(f_i \circ x_i^*)(\theta) = \partial x_i^*(\theta)^T \nabla f_i(x_i^*(\theta)), \qquad i = 1, \ldots, m, \tag{1.2}$$

where $\partial x_i^*(\theta)$ is the derivative of the minimizer $x_i^*(\theta)$ with respect to the parameters θ. However, in the context of large-scale problems and general lower-level objectives g_i, access to the true lower-level minimizers $x_i^*(\theta)$ is unreasonable and so (1.2) cannot be evaluated.

The primary purpose of this work is to develop methods for efficiently evaluating $\nabla(f_i \circ x_i^*)$. Such methods compute an approximate lower-level solution to be used in some form to then compute an approximation to the hypergradient (1.2). While this approach has been often used successfully in practice, the interplay between the accuracy of the computed approximation to $x_i^*(\theta)$ and its impact on the hypergradients are neither fully understood nor fully utilized.

In this work we introduce a unified framework for computing approximate hypergradients, which come with numerous concrete implementations and several error bounds suitable for different purposes. Our main theoretical results are:

- Showing that two promising hypergradient estimation algorithms, based on the implicit function theorem [17, 18] and inexact automatic differentiation (AD)/backpropagation [19] respectively, are essentially the same underlying algorithm. This surprising result allows us to unify two very disparate hypergradient estimation methodologies into a general framework.
- Deriving general error bounds for the underlying general hypergradient estimation framework. Our error bounds are both a priori, based on known convergence rates of the constituent algorithms, and a posteriori, yielding computable error estimates suitable for use inside a bilevel optimization framework.

Our numerical results then compare the accuracy and efficiency of different hypergradient estimation techniques, as well as studying the impact of hypergradient algorithms on the performance of bilevel optimization routines. Importantly, and perhaps surprisingly, our numerical evidence shows that, in the context of bilevel optimization, *the choice of hypergradient algorithm is at least as important as the choice of lower-level solver*.

1.1. *Existing work*

An explicit form for the derivative of the minimizer of a smooth, strongly convex optimization problem is given by the classical implicit function theorem (e.g., [16]). This formulation, where the resulting

linear system of equations is solved inexactly using the conjugate gradient (CG) method, was used as the basis of the Hyperparameter Optimization with Approximate Gradient (HOAG) algorithm for bilevel optimization in [17]. The analysis of the underlying hypergradient estimation was refined in the more recent work [18]. Our unified approach is fundamentally based on these ideas, but we demonstrate how this approach also incorporates AD-based methods and have a more refined error analysis.

The other hypergradient estimation technique we consider is reverse-mode automatic differentiation, also known as backpropagation, of a lower-level iterative solver. Originally, AD for iterative methods was first applied to fixed point iterations [20]. This was extended more recently in [19] to parametric strongly convex optimization using gradient descent and heavy ball momentum. Here, the authors prove sub-optimal linear convergence rates of hypergradients using AD, and introduce an 'inexact' AD method with improved linear convergence rates. Our analysis unifies the accelerated 'inexact' approach from [19] with the traditional analysis from [17, 18], introduces a new family of a posteriori bounds, and carefully considers the numerical performance of these options.

Both these perspectives (implicit function theorem and backpropagation) were considered separately in [21] where the lower-level problem (1.1b) is replaced with a fixed-point iteration, and (separate) a priori bounds are derived in both cases. Our work is similar in considering both techniques, but we give a unified perspective showing how they can be seen as the same method, and hence unified results can be shown. We also extend the analysis by providing computable a posteriori bounds, and showing that these are often tighter in practice.

More generally, several approaches exist for solving the general bilevel problem (1.1) (instead of considering specifically the estimation of hypergradients). This typically involves alternating between a given number of iterations of a solver for the lower-level problem (1.1b) with a (possibly different) number of iterations of an upper-level solver, both typically first-order methods. In [22], the number of lower-level gradient descent iterations is pre-specified. In [23], the authors use a constant number of lower-level iterations for each upper-level iteration. Alternatively, [24] uses just one iteration but with different upper- vs. lower-level stepsizes.

When the lower-level problem is replaced by a finite algorithm, then gradients can be computed exactly, see, e.g., [25, 26, 27]. This is not the case here as the lower-level solution can only be approximated by the limit of an algorithm and thus requires special care.

In [28] the authors propose an alternative optimality system including the upper-level unknown, the lower-level unknown and the adjoint state which corresponds to the derivative of the lower-level unknown with respect to the upper-level parameters.

Both the upper- and lower-level problem can be extended to stochastic optimization problems with only stochastic gradients available, see, e.g., [23, 29]. While this is clearly a challenging research question itself, it is fairly independent of the challenge considered in the present paper.

1.2. *Contributions*

In this work we introduce a unified perspective on hypergradient estimation, incorporating the different techniques analyzed in [17, 18, 19, 21]. This includes estimates of the hypergradient based on the implicit function theorem, with inexact linear solves using iterative methods such as CG, and gradient descent and heavy ball variants of the inexact AD method from [19] (which converge faster than standard AD). We show that all of these approaches correspond to the same basic procedure: approximately solve the lower-level problem and then approximately solve the resulting implicit function theorem linear system. As far as we are aware, showing that inexact AD is the same as using the implicit function theorem is a new result showing a surprising connection between analytic and symbolic gradient estimation.

With our resulting unified method, we prove a priori bounds similar to those in [17, 18, 21], where first-order methods applied to both subproblems yields a linear convergence rate. Our analysis is extremely general: it enables both subproblems to be solved with any suitable algorithm, and run for any number of iterations or up to any stopping tolerance.

We then extend our analysis to prove a posteriori bounds on our flexible hypergradient estimators. Here we construct explicit and computable error bounds on the hypergradient. These bounds are potentially useful if we wish to apply existing algorithms for nonconvex optimization with inexact gradients to solve (1.1), for example frameworks such as [30, 31].

We then study different variants of our hypergradient algorithm numerically. First we use a simple linear least-squares problem to demonstrate the correctness of our bounds, and show that our new a posteriori analysis typically provides stronger bounds than the more common a priori analysis (as well as being computable in practice). We then consider a data hypercleaning problem, comparing different combinations of algorithms for the lower-level solver and the implicit function theorem linear solver. Unsurprisingly, the better the algorithms used (e.g., heavy ball rather than gradient descent), the faster the optimization progresses. However perhaps surprisingly, we show that the choice of hypergradient algorithm is at least as important as the choice of lower-level solver for the purposes of efficient bilevel optimization, demonstrating the importance of the hypergradient estimation problem. Lastly, we conclude by applying our method to the bilevel problem of finding a neural net regularizer for image denoising, and demonstrate that the obtained regularizer can outperform the standard total variation regularizer.

The paper is structured as follows: in Section 2 we summarize the existing implicit function theorem and inexact backpropagation theory. In Section 3 we prove that inexact backpropagation is just a special case of the implicit function theorem approach which motivates our unified hypergradient framework. We prove the a priori and a posteriori error bounds for our approach in Section 4 and give numerical results in Section 5.

1.3. *Notation*

Throughout, we will use $\|\cdot\|$ to be the Euclidean norm of vectors and operator 2-norm of matrices. Both partial and total derivatives are denoted by ∂. If the function we take the derivative of is scalar-valued, then we denote its derivative by ∇ and refer to it as the gradient. Specifically, given (1.1) we have

- $\partial_y g_i : \mathbb{R}^d \times \mathbb{R}^n \to \mathbb{R}^d$ and $\partial_{yy} g_i : \mathbb{R}^d \times \mathbb{R}^n \to \mathbb{R}^{d\times d}$ are the gradient and Hessian (respectively) of $g_i(y,\theta)$ with respect to y (for fixed θ). That is, $[\partial_y g_i(y,\theta)]_j := \frac{\partial g_i(y,\theta)}{\partial y_j}$ and $[\partial_{yy} g_i(y,\theta)]_{j,k} := \frac{\partial^2 g_i(y,\theta)}{\partial y_j \partial y_k}$;
- $\partial_y \partial_\theta g_i : \mathbb{R}^d \times \mathbb{R}^n \to \mathbb{R}^{d\times n}$ is the Jacobian of $\partial_y g_i(y,\theta)$ with respect to θ, where $[\partial_y \partial_\theta g_i(y,\theta)]_{j,k} := \frac{\partial^2 g_i(y,\theta)}{\partial y_j \partial \theta_k}$;
- $\partial x_i^* : \mathbb{R}^n \to \mathbb{R}^{d\times n}$ is the derivative of $x_i^*(\theta)$ with respect to θ, with $[\partial x_i^*(\theta)]_{j,k} := \frac{\partial [x_i^*(\theta)]_j}{\partial \theta_k}$;
- $\nabla f_i : \mathbb{R}^d \to \mathbb{R}^d$ is the gradient of $f_i(x)$ with respect to x (or θ for ∇F and ∇r).

At various points throughout we will drop the indexing on i and explicit dependencies on θ for simplicity of presentation.

Note that this framework also handles complex-valued inputs and operators (with real-valued objectives to ensure optimization over an ordered field), by treating the real and imagingary parts of any complex quantities as two real-valued quantities.

2. Background

We begin by outlining three existing approaches for calculating the hypergradient

$$\nabla F(\theta) = \frac{1}{m}\sum_{i=1}^{m} \nabla(f_i \circ x_i^*)(\theta) + \nabla r(\theta) = \frac{1}{m}\sum_{i=1}^{m} \partial x_i^*(\theta)^T \nabla f_i(x_i^*(\theta)) + \nabla r(\theta), \tag{2.1}$$

where the second equality follows from the chain rule.[1] In particular, we are concerned with how we may accurately compute $\nabla F(\theta)$ in the setting where $x_i^*(\theta)$ cannot be exactly determined. For ∂x_i^* to be well-defined, we require the following assumptions on g_i.

Assumption 1 *For each $i = 1, \dots, m$, the lower-level objective g_i is twice continuously differentiable in y and there exists $0 < \mu_i(\theta) \leq L_i(\theta)$ such that $\mu_i(\theta) I \preceq \partial_{yy} g_i(y,\theta) \preceq L_i(\theta) I$ for all y, where $A \preceq B$ means $B - A$ is positive semidefinite. Furthermore, each of g_i, $\partial_y g_i$ and $\partial_{yy} g_i$ are continuous in θ.*

In practice, Assumption 1 may require the domain of θ to be restricted, which means more careful selection of the upper-level solver is required, but this does not affect the results below regarding the inexact computation of $\nabla F(\theta)$.

It is well known (see, e.g., [15, 16]) that under Assumption 1, the map $\theta \mapsto x_i^*(\theta)$ is continuously differentiable with derivative $\partial x_i^*(\theta) = D_i(x_i^*(\theta), \theta)^T$. This formulation relies on the function

$$D_i(y,\theta) := -B_i(y,\theta)^T A_i(y,\theta)^{-1} \in \mathbb{R}^{n \times d} \tag{2.2}$$

and associated matrices

$$A_i(y,\theta) := \partial_{yy} g_i(y,\theta) \in \mathbb{R}^{d \times d} \quad \text{and} \quad B_i(y,\theta) := \partial_y \partial_\theta g_i(y,\theta) \in \mathbb{R}^{d \times n}. \tag{2.3}$$

Note that Assumption 1 implies that $A_i(y,\theta)$ is invertible and thus D_i is well-defined. Moreover, for all y and θ it holds that

$$\|A_i(y,\theta)^{-1}\| \leq \mu_i(\theta)^{-1}. \tag{2.4}$$

Remark 1 *From here, we drop the indexation by i and the explicit dependence on θ (since θ does not vary throughout) for simplicity of notation. That is, we will now write $A(y)$, x^* and μ instead of $A_i(y,\theta)$, $x_i^*(\theta)$ and $\mu_i(\theta)$, for example.*

Inserting (2.2) into (2.1) and ignoring the average and r for now, leads to another formulation of the hypergradient

$$h^* := -B(x^*)^T A(x^*)^{-1} \nabla f(x^*) = D(x^*) \nabla f(x^*). \tag{2.5}$$

This formulation of the hypergradient is not practical: it is too costly to compute to high accuracy for most relevant applications since x^* has to be computed iteratively (via a suitable strongly convex solver) and a system of linear equations the size of the number of lower-level unknowns (often exceeding millions in imaging applicatons) has to be solved. Therefore, we investigate methods which acknowledge

[1] Our analysis is also suitable for the more general case where f_i has an explicit dependence on θ, i.e. $f_i(x_i^*(\theta), \theta)$, but we use our simpler formulation for ease of presentation.

Algorithm 1 IFT+CG to compute gradient estimate $h_{\varepsilon,\delta}$

Require: tolerances $\varepsilon, \delta > 0$

1: Find an approximate solution x_ε satisfying (2.6).

2: Using CG, find $q_{\varepsilon,\delta}$ satisfying

$$\|A(x_\varepsilon)q_{\varepsilon,\delta} - \nabla f(x_\varepsilon)\| \leq \delta. \tag{2.8}$$

3: Compute gradient estimate $h_{\varepsilon,\delta} := -B(x_\varepsilon)^T q_{\varepsilon,\delta}$.

the fact that both computations cannot or should not be carried out accurately. Particularly, we focus on methods which are independent of the specific algorithm for solving the lower-level problem (1.1b), and so instead we assume that we have an approximate minimizer x_ε of (1.1b), such that

$$\|x_\varepsilon - x^*\| \leq \varepsilon. \tag{2.6}$$

Finally, we will use the following smoothness assumptions for the results below.

Assumption 2 *A (resp. B) is L_A- (resp. L_B-) Lipschitz continuous in y, uniformly for all θ.*

Assumption 3 *∇f is Lipschitz continuous with constant $L_{\nabla f}$.*

2.1. *Implicit Function Theorem Approach*

Given (2.6) and (2.5), a natural approximation is

$$h^* \approx h_\varepsilon := -B(x_\varepsilon)^T A(x_\varepsilon)^{-1} \nabla f(x_\varepsilon). \tag{2.7}$$

For large-scale problems, full matrix inversion/linear solves is impractical and so iterative methods are preferred. Given that $A(x_\varepsilon)$ is symmetric positive definite by Assumption 1, we consider using the conjugate gradient method (CG). The symmetry of A gives two possible approaches:

- Approximately solve $A(x_\varepsilon)^{-1}\nabla f(x_\varepsilon)$ and pre-multiply by $-B(x_\varepsilon)^T$
- Approximately solve $B(x_\varepsilon)^T A(x_\varepsilon)^{-1} = [A(x_\varepsilon)^{-1}B(x_\varepsilon)]^T$ and post-multiply by $\nabla f(x_\varepsilon)$.

Of these, the former is preferable as the linear solve has only one right-hand side, reducing the total number of matrix-vector products. This leads to a natural algorithm to compute the hypergradient, given in Algorithm 1, which we refer to as IFT+CG, relating to its association with the implicit function theorem.

The first analysis of IFT+CG in the context of bilevel optimization was in [17], which gives the following.

Theorem 4 (Theorem 1, [17]) *Suppose Assumptions 1, 2 and 3 hold, and ε is sufficiently small*[2]*. Then the error produced by IFT+CG with $\delta = \varepsilon$ satisfies $\|h_{\varepsilon,\varepsilon} - h^*\| = \mathcal{O}(\varepsilon)$.*

[2] Specifically, the proof requires that $\sum_k \varepsilon_k < \infty$ and we take k sufficiently large, where ε_k is the value of ε chosen in iteration k of the bilevel optimization.

This result was strengthened in the recent work [18], which considers IFT but with any method for solving (2.8), not just CG.

Theorem 5 (Theorem 3.1.2, [18]) *Suppose Assumptions 1, 2 and 3 hold, and* $\varepsilon < \frac{\mu}{2\max\{L_A, L_B\}}$. *Then the error produced by IFT+CG satisfies* $\|h_{\varepsilon,\delta} - h^*\| \leq C(\varepsilon + \delta)$ *for some constant* C.

Our analysis in Section 4 improves on Theorems 4 and 5 by removing the restriction on ε and making all constants explicit.

2.2. *Inexact Automatic Differentiation*

We now consider an alternative approach for hypergradient estimation based on automatic differentiation (backpropagation), as developed in [19].

The motivation is to consider solving the lower-level problem to find x_ε (2.6) by applying a first-order method to (1.1b) such as gradient descent

$$x^{(k+1)} = x^{(k)} - \alpha \nabla g(x^{(k)}), \tag{2.9}$$

or Polyak's heavy ball momentum

$$x^{(k+1)} = x^{(k)} - \alpha \nabla g(x^{(k)}) + \beta(x^{(k)} - x^{(k-1)}). \tag{2.10}$$

If we suppose that K iterations of either (2.9) or (2.10) are run from a fixed starting point $x^{(0)}$, then reverse-mode automatic differentiation (AD) with respect to θ applied to (2.10) gives: initialize $\widetilde{x}^{(K)} := \nabla f(x^{(K)})$, $\widetilde{x}^{(K+1)} := 0 \in \mathbb{R}^d$ and $h^{(0)} := 0 \in \mathbb{R}^n$, then iterate

$$h^{(k+1)} = h^{(k)} - \alpha B(x^{(K-k-1)})^T \widetilde{x}^{(K-k)}, \tag{2.11a}$$

$$\widetilde{x}^{(K-k-1)} = \widetilde{x}^{(K-k)} - \alpha A(x^{(K-k-1)})\widetilde{x}^{(K-k)} + \beta(\widetilde{x}^{(K-k)} - \widetilde{x}^{(K-k+1)}), \tag{2.11b}$$

for $k = 0, \ldots, K-1$. For gradient descent (2.9), the same iteration (2.11) holds but with $\beta = 0$. In both cases, the final gradient estimator is $h^{(K)}$.

The key insight of [19] is that (2.11) can be made to converge faster[3] (and with fewer Hessian/Jacobian evaluations) by replacing $B(x^{(K-k)-1})$ and $A(x^{(K-k)-1})$ with the final Jacobian and Hessian, $B(x^{(K)})$ and $A(x^{(K)})$ for all k. This motivates the inexact automatic differentiation (IAD) methods IAD+GD and IAD+HB given in Algorithms 2 and 3 respectively.

The main result from [19] is the following.

Theorem 6 (Propositions 10 & 17, [19]) *Suppose Assumptions 1 and 2 hold. Further assume that* $\|B(y)\| \leq B_{\max}$ *for each* y.[4] *Then:*

- *If* $\alpha \leq 1/L$, *then there exists* $\lambda_{GD} \in [0,1)$ *such that IAD+GD (fixed K) gives a hypergradient estimate* $h^{(K)}$ *satisfying*

$$\|h^{(K)} - D(x^{(K)})\nabla f(x^{(K)})\| \leq \lambda_{GD}^K \frac{B_{\max}}{\mu} \|\nabla f(x^{(K)})\|. \tag{2.14}$$

  The optimal rate[5] $\lambda_{GD}^* = (L-\mu)/(L+\mu)$ *is attained if* $\alpha = 2/(L+\mu)$.

[3] Linearly with an improved rate.
[4] Technically, [19] also requires that $B_{\max}$ does not depend on θ and that the bound holds uniformly for all θ.
[5] Here, L and μ denote $L(\theta)$ and $\mu(\theta)$, c.f. Remark 1.

Algorithm 2 IAD+GD (fixed K) to compute gradient estimate $h^{(K)}$

Require: iteration count K and stepsize α ▷ (note: this represents a code comment)

1: run K iterations of (2.9) to get $x^{(K)} \approx x^*$.
2: initialize $\widetilde{x}^{(0)} := \nabla f(x^{(K)})$ and $h^{(0)} = 0 \in \mathbb{R}^n$
3: **for** $k = 0, \ldots, K-1$ ▷ K iterations of *inexact* reverse-mode AD

$$h^{(k+1)} = h^{(k)} - \alpha B(x^{(K)})^T \widetilde{x}^{(k)}, \tag{2.12a}$$

$$\widetilde{x}^{(k+1)} = \widetilde{x}^{(k)} - \alpha A(x^{(K)}) \widetilde{x}^{(k)}. \tag{2.12b}$$

Algorithm 3 IAD+HB (fixed K) to compute gradient estimate $h^{(K)}$

Require: iteration count K, stepsize α, and momentum parameter β

1: run K iterations of (2.10) to get $x^{(K)} \approx x^*$.
2: initialize $\widetilde{x}^{(0)} := \nabla f(x^{(K)})$, $\widetilde{x}^{(-1)} = 0 \in \mathbb{R}^d$ and $h^{(0)} = 0 \in \mathbb{R}^n$
3: **for** $k = 0, \ldots, K-1$ ▷ K iterations of *inexact* reverse-mode AD

$$h^{(k+1)} = h^{(k)} - \alpha B(x^{(K)})^T \widetilde{x}^{(k)}, \tag{2.13a}$$

$$\widetilde{x}^{(k+1)} = \widetilde{x}^{(k)} - \alpha A(x^{(K)}) \widetilde{x}^{(k)} + \beta(\widetilde{x}^{(k)} - \widetilde{x}^{(k-1)}). \tag{2.13b}$$

- *If $\beta \in [0,1)$, $\alpha \leq 2(1+\beta)/L$ and $\gamma > 0$, then there exists $\lambda_{HB} \in [0,1)$ such that IAD+HB (fixed K) gives a hypergradient estimate H_K satisfying*

$$\|h^{(K)} - D(x^{(K)})\nabla f(x^{(K)})\| \leq c(\lambda_{HB} + \gamma)^K \frac{B_{\max}}{\mu} \|\nabla f(x^{(K)})\|, \tag{2.15}$$

for some constant $c > 0$. The optimal rate $\lambda^*_{HB} = (\sqrt{L} - \sqrt{\mu})/(\sqrt{L} + \sqrt{\mu})$ is attained if $\alpha = 4/(\sqrt{L} + \sqrt{\mu})^2$ and $\beta = (\lambda^*_{HB})^2$.

By comparison, the exact AD iterations (2.11) are shown to have sub-optimal linear convergence rates of size $\mathcal{O}(K\lambda^K)$ [19, Propositions 8 & 15].

We note that in their formulation as stated both IAD+GD (fixed K) and IAD+HB (fixed K) require one Hessian-vector and one Jacobian-vector product at each iteration of the hypergradient calculation (step 2). By contrast, IFT+CG only requires one Jacobian-vector product at the end of calculation, rather than one per iteration (but still one Hessian-vector product per iteration). This additional computational cost can be alleviated by reformulating the algorithms, e.g., as in Section 3.

In the next section we show that these inexact AD methods are actually variants of the IFT approach, and provide a unified error analysis of all three methods.

3. Unified Hypergradient Computation Algorithm

We revisit the inexact AD framework to realize that it is actually an approximate IFT method with specific algorithmic choices. Thus, at the end of this section we propose a unified framework that encompasses all methods in Section 2.

The next theorems make said observation for inexact AD with GD and HB.

Theorem 7 *Let $x^{(K)}$ denote the output of GD (2.9) after K iterations and let $\Phi(x) = \frac{1}{2}x^T A(x^{(K)})x - \nabla f(x^{(K)})^T x$. Then the gradient estimate $h^{(K)}$ after K iterations of the inexact AD iterations (2.12) can also be computed as $h^{(K)} = -B(x^{(K)})^T q^{(K)}$ with $q^{(0)} = 0$ and*

$$q^{(k+1)} = q^{(k)} - \alpha \nabla \Phi(q^{(k)}), \quad k = 0, \dots, K-1. \tag{3.1}$$

Proof Notice that the iteration (2.12) can be written as $h^{(K)} = -\alpha B(x^{(K)})^T \sum_{k=0}^{K-1} \tilde{x}^{(k)}$. Hence we want to show that the new iteration (3.1) satisfies $q^{(K)} = \alpha \sum_{k=0}^{K-1} \tilde{x}^{(k)}$ which we prove by induction where we will frequently use $\nabla \Phi(q) = Aq - \nabla f(x^{(K)})$ and $\tilde{x}^{(0)} = \nabla f(x^{(K)})$.

For $K = 0$, using the initial condition and $\nabla \Phi(q^{(0)}) = -\nabla f(x^{(K)})$, (3.1) implies

$$q^{(1)} = q^{(0)} - \alpha \nabla \Phi(q^{(0)}) = \alpha \nabla f(x^{(K)}) = \alpha \tilde{x}^{(0)}.$$

Now, let the assertion be true for $K-1$. Due to the initial condition, an alternative way to write the iterations (2.12) is

$$\tilde{x}^{(K-1)} = \tilde{x}^{(0)} - \sum_{k=0}^{K-2} \alpha A \tilde{x}^{(k)}.$$

Thus, with the induction hypothesis

$$\tilde{x}^{(K-1)} = \nabla f(x^{(K)}) - A q^{(K-1)} = -\nabla \Phi(q^{(K-1)})$$

and therefore

$$q^{(K)} = q^{(K-1)} - \alpha \nabla \Phi(q^{(K-1)}) = \alpha \sum_{k=0}^{K-2} \tilde{x}^{(k)} + \alpha \tilde{x}^{(K-1)} = \alpha \sum_{k=0}^{K-1} \tilde{x}^{(k)}. \qquad \square$$

A similar observation can be made for HB.

Theorem 8 *Let $x^{(K)}$ denote the output of HB (2.10) after K iterations and let $\Phi(x) = \frac{1}{2}x^T A(x^{(K)})x - \nabla f(x^{(K)})^T x$. Then the gradient estimate $h^{(K)}$ after K iterations of the inexact AD iterations (2.13) can also be computed as $h^{(K)} = -B(x^{(K)})^T q^{(K)}$ with $q^{(0)} = q^{(-1)} = 0$ and*

$$q^{(k+1)} = q^{(k)} - \alpha \nabla \Phi(q^{(k)}) + \beta (q^{(k)} - q^{(k-1)}), \qquad k = 0, 1, \dots, K-1. \tag{3.2}$$

Proof The proof is similar to the proof of Theorem 7. As before, we notice that the iteration (2.13) can be written as $h^{(K)} = -\alpha \sum_{k=0}^{K-1} B(x^{(K)})^T \tilde{x}^{(k)}$. Hence we want to show that the new iteration (3.2) satisfies $q^{(K)} = \alpha \sum_{k=0}^{K-1} \tilde{x}^{(k)}$ which we prove by induction.

For $K = 0$, using the initial conditions and $\nabla\Phi(q^{(0)}) = -\nabla f(x^{(K)})$, (3.2) implies

$$q^{(1)} = q^{(0)} - \alpha\nabla\Phi(q^{(0)}) + \beta(q^{(0)} - q^{(-1)}) = \alpha\nabla f(x^{(K)}) = \alpha\tilde{x}^{(0)}.$$

Similarly for $K = 1$,

$$\begin{aligned} q^{(2)} &= q^{(1)} - \alpha\nabla\Phi(q^{(1)}) + \beta(q^{(1)} - q^{(0)}) = \alpha\tilde{x}^{(0)} - \alpha(A(\alpha\tilde{x}^{(0)}) - \nabla f(x^{(K)})) + \beta\alpha\tilde{x}^{(0)} \\ &= \alpha\tilde{x}^{(0)} + \alpha\left(\tilde{x}^{(0)} - \alpha A\tilde{x}^{(0)} + \beta(\tilde{x}^{(0)} - \tilde{x}^{(-1)})\right) = \alpha\tilde{x}^{(0)} + \alpha\tilde{x}^{(1)}. \end{aligned}$$

Now, let the assertion be true for $K-1$ and $K-2$. Due to the initial conditions, an alternative way to write the iterations (2.13b) is

$$\tilde{x}^{(K-1)} = \tilde{x}^{(0)} - \sum_{k=0}^{K-2} \alpha A\tilde{x}^{(k)} + \beta\tilde{x}^{(K-2)},$$

Thus,

$$\begin{aligned} q^{(K)} &= q^{(K-1)} - \alpha\nabla\Phi(q^{(K-1)}) + \beta(q^{(K-1)} - q^{(K-2)}) \\ &= \alpha\sum_{k=0}^{K-2}\tilde{x}^{(k)} - \alpha(A(\alpha\sum_{k=0}^{K-2}\tilde{x}^{(k)}) - \nabla f(x^{(K)})) + \beta\alpha\tilde{x}^{(K-2)} \\ &= \alpha\sum_{k=0}^{K-2}\tilde{x}^{(k)} + \alpha\left(-\sum_{k=0}^{K-2}\alpha A\tilde{x}^{(k)} + \tilde{x}^{(0)} + \beta\tilde{x}^{(K-2)}\right) = \alpha\sum_{k=0}^{K-2}\tilde{x}^{(k)} + \alpha\tilde{x}^{(K-1)} = \alpha\sum_{k=0}^{K-1}\tilde{x}^{(k)}. \end{aligned}$$ □

The analysis in Theorem 7 and 8 implies that $q^{(K)}$ approximately solves $A(x^{(K)})q = \nabla f(x^{(K)})$. Thus, inexact AD is a special case of IFT with GD/HB as the lower-level solver and GD/HB to solve the system of linear equations for the upper-level hypergradient estimate. This observation motivates us to define a generalised IFT algorithm which captures both approaches outlined in Section 2.

Algorithm 4 IFT to compute gradient estimate $\widetilde{h}$

1: Find an approximate solution $\tilde{x}$ of the lower-level problem using any method of choice with any number of iterations or accuracy.
2: Find an approximate solution $\tilde{q}$ solving $A(\tilde{x})q = \nabla f(\tilde{x})$ using any method of choice with any number of iterations or accuracy.
3: Return the gradient estimate $\tilde{h} := -B(\tilde{x})^T\tilde{q}$.

Remark 2 *If we choose a method in step 1 to find a solution with ε accuracy and CG in step 2 to find a solution with residual accuracy δ, then we recover Algorithm 1 as proposed in [17, 18]. If we choose GD/HB in step 1 for K iterations and GD/HB in step 2 for K iterations, then we get Algorithms 2 and 3 as proposed in [19]. However, going forward both strategies can also be combined given more flexibility to the user.*

4. A priori and a posteriori error analysis

We now derive new a priori and a posteriori error bounds for hypergradients as computed in Algorithm 4. The a priori bound shows explicit convergence results that are independent of the lower-level solution estimate $\widetilde{x}$. By contrast, the a posteriori bound gives an estimate on the error that is computable (in the sense that no knowledge of x^* is required). Note also, that in contrast to [17, 18] (see also Theorem 4 and 5), we do not make any assumption on the size of the errors, nor that these estimates are part of a bilevel programming scheme.

Theorem 9 (A posteriori bound) *Suppose Assumptions 1, 2 and 3 hold. Let $\widetilde{x}, \widetilde{q}$ and $\widetilde{h}$ be the output of Algorithm 4. Let $\widetilde{\varepsilon} := \|\nabla g(\widetilde{x})\|/\mu$ and $\widetilde{\delta} := \|A(\widetilde{x})\widetilde{q} - \nabla f(\widetilde{x})\|$. Note $\|\widetilde{x} - x^*\| \leq \widetilde{\varepsilon}$. Moreover, we define*

$$c(x) := \frac{L_{\nabla f}\|B(x)\|}{\mu} + L_{A^{-1}}\|\nabla f(x)\|\|B(x)\| + \frac{L_B\|\nabla f(x)\|}{\mu},$$

where $L_{A^{-1}}$ is the Lipschitz constant for $A(x)^{-1}$, which exists by Lemma 14 below. Then, the following a posteriori bound holds:

$$\|\widetilde{h} - h^*\| \leq c(\widetilde{x})\widetilde{\varepsilon} + \frac{\|B(\widetilde{x})\|}{\mu}\widetilde{\delta} + \frac{L_B L_{\nabla f}}{\mu}\widetilde{\varepsilon}^2.$$

Theorem 10 (A priori bound) *Suppose Assumptions 1, 2 and 3 hold. Let $\widetilde{x}, \widetilde{q}$ and $\widetilde{h}$ be the output of Algorithm 4, computed such that $\|\widetilde{x} - x^*\| \leq \varepsilon$ and $\|A(\widetilde{x})\widetilde{q} - \nabla f(\widetilde{x})\| \leq \delta$. Then, with c as defined in Theorem 9 the following a priori bound holds:*

$$\|\widetilde{h} - h^*\| \leq c(x^*)\varepsilon + \frac{\|B(x^*)\|}{\mu}\delta + \frac{L_B L_{\nabla f}}{\mu}\varepsilon^2 + \frac{L_B}{\mu}\delta\varepsilon.$$

In particular, $\|\widetilde{h} - h^*\| = \mathcal{O}(\varepsilon + \delta + \varepsilon^2 + \delta^2)$ and $\|\widetilde{h} - h^*\| \to 0$ as $\varepsilon, \delta \to 0$.

4.1. *Proofs of Theorems 9 and 10*

Lemma 11 *Assume that $\|A(\widetilde{x})^{-1}\| \leq \mu^{-1}$ for any $\widetilde{x}$. Then, for any $\widetilde{x}$ and $\widetilde{q}$ it holds that*

$$\|B(\widetilde{x})^T\widetilde{q} - B(\widetilde{x})^T A(\widetilde{x})^{-1}\nabla f(\widetilde{x})\| \leq \frac{\|B(\widetilde{x})\|}{\mu}\|A(\widetilde{x})\widetilde{q} - \nabla f(\widetilde{x})\|.$$

Proof A direct calculation and multiplying with $A(\widetilde{x})^{-1}A(\widetilde{x})$ it holds that

$$\begin{aligned}\|B(\widetilde{x})^T\widetilde{q} - B(\widetilde{x})^T A(\widetilde{x})^{-1}\nabla f(\widetilde{x})\| &\leq \|B(\widetilde{x})\|\|\widetilde{q} - A(\widetilde{x})^{-1}\nabla f(\widetilde{x})\| \\ &= \|B(\widetilde{x})\|\|A(\widetilde{x})^{-1}(A(\widetilde{x})\widetilde{q} - \nabla f(\widetilde{x}))\| \\ &\leq \|B(\widetilde{x})\|\|A(\widetilde{x})^{-1}\|\|A(\widetilde{x})\widetilde{q} - \nabla f(\widetilde{x})\|\end{aligned}$$

The assertion then follows from $\|A(\widetilde{x})^{-1}\| \leq \mu^{-1}$. □

Lemma 12 *Let A^{-1} and B be Lipschitz continuous with constants $L_{A^{-1}}$ and L_B, respectively, and $\|A(x)^{-1}\| \leq \mu$ for any x. Then, the mapping D is locally Lipschitz continuous. Specifically, for any x_1, x_2 and $L_D(x) := \|B(x)\|L_{A^{-1}} + L_B/\mu$ it holds that*

$$\|D(x_1) - D(x_2)\| \leq L_D(x_1)\|x_1 - x_2\|.$$

Proof Adding and subtracting $B(x_1)^T A(x_2)^{-1}$ and the triangle inequality yields

$$\begin{aligned}\|D(x_1) - D(x_2)\| &= \|B(x_1)^T A(x_1)^{-1} - B(x_2)^T A^{-1}(x_2)\| \\ &\leq \|B(x_1)\|\|A(x_1)^{-1} - A(x_2)^{-1}\| + \|A(x_2)^{-1}\|\|B(x_1) - B(x_2)\|\end{aligned}$$

Then the result follows from Lipschitz continuity as well as the estimate $\|A(x_2)^{-1}\| \leq \mu$. □

Lemma 13 *Let the assumptions of Lemma 12 hold and let ∇f be Lipschitz continuous with constant $L_{\nabla f}$. Let c be as defined in Theorem 9. Then, for any x_1, x_2 it holds that*

$$\|D(x_1)\nabla f(x_1) - D(x_2)\nabla f(x_2)\| \leq c(x_1)\|x_1 - x_2\| + \frac{L_B L_{\nabla f}}{\mu}\|x_1 - x_2\|^2.$$

Proof We add and subtract $D(x_2)\nabla f(x_1)$ and use the triangle inequality to yield

$$\|D(x_1)\nabla f(x_1) - D(x_2)\nabla f(x_2)\| \leq \|\nabla f(x_1)\|\|D(x_1) - D(x_2)\| + \|D(x_2)\|\|\nabla f(x_1) - \nabla f(x_2)\|.$$

Notice further that

$$\|D(x_2)\| = \|B(x_2)^T A(x_2)^{-1}\| \leq \|B(x_2)\|\|A(x_2)^{-1}\| \leq \frac{\|B(x_2)\|}{\mu}.$$

Combining both inequalities and invoking Lemma 12 leads to

$$\begin{aligned}\|D(x_1)\nabla f(x_1) - D(x_2)\nabla f(x_2)\| &\leq \|\nabla f(x_1)\|\|D(x_1) - D(x_2)\| + \|D(x_2)\|\|\nabla f(x_1) - \nabla f(x_2)\| \\ &\leq \|\nabla f(x_1)\|L_D(x_1)\|x_1 - x_2\| + \frac{\|B(x_2)\|}{\mu}L_{\nabla f}\|x_1 - x_2\| \\ &\leq \frac{\|B(x_1)\|}{\mu}L_{\nabla f}\|x_1 - x_2\| + \frac{L_B}{\mu}L_{\nabla f}\|x_1 - x_2\|^2 + \|\nabla f(x_1)\|L_D(x_1)\|x_1 - x_2\|,\end{aligned}$$

where the last inequality follows from the Lipschitz continuity of B,

$$\|B(x_2)\| \leq \|B(x_1)\| + \|B(x_1) - B(x_2)\| \leq \|B(x_1)\| + L_B\|x_1 - x_2\|. \quad \square$$

The previous two Lemmas needed the Lipschitz continuity of A^{-1}. We show next that this follows directly from our assumptions on A.

Lemma 14 *Let A be Lipschitz continuous with constant L_A and $\|A(x)^{-1}\| \leq \mu$ for any x. Then, the mapping A^{-1} is Lipschitz continuous with constant $L_{A^{-1}} \leq L_A/\mu^2$.*

Proof Straightforward calculations lead to

$$\|A(x)^{-1} - A(y)^{-1}\| = \|A(x)^{-1}(A(y) - A(x))A(y)^{-1}\| \leq \|A(x)^{-1}\|\|A(y) - A(x)\|\|A(y)^{-1}\|$$

and the assertion follows directly from the assumptions on A. □

We are now able to prove our main results.

Proof of Theorem 9 We start bounding the error in the hypergradient using Lemma 11.

$$\begin{aligned}\|\widetilde{h} - h\| &= \|B(\widetilde{x})^T \widetilde{q} - D(x^*)\nabla f(x^*)\| \\ &\leq \|B(\widetilde{x})^T \widetilde{q} - B(\widetilde{x})^T A(\widetilde{x})^{-1}\nabla f(\widetilde{x})\| + \|D(\widetilde{x})\nabla f(\widetilde{x}) - D(x^*)\nabla f(\widetilde{x}^*)\| \\ &\leq \frac{\|B(\widetilde{x})\|}{\mu}\|A(\widetilde{x})\widetilde{q} - \nabla f(\widetilde{x})\| + \|D(\widetilde{x})\nabla f(\widetilde{x}) - D(x^*)\nabla f(\widetilde{x}^*)\| \end{aligned} \tag{4.1}$$

For the a posteriori bound, we invoke Lemma 13 with $x_1 = \widetilde{x}$ and $x_2 = x^*$ and apply it to (4.1),

$$\begin{aligned}\|\widetilde{h} - h\| &\leq \frac{\|B(\widetilde{x})\|}{\mu}\|A(\widetilde{x})\widetilde{q} - \nabla f(\widetilde{x})\| + \|D(\widetilde{x})\nabla f(\widetilde{x}) - D(x^*)\nabla f(\widetilde{x}^*)\| \\ &\leq \frac{\|B(\widetilde{x})\|}{\mu}\|A(\widetilde{x})\widetilde{q} - \nabla f(\widetilde{x})\| + c(\widetilde{x})\|\widetilde{x} - x^*\| + \frac{L_B L_{\nabla f}}{\mu}\|\widetilde{x} - x^*\|^2.\end{aligned}$$

Then using the notation $\widetilde{\varepsilon} = \|\nabla g(\widetilde{x})\|/\mu$ and $\widetilde{\delta} = \|A(\widetilde{x})\widetilde{q} - \nabla f(\widetilde{x})\|$ we arrive at the assertion,

$$\|\widetilde{h} - h\| \leq \frac{\|B(\widetilde{x})\|}{\mu}\widetilde{\delta} + c(\widetilde{x})\widetilde{\varepsilon} + \frac{L_B L_{\nabla f}}{\mu}\widetilde{\varepsilon}^2. \qquad \square$$

Proof of Theorem 10 As in the proof of Theorem 9 we use Lemma 11 to get (4.1). For the a priori bound, we then invoke Lemma 13 with $x_1 = x^*$ and $x_2 = \widetilde{x}$ and apply it to (4.1),

$$\begin{aligned}\|\widetilde{h} - h\| &\leq \frac{\|B(\widetilde{x})\|}{\mu}\|A(\widetilde{x})\widetilde{q} - \nabla f(\widetilde{x})\| + \|D(\widetilde{x})\nabla f(\widetilde{x}) - D(x^*)\nabla f(\widetilde{x}^*)\| \\ &\leq \frac{\|B(\widetilde{x})\|}{\mu}\|A(\widetilde{x})\widetilde{q} - \nabla f(\widetilde{x})\| + c(x^*)\|\widetilde{x} - x^*\| + \frac{L_B L_{\nabla f}}{\mu}\|\widetilde{x} - x^*\|^2.\end{aligned}$$

Using the a priori estimates $\|A(\widetilde{x})\widetilde{q} - \nabla f(\widetilde{x})\| \leq \delta$ and $\|\widetilde{x} - x^*\| \leq \varepsilon$ together with the Lipschitz continuity of B yields

$$\begin{aligned}\|\widetilde{h} - h\| &\leq \frac{\|B(\widetilde{x})\|}{\mu}\|A(\widetilde{x})\widetilde{q} - \nabla f(\widetilde{x})\| + c(x^*)\|\widetilde{x} - x^*\| + \frac{L_B L_{\nabla f}}{\mu}\|\widetilde{x} - x^*\|^2 \\ &\leq \frac{\|B(\widetilde{x})\|}{\mu}\delta + c(x^*)\varepsilon + \frac{L_B L_{\nabla f}}{\mu}\varepsilon^2 \leq \frac{\|B(x^*)\|}{\mu}\delta + \frac{L_B}{\mu}\delta\varepsilon + c(x^*)\varepsilon + \frac{L_B L_{\nabla f}}{\mu}\varepsilon^2. \qquad \square\end{aligned}$$

4.2. *Specialized a priori bounds*

The above framework is generic in that no specific algorithms are required for the lower-level solver and linear system solver. Our a posteriori bounds are completely solver independent, because they use $\|\nabla g(\widetilde{x})\|$ and $\|A(\widetilde{x})\widetilde{q} - \nabla f(\widetilde{x})\|$ as the key error metrics, which are always available from the solver. However our a priori bounds are solver-dependent, based on their specific convergence rates. We now give some concrete examples of the a priori bounds for specific solver choices.

For the lower-level solver, we require $\|\widetilde{x} - x^*\| \leq \varepsilon$. In terms of the iteration count k, for gradient descent (2.9) with the optimal stepsize $\alpha = 2/(L+\mu)$ we have [19, Lemma 6]

$$\|x^{(k)} - x^*\| \leq (\lambda^*_{\mathrm{GD}})^k \|x^{(0)} - x^*\|, \tag{4.2}$$

for all k, where $\lambda^*_{\mathrm{GD}} = (L-\mu)/(L+\mu)$. Alternatively, if we use heavy ball (2.10) with the optimal stepsize $\alpha = 4/(\sqrt{L}+\sqrt{\mu})^2$ and momentum $\beta = (\lambda^*_{\mathrm{HB}})^2$, where $\lambda^*_{\mathrm{HB}} := (\sqrt{L}-\sqrt{\mu})/(\sqrt{L}+\sqrt{\mu})$, we have the following [19, Lemma 13]: for all $\gamma > 0$, there exists $c > 0$ such that

$$\|x^{(k)} - x^*\| \leq c(\lambda^*_{\mathrm{HB}} + \gamma)^k \|x^{(0)} - x^*\|, \tag{4.3}$$

for all k. As a final example for the lower-level solver, we consider FISTA adapted for strongly convex problems, [32, Algorithm 5]. Combining [32, Theorem 4.10] with the identity $\|x - x^*\|^2 \leq (2/\mu)[g(x) - g(x^*)]$ (e.g., [33, Theorem 2.1.7]) gives

$$\|x^{(k)} - x^*\|^2 \leq \min\left\{ \left(1 + \frac{\sqrt{\mu}}{\sqrt{L}}\right)(\lambda^*_{\mathrm{FISTA}})^k, \frac{4}{(k+1)^2} \right\} \frac{L}{\mu} \|x^{(0)} - x^*\|^2, \tag{4.4}$$

where $\lambda^*_{\mathrm{FISTA}} := 1 - \sqrt{\mu/L}$. Of these results, although heavy ball and FISTA both have an accelerated linear rate compared to gradient descent, we do have $\lambda^*_{\mathrm{FISTA}} > \lambda^*_{\mathrm{HB}}$, with a larger difference for well-conditioned problems.

For the linear system solver, our goal is to make $\|A(\widetilde{x})q - \nabla f(\widetilde{x})\|$ small by minimizing $\Phi(q) = \frac{1}{2}q^T A(\widetilde{x})q - \nabla f(\widetilde{x})^T q$. We note that $A(\widetilde{x})q - \nabla f(\widetilde{x}) = \nabla\Phi(q)$, and Φ is μ-strongly convex and has L-Lipschitz gradients. We can combine the above lower-level solver results with the identity $\mu\|q^{(k)} - q^*\| \leq \|A(\widetilde{x})q - \nabla f(\widetilde{x})\| \leq L\|q^{(k)} - q^*\|$ where $q^* = A(\widetilde{x})^{-1}\nabla f(\widetilde{x})$, and if we take $q^{(0)} = 0$ then the initial residual is $\|\nabla f(\widetilde{x})\|$, we have

$$\|A(\widetilde{x})q^{(k)} - \nabla f(\widetilde{x})\| \leq \frac{L}{\mu}(\lambda^*_{\mathrm{GD}})^k \|\nabla f(\widetilde{x})\|, \tag{4.5}$$

for gradient descent, and for heavy ball: for all $\gamma > 0$, there exists $c > 0$ such that

$$\|A(\widetilde{x})q^{(k)} - \nabla f(\widetilde{x})\| \leq c\frac{L}{\mu}(\lambda^*_{\mathrm{HB}} + \gamma)^k \|\nabla f(\widetilde{x})\|. \tag{4.6}$$

Lastly, we consider the a priori convergence rate of CG. Combining the standard linear convergence rate in $\|q^{(k)} - q^*\|_A$ (e.g., [34, eq. (5.36)]) with the Rayleigh quotient inequalities $\mu\|y\|^2 \leq \|y\|_A^2 \leq L\|y\|^2$ we get the rate

$$\|A(\widetilde{x})q^{(k)} - \nabla f(\widetilde{x})\| \leq 2\frac{L^{3/2}}{\mu^{3/2}}(\lambda^*_{\mathrm{HB}})^k \|\nabla f(\widetilde{x})\|, \tag{4.7}$$

and we recover the same accelerated linear rate as heavy ball momentum. These results cover the three motivating methods described in Section 2.

We specifically note that for the linear solve step, heavy ball has the same accelerated rate as CG, but with an unknown constant, so CG is to be preferred even without considering the extra convergence theory available for CG (e.g., finite termination in exact arithmetic).

5. Numerical results

We now present numerical comparisons of the different hypergradient estimation methods. Our results have three components: in Section 5.1 we compare the quality of the a priori and a posteriori error bounds, in Section 5.2 we show how the choice of hypergradient estimation method impacts the quality of the overall optimization, and lastly in Section 5.3 we demonstrate the utility of our approach for learning high-quality neural network regularizers for image denoising.

5.1. *Quality of error bounds*

We first use a simple example problem to compare the quality of the a priori and a posteriori bounds derived in Section 4. The example problem is a simple linear least-squares problem taken from [35, Section 6.1]:

$$\min_{\theta\in\mathbb{R}^{10}} F(\theta) := \|A_1 x^*(\theta) - b_1\|_2^2, \tag{5.1a}$$

$$\text{s.t. } x^*(\theta) := \arg\min_{x\in\mathbb{R}^{10}} \|A_2 x + A_3\theta - b_2\|_2^2, \tag{5.1b}$$

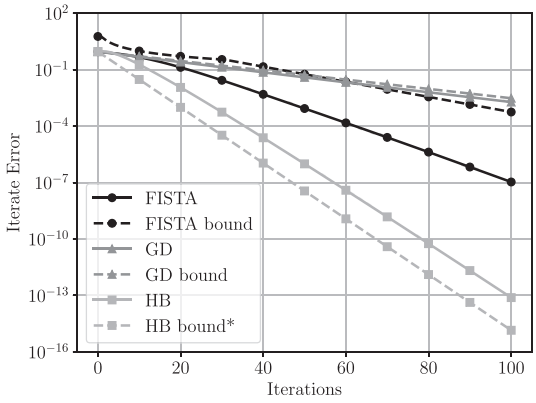
where $A_i \in \mathbb{R}^{1000\times 10}$ have random i.i.d. entries from $\text{Unif}([0,1])$, and $b_i \in \mathbb{R}^{1000}$ are given by $b_1 = A_1\hat{x}_1 + 0.01y_1$ and $b_2 = A_2\hat{x}_2 + A_3\widetilde{\theta} + 0.01y_2$ where $\hat{x}_1$, $\hat{x}_2$ and $\widetilde{\theta} \in \mathbb{R}^{10}$ have i.i.d. $\text{Unif}([0,1])$ entries and $y_1, y_2 \in \mathbb{R}^{1000}$ are independent standard Gaussian vectors. For our experiments we pick the test evaluation point θ to be the vector of all ones.

Because of the simple structure of this problem, it is easy to compute $x^*(\theta)$ analytically and get all requisite Lipschitz constants. Hence we can explicitly compute the true hypergradient $\nabla F(\theta)$ and all error bounds explicitly. The only exception is the a priori bound for HB/IAD+HB, which has the unknown constants c and γ in (4.3) and (4.6). For illustration, we choose $c = 1$ and $\gamma = 0$ but there is no guarantee that this will give a true bound and such results are denoted with an asterisk in the figures below.

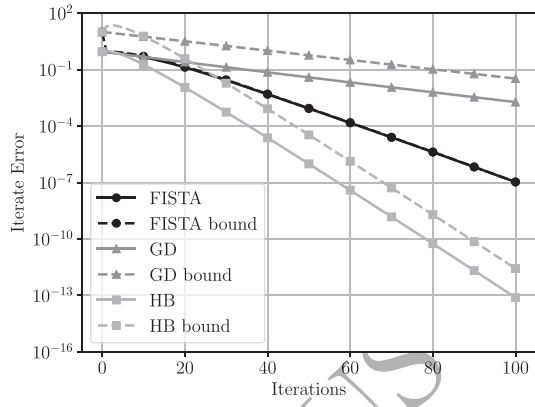
In our results, we compare the three different lower-level solvers discussed in Section 4.2: GD (2.9), HB (2.10) and FISTA (adapted for strongly convex problems as per [32, Algorithm 5]), all with optimal stepsize and momentum parameters. We also use the three hypergradient methods discussed in Section 4.2, namely CG, GD and HB. We run the lower-level solvers for up to 100 iterations to get $\widetilde{x} \approx x^*$ (except for Figure 1 where $\widetilde{x} = x^*$ is used), and the hypergradient solvers for up to 200 iterations.

Firstly, Figures 1(a,b) show the a priori and a posteriori bounds on the lower-level solvers, where the a priori bounds are from the standard linear convergence rates for the lower-level solvers (i.e. (4.2), (4.3) and (4.4) for GD, HB and FISTA respectively) and the a posteriori results use $\|\widetilde{x} - x^*\| \le \|\nabla g(\widetilde{x})\|/\mu$. As in [15, Figure 3], we find that the a posteriori bounds are much tighter for FISTA (and HB given that the a priori bounds are uncomputable), although the a priori bounds are better for the slowest method, GD.

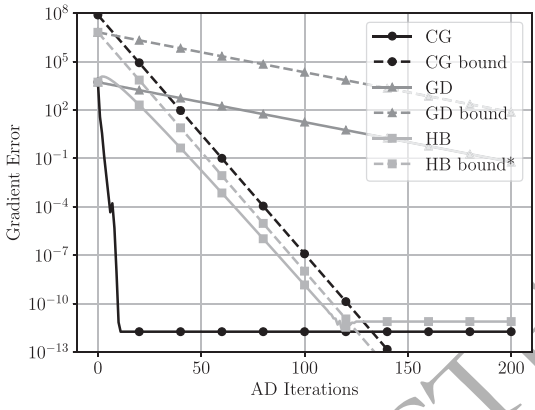
Figures 1(c,d) then show the a priori and a posteriori bounds on the hypergradient estimates, using the exact value $\widetilde{x} = x^*$ (i.e. $\varepsilon = 0$). We see that the a posteriori bounds are significantly tighter for CG and GD (and are the only valid option for HB). Furthermore, the a priori bounds are not always valid once



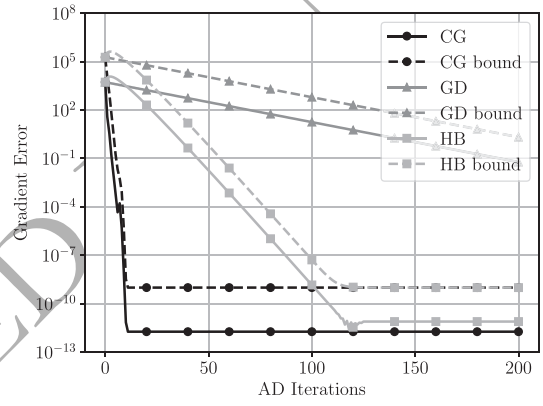
(a) Lower-level solve, a priori bounds.



(b) Lower-level solve, a posteriori bounds.



(c) AD comparison, a priori bounds (exact $\widetilde{x} = x^*(\theta)$)



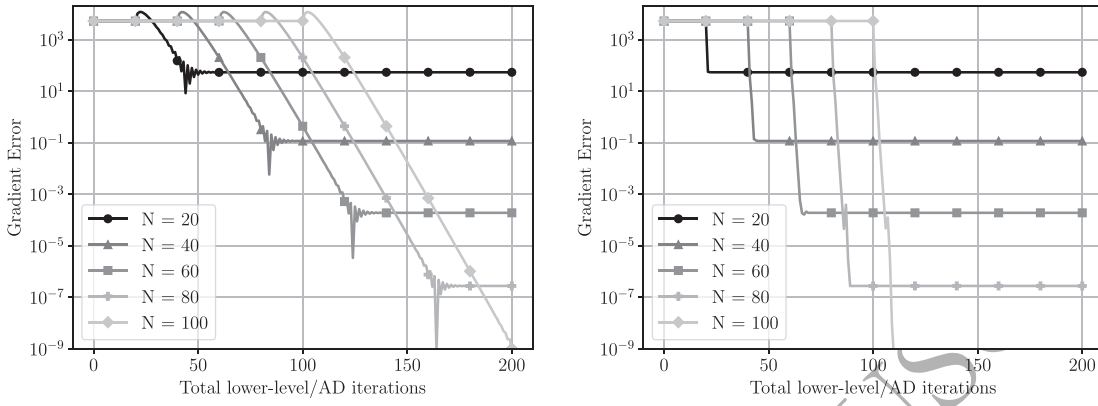
(d) AD comparison, a posteriori bounds (exact $\widetilde{x} = x^*(\theta)$)

FIG. 1. Simple quadratic AD comparison. *The a priori bound for heavy ball uses $c = 1$ and $\gamma = 0$ in (4.3) for (a,b) and (4.6) for (c,d), but in reality these constants are not known and so there is no guarantee that this will actually be an upper bound on the error. Note: in (b), the FISTA bound is very tight and is almost on top of the true FISTA error.

the hypergradient error is small enough that rounding errors are significant, whereas the a posteriori bounds automatically handle this.

We also consider the same results as Figures 1(c,d), but where $\widetilde{x} \neq x^*$ (i.e. $\varepsilon > 0$). These results are shown in Figure A.1 in Appendix A. Specifically, the fastest lower-level solver (HB) was run for $N \in \{20, 60, 100\}$ iterations and $\widetilde{x}$ was taken as the final iterate, corresponding to $\varepsilon \in \{1.1 \times 10^{-2}, 3.9 \times 10^{-8}, 7.7 \times 10^{-14}\}$ respectively. Here, we see that the a priori bounds are tighter for large ε, but the a posteriori bounds become more useful as $\varepsilon \to 0$.

Lastly, Figure 2 shows the true hypergradient errors from Figure A.1 in Appendix A, but comparing the overall gradient error against total computational work (measured as the sum of lower-level iterations and hypergradient iterations), for different levels of lower-level solve accuracy. Here, we are interested in considering how to allocate a given budget of computational resources between producing more accurate lower-level solves and more accurate hypergradients. We see that there is a genuine



(a) Using HB for AD method

(b) Using CG for AD method

FIG. 2. Simple quadratic problem: comparing actual gradient error versus total computational work (lower-level solve plus hypergradient iterations) for different accuracies of lower-level solve (N is the number of heavy ball iterations used to compute $\widetilde{x}$).

trade-off that must be considered: larger N for more accurate lower-level solves ultimately can give significantly more accurate hypergradients, but for very small budgets a smaller N should be used to allow the hypergradient iteration to run for sufficiently long. The trade-off that appears here aligns with the necessary balance between ε and δ inherent in our a priori bound (Theorem 10).

5.2. *Impact of Hypergradient Method on Optimization Quality*

We now consider a more realistic example problem to answer the question: how does the choice of hypergradient method affect the quality of the overall bilevel optimization process? To answer this question we use a data hypercleaning problem from [36, Appendix B]. This process is to learn weights for all training examples in a supervised learning problem, where some training examples have corrupted labels (and so the standard equal weighting is not ideal), by minimizing loss over a validation dataset. In this case, we consider multi-class logistic regression on MNIST with a cross-entropy loss:

$$\min_{\theta\in\mathbb{R}^{N_{\text{train}}}} F(\theta) := \frac{1}{N_{\text{val}}}\sum_{i=1}^{N_{\text{val}}} \ell(X^*(\theta)x_i^{\text{val}}, y_i^{\text{val}}), \tag{5.2a}$$

$$\text{s.t.} \quad X^*(\theta) := \operatorname*{arg\,min}_{X\in\mathbb{R}^{n_c\times d}} \frac{1}{N_{\text{train}}}\sum_{j=1}^{N_{\text{train}}} \sigma(\theta_j)\ell(Xx_j^{\text{train}}, y_j^{\text{train}}) + C\|X\|_F^2, \tag{5.2b}$$

where $\ell(y_{\text{est}}, y_{\text{true}}) : \mathbb{R}^{n_c}\times\mathbb{R}^{n_c}\to\mathbb{R}$ is the cross-entropy loss, and $\sigma(\cdot)$ is the sigmoid function. We have $n_c = 10$ classes, feature size $d = 785$, ℓ_2 penalty $C = 0.001$ (as chosen in [36]) and dataset sizes $N_{\text{train}} = 20000$ and $N_{\text{val}} = 5000$. A randomly chosen 10% of the training labels y_j^{train} are corrupted by choosing an incorrect label uniformly at random. The goal of the hypercleaning problem is effectively to encourage the lower-level weights $\sigma(\theta_j)\to 0$ where y_j^{train} is corrupted and $\sigma(\theta_j)\to 1$ otherwise.

As in Section 5.1, we use GD, HB and FISTA as lower-level solvers and CG, GD and HB as hypergradient algorithms. To solve the full bilevel problem, we run gradient descent with constant step-size (in this case taking $\alpha = 10$) on the upper-level problem using the calculated inexact hypergradients.

We use warm restarts for the lower-level solver, choosing $x^{(0)}$ in (2.9) or (2.10) (for example) to be the final value found in the previous iteration (with the previous value of θ). Since we are interested in the impact on the full upper-level solve, we show how the upper-level objective $F(\theta)$ decreases as a function of total computational work, taken as the sum of the total lower-level iterations and hypergradient iterations. We use this measure as each iteration of these requires one lower-level gradient and one lower-level Hessian-vector product respectively (with a similar cost).[6]

Our results are shown in Figure 3, for the different choices $\varepsilon, \delta \in \{10^{-2}, 10^{-1}\}$. We omit the results using GD as a hypergradient algorithm since they are all significantly worse than HB and CG (although we do show results with GD as a lower-level solver).

Comparing lines of the same shade (i.e. same lower-level solver), it is clear that the choice of hypergradient method has a substantial impact on the speed of the overall optimization. Indeed, our results suggest that the choice of hypergradient algorithm is at least as important as the choice of lower-level solver. In Figure 3(c,d), it is even the case that using GD as a lower-level solver with CG for hypergradients outperforms using HB for both (i.e. a non-accelerated lower-level solver with CG can outperform using an accelerated solver for both steps).

In Figure 3(b,d), we see that the fastest solver (HB lower-level/CG hypergradients) plateaus after sufficient time. This is because the solver has reached a level of accuracy where the first hypergradient iteration $q^{(0)} = 0$ has a sufficiently small residual and so a zero hypergradient is returned. However this is not a fundamental limit: the level of $F(\theta)$ corresponding to the plateau is exceeded in Figure 3(c) by taking a smaller value of δ. This suggests that a dynamic upper-level algorithm where ε and δ are carefully decreased to zero may be a superior method (c.f. the fixed decrease schedule for the bilevel solver HOAG [17]). The development and analysis of such an approach is delegated to future work.

5.3. *Quality of Learned Regularizer*

Lastly, we include an example demonstrating that our approach is capable of learning interesting regularizers for image denoising that outperform standard methods.

Here, our test problem is image denoising using variational regularization where the regularizer is an input-convex neural network (ICNN) [6, 7]. Thus, the lower-level problems g_i (1.1b) have the form $g_i(y, \theta) := \|y - z_i\|^2 + R(y, \theta)$. In more detail, we take $R(y, \theta)$ to be a linear combination of an input-convex neural network and an ℓ_2 penalty, i.e.

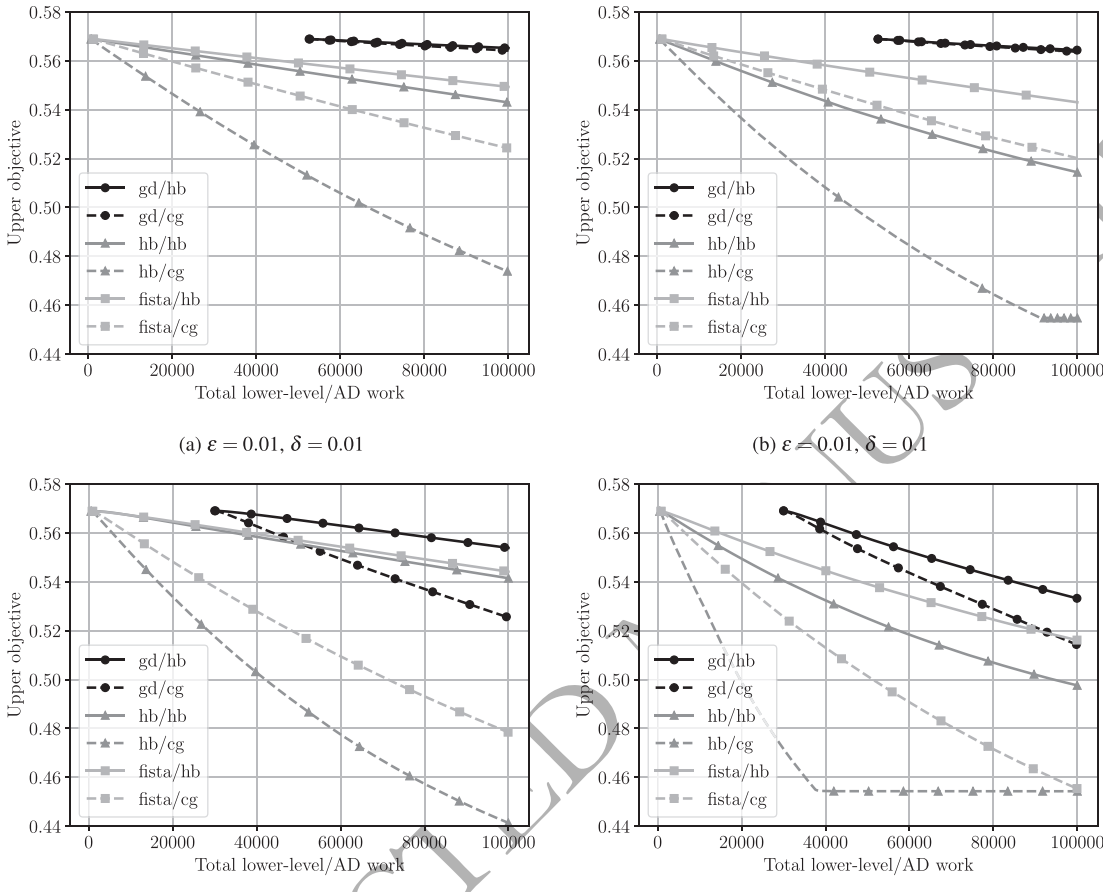
$$R(y, \theta) = \log(1 + e^{\theta_1}) S(y, \theta_{3:\text{end}}) + \log(1 + e^{\theta_2}) \|y\|^2. \tag{5.3}$$

The map S is a shallow neural network which comprises of a single convolutional layer (with kernel length 3 and two output channels) with a softplus activation function, followed by an averaging output layer. The use of the softplus function $t \mapsto \log(1 + e^t)$ in (5.3) is to ensure non-negativity of the coefficients and thus convexity of R in y. With this architecture, we have $n = 8$ parameters θ to learn. Our implementation of this regularizer is based on [7].[7]

We use a least-squares loss as the upper-level objective (1.1a), i.e. $f_i(y) := \|y - x_i\|^2$, where we have training data pairs $x_i, z_i \in \mathbb{R}^d$, corresponding to ground truth images x_i and noisy images z_i. Throughout, our dataset comprises true images $x_i \in \mathbb{R}^d$ for $d = 64$ which are discontinuous and piecewise linear with 4 segments, each with a random slope generated from $\text{Unif}([-10, 10])$ and with a jump

[6] We ignore the contribution of Jacobian-vector products in the hypergradient calculation, since there is only one per calculation compared to one Hessian-vector product per iteration of the hypergradient calculations.

[7] https://github.com/Subhadip-1/data_driven_convex_regularization



(a) $\varepsilon = 0.01$, $\delta = 0.01$ (b) $\varepsilon = 0.01$, $\delta = 0.1$

(c) $\varepsilon = 0.1$, $\delta = 0.01$ (d) $\varepsilon = 0.1$, $\delta = 0.1$

FIG. 3. Data hypercleaning results: upper-level objective achieved for a given amount of total work (cumulative lower-level iterations plus AD iterations) for different combinations of lower-level solver and AD method.

of size Unif$([-1,1])$ between segments. The noisy data z_i come from perturbing x_i with independent component-wise noise drawn from Unif$([-\delta,\delta])$ with $\delta = \frac{0.2}{d}\sum_{j=1}^{d}|x_j|$. To reflect a realistic imaging setting where full uncorrupted data acquisition (i.e. collecting suitable x_i) is generally difficult, we consider a setting where we have $m = 10$ training and 10 test images. An example image may be seen in Figure 5.

We will compare the ICNN (5.3) with the classical total variation (TV) regularizer $R(y,\theta) = \theta\,\mathrm{TV}(y)$ as implemented in [15]. The hyperparameter to be learned is the regularization weight $\theta > 0$ and $\mathrm{TV}(y) := \sum_j \|\hat{\nabla} y_j\|$ is the discretized total variation, i.e. $\hat{\nabla} y_j$ is a forward difference approximation to the gradient of y at pixel j. This regularizer is built around the prior that y is approximately piecewise constant.

We now demonstrate that the considered bilevel learning framework with the optimized gradient estimates can produce high-quality learned input-convex neural net regularizers for image denoising.

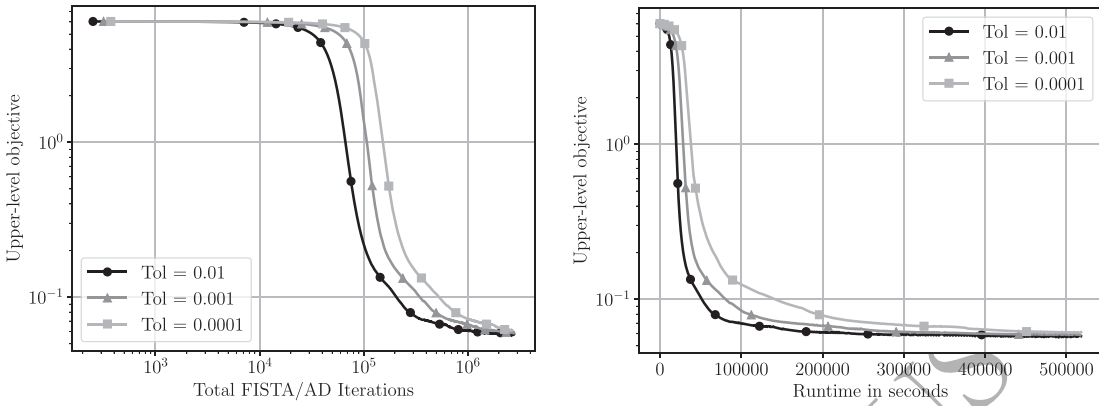


FIG. 4. Convergence of bilevel solver with tolerance $\varepsilon = \delta$ constant. It can be seen that computing hypergradients with higher accuracy does not lead to an overall efficient algorithm.

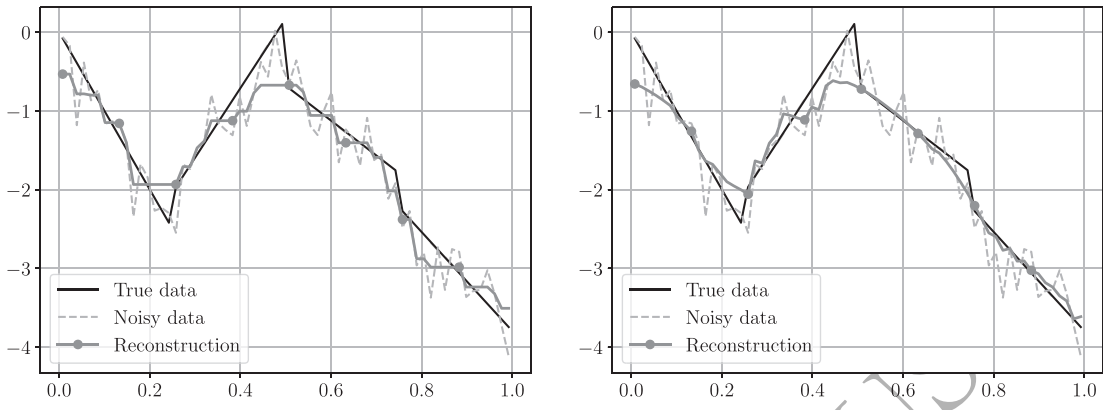
TABLE 1 *Quantitative comparison between TV and ICNN. Training and test losses of tuned regularizers from different frameworks. ICNN with lowest accuracy hypergradients lead to the smallest test loss.*

Method	Training Accuracy	Training loss	Test loss
TV		0.0601	0.0558
ICNN	$\varepsilon = \delta = 10^{-2}$	**0.0567**	**0.0552**
	$\varepsilon = \delta = 10^{-3}$	0.0586	0.0569
	$\varepsilon = \delta = 10^{-4}$	0.0607	0.0582

We ran the same bilevel solver (gradient descent) as in Section 5.2, now with a constant upper-level stepsize of 0.01. All hypergradients were calculated using FISTA as the lower-level solver and CG to solve the IFT linear system with $\varepsilon = \delta \in \{10^{-2}, 10^{-3}, 10^{-4}\}$ constant. All solvers were run for 6 days.

The resulting upper-level objective decrease (versus computational budget) is shown in Figure 4. Separately, we also use the derivative-free approach of [15] to tune the TV regularizer weight ($\theta \approx 0.026$ being optimal for this training dataset). Using the best learned hyperparameters θ from each method, the resulting training and test losses are shown in Table 1. We see that the proposed bilevel framework with $\varepsilon = \delta = 10^{-2}$ can produce an improved training loss and test loss than a highly tuned version of the custom-designed TV regularizer. By comparison, the input-convex neural net regularizer had no a prior information about the dataset. We conclude by noting that we would expect the training loss to be strictly decreasing as $\varepsilon = \delta$ is reduced, which is not observed in Table 1. This can be attributed to our finite computational budget, and shows the practical trade-off between high accuracy and practical efficiency.

Interestingly, TV and ICNN with optimized parameters have very different properties. In Figure 5 we show the two reconstructions for an example image from the test dataset with similar upper-level losses. Clearly the TV regularizer yields reconstructions which have a strong piecewise constant preference, but the ICNN regularizer produces much smoother reconstructions with occasional jumps.



(a) TV (loss 0.0601) (b) ICNN $\varepsilon = \delta = 10^{-2}$ (loss 0.0497)

FIG. 5. Qualitative comparison between TV and ICNN regularization using final tuned parameters. While the general approximation capabilities of the two models are similar it is worth pointing out that the learned ICNN does not lead the the staircasing artefact apparent in the TV reconstruction.

6. Conclusion

This work has demonstrated that the promising inexact AD approach for computing hypergradients is equivalent to using the implicit function theorem. This leads to a simple, unified framework for approximating hypergradients. Our framework is flexible, with no specific requirements on solver choices and termination conditions, and is accompanied by standard linear convergence rates as well as new, computable a posteriori error bounds. In practice these a posteriori bounds are also typically more accurate. Importantly, our results also demonstrate that careful selection of the hypergradient approximation method is as important for bilevel optimization as choosing a lower-level solver. Our results provide a promising foundation for building practical and rigorous bilevel optimization methods which will be addressed in future work.

Acknowledgments

This work is supported in part by funds from EPSRC (EP/S026045/1, EP/T026693/1, EP/V026259/1) and the Leverhulme Trust (ECF-2019-478).

REFERENCES

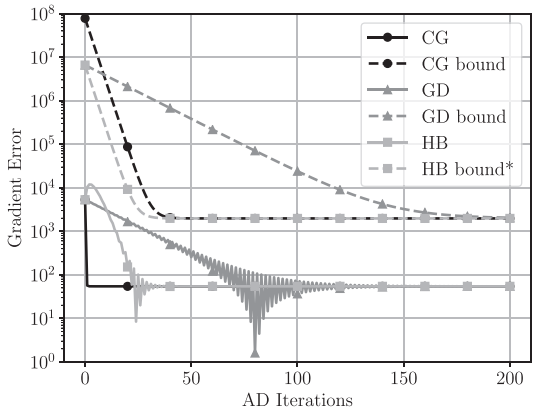
1. Stephan Dempe, Vyacheslav Kalashnikov, Gerardo A. Pérez-Valdés, and Nataliya Kalashnykova. *Bilevel Programming Problems-Theory, Algorithms and Applications to Energy Networks*. Springer, 2015.
2. Juan Carlos De los Reyes and David Villacís. Bilevel optimization methods in imaging. In *Handbook of Mathematical Models and Algorithms in Computer Vision and Imaging*, pages 1–34. Springer Nature Switzerland AG, 2021.
3. Caroline Crockett and Jeffrey A. Fessler. Bilevel methods for image reconstruction. *Foundations and Trends in Signal Processing*, 15(2–3):121–289, 2022.
4. Karl Kunisch and Thomas Pock. A Bilevel Optimization Approach for Parameter Learning in Variational Models. *SIAM Journal on Imaging Sciences*, 6(2):938–983, 2013.

5. Juan Carlos De Los Reyes and Carola-Bibiane Schönlieb. Image Denoising: Learning the Noise Model via Nonsmooth PDE-Constrained Optimization. *Inverse Problems and Imaging*, 7:1183–1214, 2013.
6. Brandon Amos, Lei Xu, and J. Zico Kolter. Input convex neural networks. In *Proceedings of the 34th International Conference on Machine Learning*, volume 70, pages 146–155, 2017.
7. Subhadip Mukherjee, Sören Dittmer, Zakhar Shumaylov, Sebastian Lunz, Ozan Öktem, and Carola-Bibiane Schönlieb. Learned convex regularizers for inverse problems. *NeurIPS 2021 Workshop on Deep Learning and Inverse Problems*, available at arXiv:2008.02839, 2021.
8. Laurent Hoeltgen, Simon Setzer, and Joachim Weickert. An optimal control approach to find sparse data for Laplace interpolation. In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, volume 8081 LNCS, pages 151–164, 2013.
9. Yunjin Chen, Rene Ranftl, and Thomas Pock. A bi-level view of inpainting-based image compression. In *19th Computer Vision Winter Workshop*, pages 19–26, 2014.
10. Ferdia Sherry, Martin Benning, Juan Carlos De los Reyes, Martin J. Graves, Georg Maierhofer, Guy Williams, Carola-Bibiane Schönlieb, and Matthias Joachim Ehrhardt. Learning the sampling pattern for MRI. *IEEE Transactions on Medical Imaging*, 39(12):4310–4321, 2020.
11. M. D. McKay, R. J. Beckman, and W. J. Conover. A comparison of three methods for selecting values of input variables in the analysis of output from a computer code. *Technometrics*, 21(2):239–245, 1979.
12. James Bergstra and Yoshua Bengio. Random search for hyper-parameter optimization. *Journal of Machine Learning Research*, 13:281–305, 2012.
13. Frank Hutter, Holger Hoos, and Kevin Leyton-Brown. Sequential model-based optimization for general algorithm configuration. *International Conference on Learning and Intelligent Optimization*, pages 507–523, 2011.
14. Jasper Snoek, Hugo Larochelle, and Ryan P. Adams. Practical Bayesian optimization of machine learning algorithms. In *Advances in Neural Information Processing Systems 25 (NIPS 2012)*, pages 2951–2959, 2012.
15. Matthias J Ehrhardt and Lindon Roberts. Inexact derivative-free optimization for bilevel learning. *Journal of Mathematical Imaging and Vision*, 63(5):580–600, 2021.
16. Yoshua Bengio. Gradient-based optimization of hyperparameters. *Neural Computation*, 12(8):1889–1900, 2000.
17. Fabian Pedregosa. Hyperparameter optimization with approximate gradient. In *Proceedings of the 33rd International Conference on Machine Learning*, pages 737–746, New York, 2016.
18. Nicolas Zucchet and João Sacramento. Beyond backpropagation: implicit gradients for bilevel optimization. *Neural Computation*, 34(12):2309–2346, 2022.
19. Sheheryar Mehmood and Peter Ochs. Automatic differentiation of some first-order methods in parametric optimization. In *Proceedings of the 23rd International Conference on Artificial Intelligence and Statistics (AISTATS)*, volume 108, pages 1584–1594, Palermo, Italy, 2020.
20. Bruce Christianson. Reverse accumulation and attractive fixed points. *Optimization Methods and Software*, 3(4):311–326, 1994.
21. Riccardo Grazzi, Luca Franceschi, Massimiliano Pontil, and Saverio Salzo. On the iteration complexity of hypergradient computation. *37th International Conference on Machine Learning, ICML 2020*, PartF16814(2):3706–3716, 2020.
22. Saeed Ghadimi and Mengdi Wang. Approximation methods for bilevel programming. *arXiv preprint arXiv:1802.02246*, 2018.
23. Kaiyi Ji, Junjie Yang, and Yingbin Liang. Bilevel optimization for machine learning: Algorithm design and convergence analysis. In *Proceedings of the 38th International Conference on Machine Learning*, pages 4882–4892, 2021.
24. Mingyi Hong, Hoi-To Wai, Zhaoran Wang, and Zhuoran Yang. A two-timescale framework for bilevel optimization: Complexity analysis and application to actor-critic. *SIAM Journal on Optimization*, 33(1):147–180, 2023.
25. Peter Ochs, Rene Ranftl, Thomas Brox, and Thomas Pock. Bilevel optimization with nonsmooth lower level problems. In *International Conference on Scale Space and Variational Methods in Computer Vision*, volume

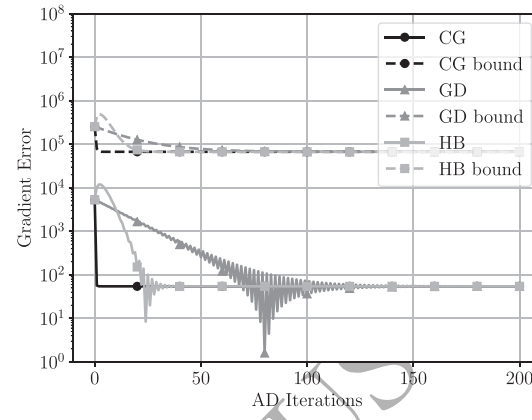
9087, pages 654–665, 2015.
26. Dougal Maclaurin, David Duvenaud, and Ryan P. Adams. Gradient-based hyperparameter optimization through reversible learning. In *Proceedings of the 32nd International Conference on Machine Learning,*, volume 37, pages 2113–2122, 2015.
27. Amirreza Shaban, Ching An Cheng, Nathan Hatch, and Byron Boots. Truncated back-propagation for bilevel optimization. In *Proceedings of the 22nd International Conference on Artificial Intelligence and Statistics (AISTATS) 2019*, volume 89, pages 1723–1732, 2019.
28. Ensio Suonperä and Tuomo Valkonen. Linearly convergent bilevel optimization with single-step inner methods. arXiv:2205.04862, 2022.
29. Riccardo Grazzi, Massimiliano Pontil, and Saverio Salzo. Convergence properties of stochastic hypergradients. In *Proceedings of the 24th International Conference on Artificial Intelligence and Statistics (AISTATS) 2021*, volume 130, pages 3826–3834, 2021.
30. A. S. Berahas, L. Cao, and K. Scheinberg. Global convergence rate analysis of a generic line search algorithm with noise. *SIAM Journal on Optimization*, 31(2):1489–1518, 2021.
31. Liyuan Cao, Albert S. Berahas, and Katya Scheinberg. First- and second-order high probability complexity bounds for trust-region methods with noisy oracles. *arXiv preprint 2205.03667*, 2022.
32. Antonin Chambolle and Thomas Pock. An introduction to continuous optimization for imaging. *Acta Numerica*, 25:161–319, 2016.
33. Yurii Nesterov. *Introductory Lectures on Convex Optimization*. Springer US, 2004.
34. Jorge Nocedal and Stephen J. Wright. *Numerical Optimization*. Springer Series in Operations Research and Financial Engineering. Springer, New York, 2nd edition, 2006.
35. Junyi Li, Bin Gu, and Heng Huang. A fully single loop algorithm for bilevel optimization without Hessian inverse. *Proceedings of the AAAI Conference on Artificial Intelligence*, 36(7):7426–7434, 2022.
36. Junjie Yang, Kaiyi Ji, and Yingbin Liang. Provably faster algorithms for bilevel optimization. In M. Ranzato, A. Beygelzimer, Y. Dauphin, P.S. Liang, and J. Wortman Vaughan, editors, *Advances in Neural Information Processing Systems*, volume 34, pages 13670–13682. Curran Associates, Inc., 2021.

A. Extra Numerical Results

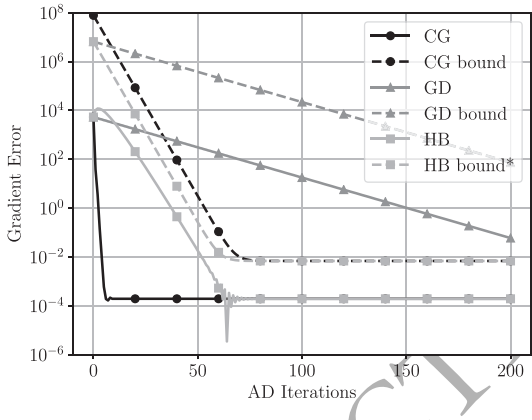
Figure A.1 below shows the same results as Figure 1(c,d), but where $\widetilde{x}$ is computed inexactly using N iterations of heavy ball.



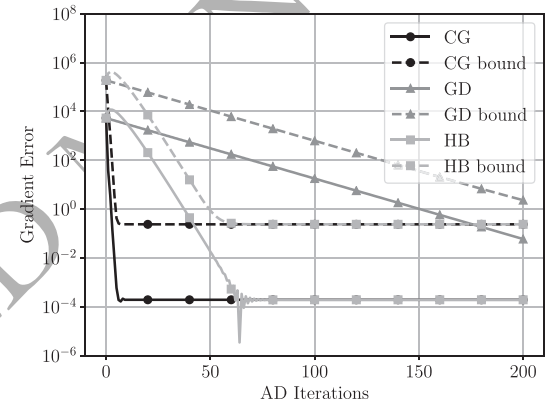
(a) $N = 20$ ($\varepsilon = 1.1 \times 10^{-2}$), a priori bounds



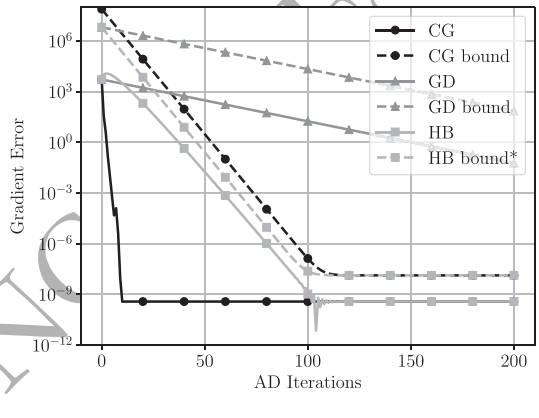
(b) $N = 20$ ($\varepsilon = 1.1 \times 10^{-2}$), a posteriori bounds



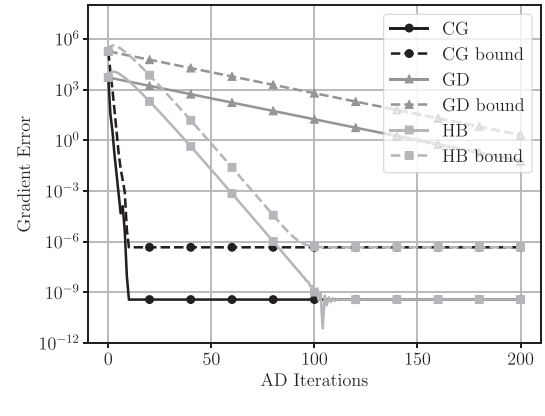
(c) $N = 60$ ($\varepsilon = 3.9 \times 10^{-8}$), a priori bounds



(d) $N = 60$ ($\varepsilon = 3.9 \times 10^{-8}$), a posteriori bounds



(e) $N = 100$ ($\varepsilon = 7.7 \times 10^{-14}$), a priori bounds



(f) $N = 100$ ($\varepsilon = 7.7 \times 10^{-14}$), a posteriori bounds

FIG. A.1. Simple quadratic AD comparison for different accuracy levels of lower-level solve ($\widetilde{x}$ from N iterations of heavy ball, yielding ε as stated). *The a priori bound for heavy ball uses $c = 1$ and $\gamma = 0$ in (4.6), but in reality these constants are not known and so there is no guarantee that this will actually be an upper bound on the error.