INDUSTRIAL AND SYSTEMS ENGINEERING

ISE

Inexact bilevel stochastic gradient methods for constrained and unconstrained lower-level problems

T. Giovannelli[1], G. Kent[1], and L. N. Vicente[1]

[1]Lehigh University

ISE Technical Report 21T-025

Inexact bilevel stochastic gradient methods for constrained and unconstrained lower-level problems

T. Giovannelli* G. D. Kent† L. N. Vicente‡

October 11, 2023

Abstract

Two-level stochastic optimization formulations have become instrumental in a number of machine learning contexts such as continual learning, neural architecture search, adversarial learning, and hyperparameter tuning. Practical stochastic bilevel optimization problems become challenging in optimization or learning scenarios where the number of variables is high or there are constraints.

In this paper, we introduce a bilevel stochastic gradient method for bilevel problems with nonlinear and possibly nonconvex lower-level constraints. We also present a comprehensive convergence theory that addresses both the lower-level unconstrained and constrained cases and covers all inexact calculations of the adjoint gradient (also called hypergradient), such as the inexact solution of the lower-level problem, inexact computation of the adjoint formula (due to the inexact solution of the adjoint equation or use of a truncated Neumann series), and noisy estimates of the gradients, Hessians, and Jacobians involved. To promote the use of bilevel optimization in large-scale learning, we have developed new low-rank practical bilevel stochastic gradient methods (BSG-N-FD and BSG-1) that do not require second-order derivatives and, in the lower-level unconstrained case, dismiss any matrix-vector products.

1 Introduction

Many real-world applications are naturally formulated using hierarchical objectives, which are organized into different nested levels. In the bilevel case, the main goal is placed into an upper optimization level, while the lower optimization level aims to determine the best response to a decision made in the upper level. Bilevel optimization has a rich literature of algorithmic development and theory (see [1, 11, 14, 15, 58, 63] for extensive surveys and books on this topic). The main applications are found in game theory, defense industry, and optimal structural design, and one has recently seen a surge of contributions to machine learning (ML) (see, e.g., [21, 34], and the recent review [35]).

*Department of Industrial and Systems Engineering, Lehigh University, Bethlehem, PA 18015-1582, USA (tog220@lehigh.edu).

†Department of Industrial and Systems Engineering, Lehigh University, Bethlehem, PA 18015-1582, USA (gdk220@lehigh.edu).

‡Department of Industrial and Systems Engineering, Lehigh University, Bethlehem, PA 18015-1582, USA (lnv@lehigh.edu).

In this paper, we consider the following nonlinear bilevel optimization problem (BLP) formulation, where we are using a standard notation (see, e.g., [14, 58, 63])

$$\begin{aligned} \min_{x\in\mathbb{R}^n,\, y\in\mathbb{R}^m} \quad & f_u(x,y) \\ \text{s.t.} \quad & x \in X \\ & y \in \operatorname*{argmin}_{y\in Y(x)} \quad f_\ell(x,y). \end{aligned} \tag{BLP}$$

The goal of the upper-level (UL) problem is to determine the optimal value of the UL function $f_u : \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}$, where the UL variables x are subjected to UL constraints ($x \in X$) and the lower-level (LL) variables y are subjected to being an optimal solution of the LL problem. In the LL problem, the LL function $f_\ell : \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}$ is optimized in the LL variables y, subject to the LL constraints $y \in Y(x)$. We will state the assumptions required for stochastic gradient descent in Subsection 3.1. We will assume f_u to be continuously differentiable in (x, y) and f_ℓ to be twice continuously differentiable in (x, y). We will also assume the LL problem to be well-defined, in the sense that an LL optimal solution $y(x)$ exists and is unique for all $x \in X$. Hence, problem BLP is equivalent to a problem posed solely in the UL variables:

$$\min_{x\in\mathbb{R}^n} \; f(x) = f_u(x, y(x)) \quad \text{s.t.} \quad x \in X. \tag{1.1}$$

Also, note that the UL constraints ($x \in X$) are only posed in the UL variables x as otherwise problem BLP could become intractable due to a disconnected feasible region in the (x, y)–space. The set X will be assumed closed and convex, which will allow us to ensure UL feasibility by applying orthogonal projections within stochastic gradient type methods.

Assuming $\nabla^2_{yy} f_\ell(x, y(x))$ is non-singular (again, see Subsection 3.1 for the statement of the assumptions), the gradient of f at x, when $Y(x) = \mathbb{R}^m$, is given by the well-known so-called adjoint gradient (also called hypergradient in the ML community)

$$\nabla f \;=\; \nabla_x f_u - \nabla^2_{xy} f_\ell \nabla^2_{yy} f_\ell^{-1} \nabla_y f_u, \tag{1.2}$$

where all gradients and Hessians on the right-hand side are evaluated at $(x, y(x))$. We denote the steepest descent direction for f at x as $d(x, y(x)) = -\nabla f(x)$. One arrives at the adjoint formula by first applying the chain rule to $f_u(x, y(x))$ to obtain

$$\nabla f \;=\; \nabla_x f_u + \nabla y \nabla_y f_u. \tag{1.3}$$

Then, the Jacobian $\nabla y(x)^\top$ of $y(x)$ can be calculated through the (sensitivity) equations $\nabla_y f_\ell(x, y(x)) = 0$. The implicit function theorem ensures $y(\cdot)$ to be continuously differentiable [51]. By taking the derivative of both sides of the equation with respect to x and utilizing the chain rule, we obtain $\nabla^2_{yx} f_\ell + \nabla^2_{yy} f_\ell \nabla y^\top = 0$ (all Hessians are evaluated at $(x, y(x))$) which yields

$$\nabla y \;=\; -\nabla^2_{xy} f_\ell \nabla^2_{yy} f_\ell^{-1}. \tag{1.4}$$

Two approaches have been proposed in the literature to deal with $\nabla^2_{yy} f_\ell^{-1}$ in (1.2). One option is to compute the adjoint gradient by first solving the linear system given by the adjoint equation $\nabla^2_{yy} f_\ell \, \lambda = \nabla_y f_u$ for the adjoint variables $\lambda = \lambda(x, y(x))$, and then calculating $\nabla_x f_u - \nabla^2_{xy} f_\ell \, \lambda$. The second option is to truncate the Neumann series given by $\nabla^2_{yy} f_\ell^{-1} = \sum_{i=0}^{\infty} (I -$

$\nabla^2_{yy} f_\ell)^i$, which, however, requires either the strong assumption of $\|\nabla^2_{yy} f_\ell\|_2 < 1$ or the knowledge of a bound on the second derivatives to guarantee the convergence of the series.

When $Y(x) \neq \mathbb{R}^m$, it is still possible to use an adjoint formula to compute the gradient of f at x by using sensitivity arguments from nonlinear programming. Such an LL constrained case will be addressed in Section 2.2.

1.1 Bilevel machine learning

A variety of problems arising in machine learning can be formulated in terms of bilevel optimization. Continual learning, neural architecture search, adversarial training, and hyperparameter tuning are among the most popular examples (see [35] for a review on this topic).

Continual Learning (CL) aims to train ML models when the static task usually considered in learning problems (classification, regression, etc.) is replaced by a sequence of tasks that become available one at a time [37], and for which training and validation datasets are increasingly larger. For each task, a CL instance is formulated as a bilevel problem, where at the UL problem one minimizes the validation error on a subset of model parameters (which includes all hyperparameters), and at the LL problem the training error is minimized on the remaining parameters. Then, a sequence of bilevel problems (one for each task) is solved. In a sense, CL is close to meta-learning [28], where the goal is to determine the best learning process. The increasing interest in CL is motivated by the demand for approaches that help neural networks learn new tasks without forgetting the previous ones, a phenomenon which is referred to as *catastrophic forgetting* [22, 26, 41, 60]. Another relevant ML area where bilevel optimization is used is Neural Architecture Search (NAS) for Deep Learning. The goal of this problem is to automate the task of designing Deep Neural Networks (DNNs) such that the network's prediction error is minimized. In recent years, NAS has been proposed in a bilevel optimization formulation [34].

Finally, two other popular classes of ML applications that can be formulated by using bilevel optimization are adversarial training and hyperparameter tuning. Adversarial training aims to robustly address adversarial examples [61] which cannot be correctly classified by ML models once a small perturbation is applied. The adversarial training problem is handled by solving a min-max problem [30], which can be reformulated as a bilevel one. The max/LL problem is posed on the variables that perturb the data in a worst-case fashion, where the min/UL problem attempts to minimize the training error on the ML model parameters [25, 40]. Hyperparameter tuning aims to find the best values for the hyperparameters used in an ML model in order to increase its performance on unseen data [3, 6, 16, 21]. In the bilevel formulations proposed in the literature, the UL problem optimizes the validation error over the hyperparameters, while the LL problem has the goal of finding the ML model parameters (e.g., neural network weights) that minimize the training error.

1.2 Bilevel stochastic descent

In bilevel stochastic optimization, f_u and f_ℓ can be interpreted as expected values, namely, $f_u = \mathbb{E}[f_u(x, y, \vartheta^u)]$ and $f_\ell = \mathbb{E}[f_\ell(x, y, \vartheta^\ell)]$, where ϑ^u and ϑ^ℓ are random variables defined in a probability space (with probability measure independent from x and y) such that i.i.d. samples can be observed or generated. The same applies to the functions possibly defining $Y(x)$. (To keep the notation simple, we are using the same f_u and f_ℓ for deterministic and random variants.)

Having in mind ML applications, the methods that we are considering are Stochastic Approximation (SA) techniques, of the type of the stochastic gradient (SG) method [10, 50, 53] for single-objective optimization. In fact, the bilevel stochastic gradient (BSG) method can be seen as an SG method applied to (1.1), which leads to $x_{k+1} = x_k - \alpha_k g_k^{\text{BSG}}$, where α_k is the step size or learning rate and g_k^{BSG} is a stochastic gradient of f. Such a stochastic gradient is obtained by sampling the gradients and Hessians in (1.2) at $(x_k, \tilde{y}_k)$, with $\tilde{y}_k$ denoting an approximation to the LL optimal solution. The stochastic gradient g_k^{BSG} is inexact when $\tilde{y}_k \neq y(x_k)$, even in the full-batch (deterministic) case.

In general, BSG methods have mainly been considered for the LL unconstrained case (i.e., $Y(x) = \mathbb{R}^m$) and are commonly classified according to the approach used to compute the BSG direction g_k^{BSG} [35]. In particular, a first category, referred to as *implicit differentiation*, is composed of algorithms that compute the BSG direction by applying the implicit function theorem and either solving the adjoint equation [46] or using a truncated Neumann series to approximate the inverse Hessian $\nabla^2_{yy} f_\ell^{-1}$ [38]. Note that using a truncated Neumann series requires $\|\nabla^2_{yy} f_\ell\|_2 < 1$, which is a strong assumption. Therefore, a common approach is to first assume $\nabla_y f_\ell$ Lipschitz continuous in y with constant C_0, and then apply the truncated Neumann series to approximate $[(1/C_0)\nabla^2_{yy} f_\ell]^{-1}$. However, this requires the knowledge of C_0, which is typically unknown in practice. A second category, referred to as *iterative differentiation*, includes all the approaches based on automatic differentiation through dynamic systems [17, 21, 39]. A general convergence theory for the two classes of algorithms was proposed in [29], which also shows that computing the BSG direction by using automatic differentiation can be less computationally efficient than using an implicit differentiation method. Therefore, our paper focuses on the first category, which has been promoted in [9, 12, 13, 24, 27, 59] (see also [7, 35] for recent reviews). These existing approaches either focus on a specific problem structure or require the LL problem to be solved to optimality at each iteration or rely on a truncated Neumann series for the inverse Hessian approximation in the adjoint gradient, which has the issues mentioned before. Among all the algorithms proposed in the papers cited above, we emphasize StocBiO [29]. StocBiO employs a truncated Neumann series with automatic differentiation and a double-loop iterative scheme, which means that multiple iterations are required at the LL problem in order for the algorithm to converge, similar to the algorithms developed in our paper.

DARTS [34] is an optimization technique related to the BSG method and has enjoyed great popularity in NAS. It always considers an inexact solution to the LL problem, and it starts an iteration by applying one step of SG to the LL problem, $\tilde{y}_k = y_k - \eta \nabla_y f_\ell(x_k, y_k)$, where η is a fixed step size. Then, it displaces the UL variables using $x_{k+1} = x_k - \eta\, g_k^{\text{DARTS}}$, where g_k^{DARTS} is computed by applying the chain rule to $\nabla_x f_u(x, y - \eta \nabla_y f_\ell(x, y))$, leading to (when using full-batch gradients)

$$g_k^{\text{DARTS}} \;=\; \nabla_x f_u(x_k, \tilde{y}_k) - \eta \nabla^2_{xy} f_\ell(x_k, y_k) \nabla_y f_u(x_k, \tilde{y}_k). \tag{1.5}$$

However, in the full-batch (deterministic) case, g_k^{DARTS} may not be a descent direction. The matrix-vector product in (1.5) is approximated by finite differences, rendering DARTS free of both second-order derivatives and matrix-vector products (see Section 5.2).

All the papers cited above focus on the LL unconstrained case. A few approaches dealing with the LL constrained case have been very recently proposed to tackle bilevel problems with LL constraints. However, all these approaches are only applicable to linear constraints or constraints only depending on y [31, 33, 57, 62, 64]. Among the algorithms proposed, we highlight SIGD [31],

which determines a direction by applying the implicit function theorem (similar to the approach that we use) but can only be applied to bilevel problems with linear LL constraints in y.

1.3 Contributions of the paper

The first main contribution of this paper is a general framework for the BSG method that applies to both the LL unconstrained ($Y(x) = \mathbb{R}^m$) and constrained ($Y(x) \neq \mathbb{R}^m$) cases. In particular, our paper represents the first work that proposes a method to address the general nonlinear and possibly nonconvex LL constrained case, which has not been covered elsewhere, neither algorithmically nor theoretically, although important ML applications give rise to bilevel problems with LL constraints.

The second main contribution is a comprehensive convergence theory for the BSG method that applies to both the LL unconstrained and constrained cases and is grounded on sensitivity principles of nonlinear optimization. The sensitivity arguments used in this paper are easily satisfied in all practical scenarios that have been proposed for ML applications requiring bilevel optimization. Our theory also comprehensively covers all possible inexact settings such as the inexact solution of the LL problem, inexact computation of the adjoint formula (due to the inexact solution of the adjoint equation or use of a truncated Neumann series), and noisy estimates of the gradients, Hessians, and Jacobians involved. Since the convergence analysis proposed is abstracted from the specifics of the approach used to handle the inverse matrix in the adjoint formula (1.2), our theory unifies two different classes of BSG methods, i.e., the ones based on the adjoint equation and the ones based on the truncated Neumann series, which have been studied separately [24, 46].

Moreover, to deal with the second-order derivatives and inverse Hessians/Jacobians in the adjoint formulas for the LL unconstrained and constrained cases, we developed two new low-rank practical implementations of the BSG method that can be applied to solve large-scale optimization problems arising in ML applications. The first one, referred to as BSG-N-FD, consists of solving the adjoint equation by using an iterative method (CG or GMRES) equipped with finite differences to dismiss any Hessian-vector products, and is grounded on theoretical principles. The second one, referred to as BSG-1, uses rank-1 Hessian approximations to avoid explicitly solving the adjoint equation (also dismissing any matrix-vector products). The use of these rank-1 approximations is inspired by Gauss-Newton methods for nonlinear least-squares problems [45] and also from the fact that the empirical risk of misclassification in ML is often a sum of non-negative terms, matching a function to a scalar which can then be considered in a least-squares fashion [4, 23]. Similar to DARTS [34] (which is extremely popular for NAS), BSG-1 is also not grounded on theoretical principles. However, both methods perform well on continual learning instances in terms of training iterations and computational time.

1.4 Organization of this paper

This paper is organized as follows. In Section 2, we describe the BSG method for the LL unconstrained case and introduce it for the LL constrained case. The assumptions on the problem functions and inexact calculations required for the convergence analysis of the method are reported in Section 3. Section 4 presents the convergence rates for the nonconvex, strongly convex, and convex cases. Numerical results for synthetic quadratic bilevel problems and continual learning instances with or without LL constraints are analyzed in Section 5, which also describes

the practical BSG-N-FD and BSG-1 algorithms. Finally, in Section 6 we draw some concluding remarks and propose ideas for future work. By default, all norms $\|\cdot\|$ used in this paper are the ℓ_2 ones.

2 The bilevel stochastic gradient method

In this section, we introduce the bilevel stochastic gradient (BSG) method for solving stochastic BLPs. Let $\{\vartheta_k^\ell\}_{k\geq 0}$ and $\{\varsigma_k^\ell\}_{k\geq 0}$ be sequences of random variables for LL gradient, Hessian, and Jacobian evaluations for the LL unconstrained and constrained cases, respectively. Similarly, let $\{\vartheta_k^u\}_{k\geq 0}$ be a sequence of random variables for UL gradient evaluations. A realization of ϑ_k^ℓ, ς_k^ℓ, and ϑ_k^u can be interpreted as a single sample or a batch of samples for mini-batch SG. For compactness of notation, let us set $\xi_k = (\vartheta_k^u, \vartheta_k^\ell)$ in the LL unconstrained case ($Y(x) = \mathbb{R}^m$), and $\xi_k = (\vartheta_k^u, \vartheta_k^\ell, \varsigma_k^\ell)$ in the LL constrained case ($Y(x) \neq \mathbb{R}^m$). All the assumptions included in this section will be rigorously stated in Subsection 3.1.

2.1 The unconstrained lower-level case

Given $(x_k, \tilde{y}_k)$, we denote by $g_x^u(x_k, \tilde{y}_k, \vartheta_k^u)$, $g_y^u(x_k, \tilde{y}_k, \vartheta_k^u)$, and $g_y^\ell(x_k, \tilde{y}_k, \vartheta_k^\ell)$ the stochastic gradient estimates that approximate $\nabla_x f_u(x_k, \tilde{y}_k)$, $\nabla_y f_u(x_k, \tilde{y}_k)$, and $\nabla_y f_\ell(x_k, \tilde{y}_k)$, respectively. The same notation applies to the stochastic Hessian estimates: $H_{xy}^\ell(x_k, \tilde{y}_k, \vartheta_k^\ell)$ and $H_{yy}^\ell(x_k, \tilde{y}_k, \vartheta_k^\ell)$ approximate $\nabla_{xy}^2 f_\ell(x_k, \tilde{y}_k)$ and $\nabla_{yy}^2 f_\ell(x_k, \tilde{y}_k)$, respectively. Based on Assumption 3.2, which will be stated in Subsection 3.1.1, $\nabla_{yy}^2 f_\ell^{-1}$ and $(H_{yy}^\ell)^{-1}$ are non-singular at all points. In the LL unconstrained case ($Y(x) = \mathbb{R}^m$), an approximate (negative) BSG (denoted as $-g_k^{\text{BSG}}$ in Subsection 1.2) can be computed directly from the adjoint formula (1.2), as follows:

$$d(x_k, \tilde{y}_k, \xi_k) \;=\; -\left(g_x^u(x_k, \tilde{y}_k, \vartheta_k^u) - H_{xy}^\ell(x_k, \tilde{y}_k, \vartheta_k^\ell) H_{yy}^\ell(x_k, \tilde{y}_k, \vartheta_k^\ell)^{-1} g_y^u(x_k, \tilde{y}_k, \vartheta_k^u)\right). \tag{2.1}$$

The data in the formula (1.2) is referred to as $D(x, y)$ or $D(x, y(x))$, depending on the point where the gradients and Hessians are evaluated:

$$D(x, y) \;=\; \left(\nabla_x f_u(x, y), \nabla_y f_u(x, y), \nabla_{xy}^2 f_\ell(x, y), \nabla_{yy}^2 f_\ell(x, y)\right). \tag{2.2}$$

The data in the calculation (2.1) is referred to as $D(x_k, \tilde{y}_k, \xi_k)$:

$$D(x_k, \tilde{y}_k, \xi_k) \;=\; \left(g_x^u(x_k, \tilde{y}_k, \vartheta_k^u), g_y^u(x_k, \tilde{y}_k, \vartheta_k^u), H_{xy}^\ell(x_k, \tilde{y}_k, \vartheta_k^\ell), H_{yy}^\ell(x_k, \tilde{y}_k, \vartheta_k^\ell)\right). \tag{2.3}$$

Note that in (2.1), $d(x_k, \tilde{y}_k, \xi_k)$ can be seen as a function of the data $D(x_k, \tilde{y}_k, \xi_k)$ as follows:

$$d(x_k, \tilde{y}_k, \xi_k) \;=\; d(D(x_k, \tilde{y}_k, \xi_k)), \tag{2.4}$$

where we are using overlapping notation for $d(\cdot)$ for the sake of simplicity and because we believe it is more powerful and effective than using a different letter.

As mentioned in Subsection 1.3, our practical implementation BSG-N-FD will solve the adjoint equation $H_{yy}^\ell(x_k, \tilde{y}_k, \vartheta_k^\ell)\lambda = g_y^u(x_k, \tilde{y}_k, \vartheta_k^u)$ by using an iterative method equipped with finite differences (see Subsection 5.1.1). The BSG-1 method will use rank-one approximations for $H_{xy}^\ell(x_k, \tilde{y}_k, \vartheta_k^\ell)$ and $H_{yy}^\ell(x_k, \tilde{y}_k, \vartheta_k^\ell)$, and then solve the resulting adjoint equation in the least-squares sense (see Subsection 5.1.2).

2.2 The constrained lower-level case

Let us now handle the LL constrained case, in which we consider

$$Y(x) \;=\; \{y \in \mathbb{R}^m \mid c_i(x,y) \leq 0,\; i \in I,\; \text{and } c_i(x,y) = 0,\; i \in E\},$$

where I and E are two finite sets of indices. As stated in the assumptions of Subsection 3.1.2, each constraint function c_i is assumed twice continuously differentiable in (x,y), for all $i \in I \cup E$. Denoting $c_I(x,y) = (c_i(x,y),\, i \in I)$ and $c_E(x,y) = (c_i(x,y),\, i \in E)$, the Lagrangian function of the LL problem is defined as $\mathcal{L}_\ell(x,y,z) = f_\ell(x,y) + c_I(x,y)^\top z_I + c_E(x,y)^\top z_E$, where z_I and z_E are Lagrange multipliers and $z = (z_I, z_E)$. We will also assume in Assumption 3.5 of Subsection 3.1.2 that there exists a $y(x)$ satisfying the LL KKT conditions with associated multipliers $(z_I(x), z_E(x))$ such that the gradients of the active constraints are linearly independent (LICQ), the strict complementarity slackness condition (SCS) is satisfied, and the second-order sufficient optimality conditions (SOSC) hold. Under such assumptions, it is well known that the Lagrange multipliers $z_I(x)$ and $z_E(x)$ associated with $y(x)$ are unique and the vector function $v(x) = (y(x), z_I(x), z_E(x))^\top$, for any given x, is once continuously differentiable [18, 19, 20, 42]. Moreover, we can write the first-order KKT system for the LL problem (see [18]) as

$$\begin{cases} \nabla_y f_\ell(x, y(x)) + \nabla_y c_I(x, y(x))\, z_I(x) + \nabla_y c_E(x, y(x))\, z_E(x) = 0, \\ z_I(x) \circ c_I(x, y(x)) = 0, \\ c_E(x, y(x)) = 0, \end{cases} \tag{2.5}$$

where $\circ$ is the element-wise multiplication operation of two vectors.

Now, for any given x, we can rewrite the KKT system (2.5) as $G(x, v(x)) = 0$ by introducing a corresponding vector function G. Applying the chain rule to $G(x, v(x)) = 0$, we obtain $\nabla_v G^\top \nabla v^\top = -\nabla_x G^\top$, with

$$\nabla_x G^\top = \begin{pmatrix} \nabla^2_{yx} \mathcal{L}_\ell \\ z_I \circ \nabla_x c_I^\top \\ \nabla_x c_E^\top \end{pmatrix} \quad \text{and} \quad \nabla_v G^\top = \begin{pmatrix} \nabla^2_{yy} \mathcal{L}_\ell & \nabla_y c_I & \nabla_y c_E \\ z_I \circ \nabla_y c_I^\top & C_I & 0 \\ \nabla_y c_E^\top & 0 & 0 \end{pmatrix}, \tag{2.6}$$

where the Hessian of $\mathcal{L}_\ell$ is evaluated at $(x, y(x), z_I(x), z_E(x))$, the Jacobian matrices $\nabla_x c_I^\top$, $\nabla_x c_E^\top$, $\nabla_y c_I^\top$, and $\nabla_y c_E^\top$ are evaluated at $(x, y(x))$, C_I is a diagonal matrix whose elements are given by $c_I(x, y(x))$, and $z_I \circ \nabla_x c_I^\top$ is a matrix obtained by multiplying the entries of z_I by the corresponding rows of $\nabla_x c_I^\top$ (a similar explanation applies to $z_I \circ \nabla_y c_I^\top$).

Since under Assumption 3.5 in Subsection 3.1.2, the Jacobian $\nabla_v G^\top$ is non-singular at $(x, v(x))$ (see [42, 45]), we obtain

$$\nabla v \;=\; \big(\nabla y, \nabla z_I, \nabla z_E\big) \;=\; -\nabla_x G\, \nabla_v G^{-1}.$$

We can now pull out the columns of this system that correspond to the $\nabla y(x)$ term by introducing an appropriate matrix $L = \begin{pmatrix} \mathbf{I}_m & \mathbf{0} \end{pmatrix}^\top$, where $\mathbf{I}_m$ is an identity matrix of size m and $\mathbf{0}$ is a null matrix of size $m \times \big(|I| + |E|\big)$, yielding

$$\nabla y(x) \;=\; -\nabla_x G \nabla_v G^{-1} L. \tag{2.7}$$

Substituting (2.7) into (1.3), we obtain the following adjoint gradient for the LL constrained case:

$$\nabla f \;=\; \nabla_x f_u - \nabla_x G \nabla_v G^{-1} L \nabla_y f_u, \tag{2.8}$$

where the gradients of f_u with respect to x and y are evaluated at $(x, y(x))$, while the Jacobians of G with respect to x and v are evaluated at $(x, v(x))$. The negative adjoint gradient provides the steepest descent direction for f at x in the deterministic case. Note that one can deal with the inverse Jacobian in (2.8) by applying the same approaches used for the LL unconstrained case, i.e., solving the adjoint equation or using a truncated Neumann series (see Subsection 2.5).

Similar to the notation used for the LL unconstrained case in Subsection 2.1, given $(x_k, \tilde{v}_k)$, with $\tilde{v}_k = (\tilde{y}_k, (\tilde{z}_I)_k, (\tilde{z}_E)_k)$, the Jacobian estimates $\mathcal{G}_x(x_k, \tilde{v}_k, \varsigma_k^\ell)^\top$ and $\mathcal{G}_v(x_k, \tilde{v}_k, \varsigma_k^\ell)^\top$ approximate $\nabla_x G(x_k, \tilde{v}_k)^\top$ and $\nabla_v G(x_k, \tilde{v}_k)^\top$, respectively. Based on Assumption 3.6 in Subsection 3.1.2, $\nabla_v G$ and $\mathcal{G}_v$ are non-singular at all points. An approximate (negative) BSG can be computed directly from the adjoint formula (2.8), as follows:

$$d(x_k, \tilde{v}_k, \xi_k) \;=\; -\Big(g_x^u(x_k, \tilde{y}_k, \vartheta_k^u) - \mathcal{G}_x(x_k, \tilde{v}_k, \varsigma_k^\ell)\, \mathcal{G}_v(x_k, \tilde{v}_k, \varsigma_k^\ell)^{-1} L\, g_y^u(x_k, \tilde{y}_k, \vartheta_k^u) \Big). \tag{2.9}$$

The data in (2.8) is referred to as $D(x, v)$ or $D(x, v(x))$, depending on the point where the gradients, Hessians, and Jacobians are evaluated:

$$D(x, v) \;=\; (\nabla_x f_u(x, y),\, L\nabla_y f_u(x, y),\, \nabla_x G(x, v),\, \nabla_v G(x, v)). \tag{2.10}$$

The data in calculation (2.9) is now referred to as $D(x_k, \tilde{v}_k, \xi_k)$:

$$D(x_k, \tilde{v}_k, \xi_k) \;=\; \Big(g_x^u(x_k, \tilde{y}_k, \vartheta_k^u),\, L\, g_y^u(x_k, \tilde{y}_k, \vartheta_k^u),\, \mathcal{G}_x(x_k, \tilde{v}_k, \varsigma_k^\ell),\, \mathcal{G}_v(x_k, \tilde{v}_k, \varsigma_k^\ell) \Big). \tag{2.11}$$

Again, note that in (2.9), $d(x_k, \tilde{v}_k, \xi_k)$ can be interpreted as a function of the data $D(x_k, \tilde{v}_k, \xi_k)$ (see the explanation for (2.4)).

It is worth mentioning that an alternative approach to obtain a direction colinear with the negative gradient of f in the LL constrained case was proposed in [55]. However, such an approach requires solving an auxiliary linear-quadratic bilevel problem, which is not practical in terms of solving large-scale ML application problems. Also, the approach in [55] does not yield the exact size of the gradient.

2.3 A unified notation for LL unconstrained and constrained cases

Our goal is to propose a general algorithm that applies to both the LL unconstrained and constrained cases. For each case, one can denote the BSG directions used in the deterministic setting (i.e., (1.2) and (2.8)) and stochastic one (i.e, (2.1) and (2.9)) by using the unified notation

$$d(D) = -(a - AB^{-1}b), \tag{2.12}$$

where $D = (a, b, A, B)$. In particular, in the LL unconstrained case, when the adjoint formula (1.2) or (2.1) is used, the data D is either the deterministic one (2.2) or the stochastic one (2.3), respectively. Similarly, in the constrained case, when the adjoint formula (2.8) or (2.9) is used, the data D is again either the deterministic one (2.10) or the stochastic one (2.11), respectively. Moreover, we use the following notation to encapsulate the LL variables in the two cases

$$w \;=\; \begin{cases} y \text{ when } Y(x) = \mathbb{R}^n, \\ v \text{ when } Y(x) \neq \mathbb{R}^n, \end{cases} \tag{2.13}$$

i.e., w is equal to y in the unconstrained case and v in the constrained case.

2.4 The BSG method

The schema of the BSG method is presented in Algorithm 1. An initial point $(x_0, \tilde{w}_0)$ and a sequence of positive scalars $\{\alpha_k\}$ are required as input. In Step 1, any arbitrary optimization method can be applied to approximately solve the LL problem, regardless of being unconstrained or constrained. In Step 2, one computes an approximate (negative) BSG, which will be denoted by $d(x_k, \tilde{w}_k, \xi_k)$ and computed through (2.1) or (2.9). We recall from (2.4) that $d(x_k, \tilde{w}_k, \xi_k) = d(D(x_k, \tilde{w}_k, \xi_k))$, where $D(x_k, \tilde{w}_k, \xi_k)$ is either (2.3) or (2.11). Finally, at Step 3, the vector x is updated by using a proper step size taken from the sequence of positive scalars. When X is a closed and convex set different from $\mathbb{R}^n$, we need to compute the orthogonal projection of $x_k + \alpha_k\, d(x_k, \tilde{w}_k, \xi_k)$ onto X (note that such a projection can be computed by solving a convex optimization problem).

Algorithm 1 Bilevel Stochastic Gradient (BSG) Method

Input: $(x_0, \tilde{w}_0)$, $\{\alpha_k\}_{k\geq 0} > 0$.

For $k = 0, 1, 2, \ldots$ **do**
Step 1. Obtain an approximation $\tilde{w}_k$ to the LL optimal solution $w(x_k)$.
Step 2. Compute a (negative) stochastic gradient approximation $d(x_k, \tilde{w}_k, \xi_k)$.
Step 3. Compute $x_{k+1} = P_X(x_k + \alpha_k\, d(x_k, \tilde{w}_k, \xi_k))$.
End do

We point out that as is usual in the literature related to SG methods, a stopping criterion is not considered due to a lack of reasonable criteria and for the need to study the asymptotic convergence properties.

2.5 Computing the BSG direction inexactly

The two approaches proposed in the literature to deal with the inverse matrix B^{-1} in (2.12) consist of either solving the adjoint equation or using a truncated Neumann series. In particular, the first approach requires solving the adjoint equation $B\,\lambda = b$ for the adjoint variables λ. The BSG direction can thus be calculated by $a - A\,\lambda$. The residual error due to the inexact solution of the adjoint equation is denoted by $\tilde{r}$, i.e., $\tilde{r} = B\,\lambda - b$. Note that the previous expression can be written as $B\,\lambda = b + \tilde{r}$, where the right-hand side can be interpreted as a perturbation of the right-hand side in the adjoint equation. Since $\lambda = B^{-1}(b + \tilde{r})$ in the inexact adjoint solve case, the BSG direction becomes

$$-(a - A\,B^{-1}(b + \tilde{r})). \tag{2.14}$$

Given a positive scalar $q > 0$ and assuming $\|B\| < 1$, the second approach is based on the Neumann series as follows:

$$B^{-1} \;=\; \sum_{i=0}^{\infty} (I - B)^i \;=\; \mathscr{B} + \tilde{R},$$

where $\mathscr{B} = \sum_{i=0}^{q} (I-B)^i$ and $\tilde{R} = \sum_{i=q+1}^{\infty} (I-B)^i$. Note that the accuracy of the approximation is an increasing function of q. An approximation to B^{-1} is given by $\mathscr{B}$, i.e., $B^{-1} \simeq \mathscr{B}$. Therefore,

in the inexact Neumann series case, the BSG direction becomes

$$-(a - A\,(B^{-1} - \tilde{R})\,b). \tag{2.15}$$

3 Assumptions, sensitivity, and smoothness

In this section, we introduce the assumptions used in the convergence analysis of the BSG method, which extends the convergence theory of the SG method to the bilevel case when the stepsize is assumed to be decaying. Using the notation introduced in (2.12)–(2.13), the convergence theory developed in this section covers both the LL unconstrained and constrained cases. The solution of the LL problem is assumed to be inexact. Our theory also applies when the BSG direction (2.12) is computed inexactly by using the approaches in Subsection 2.5, which comprehensively generalizes all existing approaches in the literature.

Given that we consider the application of the stochastic gradient method (or a similar SA technique) to solve the LL problem (see Step 1 of Algorithm 1), we will denote by ξ_k^{S1} the set of random variables for all combined iterations of the LL solution process at iteration k. Moreover, we will denote by ξ_k^{all} the set of all random variables for both the LL and UL solution processes at iteration k. Therefore, noticing that ξ_k denotes the set of random variables used at Step 2 of Algorithm 1, we denote $\xi_k^{\mathrm{all}} = (\xi_k^{\mathrm{S1}}, \xi_k)$. At each iteration, the iterate x_k is completely determined by the realizations of the independent random variables ξ_k^{S1} and ξ_k.

3.1 General assumptions

3.1.1 LL unconstrained case

We will start this subsection with general assumptions for the LL unconstrained case. Assumption 3.1 below imposes the appropriate smoothness for the problem gradients and Hessians, their boundedness, and the boundedness of their stochastic counterparts.

Assumption 3.1 (Smoothness and boundedness (LL unconstrained case)) *The gradient ∇f_u and the Hessians $\nabla^2_{xy} f_\ell$ and $\nabla^2_{yy} f_\ell$ are Lipschitz continuous. Moreover, ∇f_u, $\nabla^2_{xy} f_\ell$, and $\nabla^2_{yy} f_\ell$ and their stochastic estimates g_x^u, g_y^u, H_{xy}^ℓ, and H_{yy}^ℓ are bounded at all points.*

Assumption 3.2 below ensures the existence and uniqueness of an LL optimal solution $y(x)$ for the original problem as well as for its corresponding stochastic approximation.

Assumption 3.2 (Existence and uniqueness of solution (LL unconstrained case)) *There exists a $y(x)$ such that $\nabla f_\ell(x, y(x)) = 0$, and $\nabla^2_{yy} f_\ell$ is positive definite at all points. In the stochastic case, H_{yy}^ℓ is positive definite at all points.*

Assumption 3.3 below requires the inverse of the Hessian of f_ℓ w.r.t. y to be uniformly bounded, which is equivalent to saying that $f_\ell(x, \cdot)$ is strongly convex (a very standard assumption in the literature, see, e.g., [24]). We also need uniform boundedness in the stochastic case.

Assumption 3.3 (Uniform convexity of problem (LL unconstrained case)) *The Hessians $\nabla^2_{yy} f_\ell^{-1}$ and $(H_{xy}^\ell)^{-1}$ are uniformly bounded at all points.*

3.1.2 LL constrained case

Now, we will introduce assumptions that are specific to the LL constrained case. We will start with smoothness assumptions for the gradients and Hessians of the problem, their boundedness, and the boundedness of their stochastic counterparts.

Assumption 3.4 (Smoothness and boundedness (LL constrained case)) *The gradient ∇f_u, the Jacobians $\nabla c_I^\top$ and $\nabla c_E^\top$, and the Hessians $\nabla^2_{xy} f_\ell$, $\nabla^2_{yy} f_\ell$, $\nabla^2_{yx} c_i$, and $\nabla^2_{yy} c_i$, for all $i \in I \cup E$, are Lipschitz continuous. In addition, ∇f_u, $\nabla c_I^\top$, $\nabla c_E^\top$, $\nabla^2_{xy} f_\ell$, $\nabla^2_{yy} f_\ell$, $\nabla^2_{yx} c_i$, and $\nabla^2_{yy} c_i$, for all $i \in I \cup E$, and their stochastic estimates are uniformly bounded at all points. Finally, the multipliers z and their stochastic counterparts are uniformly bounded at all points.*

The boundedness of the multipliers is required so that all Hessians of the Lagrangian are also bounded as well as cross terms of the type $z \circ \nabla c^\top$.

We can now introduce our assumption for the existence and uniqueness of LL solutions. For that purpose, given Lagrange multipliers $(z_I(x), z_E(x))$ associated with a solution $y(x)$ of the LL KKT conditions, let us denote the cone of critical directions [45] as follows:

$$Z(x) \;=\; \left\{ d^y \neq 0 \;:\; \begin{array}{c} \nabla_y c_i(x, y(x))^\top d^y \leq 0, \;\forall i \in I(x) \\ \nabla_y c_i(x, y(x))^\top d^y = 0, \;\forall i \in I(x) \text{ with } (z_I(x))_i > 0 \\ \nabla_y c_i(x, y(x))^\top d^y = 0, \;\forall i \in E \end{array} \right\}, \tag{3.1}$$

where $I(x)$ is the index set of the active inequality constraints at $(x, y(x))$ defined in Subsection 2.2.

The linear independence constraint qualification (LICQ) ensures that the gradients $\nabla_y c_i(x, y(x))$, for all $i \in I(x) \cup E$, are linearly independent. The strict complementarity slackness condition (SCS) states that for all multipliers $(z_I(x), z_E(x))$ satisfying the LL KKT conditions at $(x, y(x))$, one has $(z_I(x))_i > 0$ for all $i \in I(x)$. The second-order sufficient condition (SOSC) states that for all multipliers $(z_I(x), z_E(x))$ satisfying the LL KKT conditions at $(x, y(x))$ and for all $d^y \in Z(x)$, where $Z(x)$ is defined by (3.1), one has $(d^y)^\top \nabla^2_{yy} \mathcal{L}_\ell(x, v(x)) d^y > 0$.

Assumption 3.5 (Existence and uniqueness of solution (LL constrained case)) *There exists a $y(x)$ satisfying the LL KKT conditions with associated multipliers $(z_I(x), z_E(x))$ such that the LICQ, SCS, and SOSC are satisfied.*

This guarantees that $y(x)$ is a strict local minimizer for the LL problem. To ensure that $y(x)$ is the unique global minimizer, one can further assume either that $\nabla^2_{yy} \mathcal{L}_\ell(x, y, z_I, z_E)$ is positive semi-definite for all (y, z_I, z_E) or that $Y(x)$ is convex and $\nabla^2_{yy} \mathcal{L}_\ell(x, y, z_I, z_E)$ is positive semi-definite for all (y, z_I, z_E) on the tangent cone to the set $Y(x)$ at $y(x)$.

As in the unconstrained case, we also need to impose some form of uniform boundedness away from singularity. In the constrained case, this is achieved through the KKT matrix of the LL problem, which is the corresponding counterpart to the Hessian of the objective function f_ℓ.

Assumption 3.6 ("Uniform convexity" of problem (LL constrained case)) *The KKT matrices $\nabla_v G^{-1}$ and $\mathcal{G}_v^{-1}$ are uniformly bounded at all points.*

3.1.3 Notation for constants

In this subsection, we introduce constants that will be used to denote bounds on gradients, Hessians, and Jacobians (Remark 3.1 below), and a bound on the second moment of the BSG direction (Assumption 3.7 below).

Remark 3.1 *As a consequence of Assumptions 3.1 and 3.4, there exist positive constants C and $\bar{C}$ such that, for any (x, y), (x, v), (x, y, ϑ), (x, y, ϑ^ℓ), and (x, v, ς^ℓ), we have $\|\nabla_y f_u\| \leq C$, $\|\nabla^2_{xy} f_\ell\| \leq C$, $\|\nabla^2_{yy} f_\ell\| \leq C$, $\|\nabla_x G\| \leq C$, $\|\nabla_v G\| \leq C$, $\|g^u_y\| \leq \bar{C}$, $\|H^\ell_{xy}\| \leq \bar{C}$, $\|H^\ell_{yy}\| \leq \bar{C}$, $\|\mathcal{G}_x\| \leq \bar{C}$, and $\|\mathcal{G}_v\| \leq \bar{C}$. Assumptions 3.3 and 3.6 imply that there exist positive constants C_ℓ and $\bar{C}_\ell$ such that, for any (x, y), (x, y, ϑ^ℓ), (x, v), and (x, v, ς^ℓ), we have $\|\nabla^2_{yy} f^{-1}_\ell\| \leq C_\ell$, $\|(H^\ell_{yy})^{-1}\| \leq \bar{C}_\ell$, $\|\nabla_v G^{-1}\| \leq C_\ell$, and $\|\mathcal{G}^{-1}_v\| \leq \bar{C}_\ell$.*

In Assumption 3.7 below, we require the BSG direction $d(x_k, w(x_k), \xi_k)$ to have a bounded second moment, which is a classical assumption in the SG literature [5]. Such an assumption will be used to derive a bound on the second moment of the approximate BSG direction (see Lemma 3.1). The expected value with respect to the probability distributions of ξ_k and ξ^{S1}_k are denoted by $\mathbb{E}_{\xi_k}[\cdot]$ and $\mathbb{E}_{\xi^{\mathrm{S1}}_k}[\cdot]$, respectively. The expected value with respect to the joint distribution of ξ_k and ξ^{S1}_k is denoted by $\mathbb{E}_{\xi^{\mathrm{all}}_k} = \mathbb{E}_{\xi_k}[\mathbb{E}_{\xi^{\mathrm{S1}}_k}[\cdot]]$.

Assumption 3.7 (Bound on the second moment of the BSG direction) *There exists a positive scalar V_d such that the vector $d(x_k, w(x_k), \xi_k)$ satisfies the following condition:*

$$\mathbb{E}_{\xi_k}[\|d(x_k, w(x_k), \xi_k)\|^2] \;\leq\; V_d.$$

3.2 Sensitivity of the approximate bilevel stochastic gradient direction

Let us recall that $f(x) = f_u(x, y(x))$. To bound the second moment of the approximate BSG direction $d(x_k, \tilde{w}_k, \xi_k)$ and the expectation of the error between the negative gradient $-\nabla f(x_k)$ and $d(x_k, \tilde{w}_k, \xi_k)$, we will need to apply sensitivity analysis arguments from nonlinear optimization. We start by assuming that at each iteration, the calculation process of the BSG direction (2.12) is (approximately) Lipschitz continuous with respect to changes in its data. Note that this result is presented as a general assumption that any stochastic algorithm for solving bilevel problems needs to satisfy in order for our convergence theory in Section 4 to hold. However, such an assumption is not restrictive, and Proposition 3.1 shows that it can be easily enforced in all practical scenarios that have been proposed for ML applications requiring bilevel optimization.

Assumption 3.8 (Sensitivity of the BSG direction) *Given any pair of data $(D_1)_k$ and $(D_2)_k$, there exists a constant $L_{BSG} > 0$ such that*

$$\|d((D_1)_k) - d((D_2)_k)\| \;\leq\; L_{BSG}(\|(D_1)_k - (D_2)_k\| + \|(r_1)_k - (r_2)_k\|), \tag{3.2}$$

where $(r_1)_k$ and $(r_2)_k$ are the residual errors in the inexact computations of $d((D_1)_k)$ and $d((D_2)_k)$. Moreover, when $(D_1)_k = D(x_k, w(x_k), \xi_k)$ and $(D_2)_k = D(x_k, \tilde{w}_k, \xi_k)$, one has

$$\|(D_1)_k - (D_2)_k\| \;\leq\; \bar{L}_{LL}\|w(x_k) - \tilde{w}_k\|, \tag{3.3}$$

where $\bar{L}_{LL} > 0$ is a constant only dependent on the Lipschitz constants of the stochastic gradients, Hessians, and Jacobians of Assumptions 3.1 and 3.4.

Proposition 3.1 below shows that the inexact ways (2.14) and (2.15) of calculating adjoint gradients or BSG directions do ensure that inequality (3.2) of Assumption 3.8 is satisfied. Parts of the proof have been published elsewhere [13]. In fact, the arguments used are the known facts that sum is Lipschitz continuous, multiplication is Lipschitz continuous if the factors are bounded, and matrix inversion is Lipschitz continuous if its singular values are bounded away from zero. Moreover, Proposition 3.1 shows that inequality (3.3) can be satisfied when assuming the Lipschitz continuity of the stochastic estimates used in the BSG directions. We point out that such an assumption is met when the gradients, Hessians, and Jacobians have a finite-sum structure (as commonly found in ML application problems) and all the terms included in the sums are Lipschitz continuous. Note that this aligns with the typical scenario in ML, where the same gradient, Hessian, or Jacobian function is evaluated at different data points. In this case, the randomness ξ consists of drawing a batch, which trivially renders the assumption true.

Proposition 3.1 (Sensitivity of the BSG direction) *Under Assumptions 3.1–3.6, given any pair of data $(D_1)_k$ and $(D_2)_k$, let $(r_1)_k$ and $(r_2)_k$ be the residual errors incurred when $d((D_1)_k)$ and $d((D_2)_k)$ are computed inexactly by either* (2.14) *(in which case $(r_1)_k$, $(r_2)_k$ are $(\tilde{r}_1)_k$, $(\tilde{r}_2)_k$) or* (2.15) *(in which case $(r_1)_k$, $(r_2)_k$ are $(\tilde{R}_1)_k$, $(\tilde{R}_2)_k$). Then, there exists a constant $L_{BSG} > 0$ such that inequality* (3.2) *of Assumption 3.8 is satisfied. When $(D_1)_k = D(x_k, w(x_k), \xi_k)$ and $(D_2)_k = D(x_k, \tilde{w}_k, \xi_k)$, assuming the stochastic estimates of the gradients, Hessians, and Jacobians in $(D_1)_k$ and $(D_2)_k$ to be Lipschitz continuous in w for all ξ_k, there exists a positive constant $\bar{L}_{LL}$ such that inequality* (3.3) *of Assumption 3.8 is satisfied.*

Proof. See Appendix A for the proof, where we omit the dependence on k for simplicity. □

We now introduce Assumption 3.9 on the absolute error of the LL optimal solution, whose validity in practice is discussed in Subsection 4.4.

Assumption 3.9 *There exists a positive scalar C_w such that*

$$\mathbb{E}_{\xi_k^{\mathrm{S1}}}[\|w(x_k) - \tilde{w}_k\|^2] \;\leq\; (C_w\,\alpha_k)^2.$$

By applying Jensen's inequality, one also has $(\mathbb{E}_{\xi_k^{\mathrm{S1}}}[\|w(x_k) - \tilde{w}_k\|])^2 \;\leq\; \mathbb{E}_{\xi_k^{\mathrm{S1}}}[\|w(x_k) - \tilde{w}_k\|^2]$, thus

$$\mathbb{E}_{\xi_k^{\mathrm{S1}}}[\|w(x_k) - \tilde{w}_k\|] \;\leq\; C_w\,\alpha_k. \tag{3.4}$$

We also need Assumption 3.10 below to hold which essentially amounts to the sampling error in the data. To enforce this assumption in practice, we refer the reader to the discussion reported in Section 4.5. Recall that $D(x_k, \tilde{w}_k)$ represents the deterministic data defining the quantity $d(x_k, \tilde{w}_k)$ (see (2.2) and (2.10)), and $D(x_k, \tilde{w}_k, \xi_k)$ the stochastic data of the calculation of $d(x_k, \tilde{w}_k, \xi_k)$ (see (2.3) and (2.11)).

Assumption 3.10 *There exists a positive scalar C_D such that*

$$\mathbb{E}_{\xi_k^{\mathrm{all}}}[\|D(x_k, \tilde{w}_k) - D(x_k, \tilde{w}_k, \xi_k)\|] \;\leq\; C_D\,\alpha_k.$$

Finally, we need to bound the residual errors introduced in Assumption 3.8. In practice, when the BSG direction is computed inexactly (see Subsection 2.5), this assumption can be enforced by either increasing the accuracy in the inexact adjoint equation solve or increasing the value of q used to truncate the Neumann series.

Assumption 3.11 *There exists a positive scalar C_e such that, for all realizations of the algorithm,*

$$\|r_k\| \;\leq\; C_e\,\alpha_k,$$

where r_k is either $(r_1)_k$ or $(r_2)_k$ in Assumption 3.8.

One is ready to establish the desired bounds (see Lemmas 3.1 and 3.2 below), for which the constants will depend on the above-introduced constants.

Lemma 3.1 *Under Assumptions 3.1–3.9 and 3.11, and assuming the stepsize sequence $\{\alpha_k\}$ bounded from above by a constant $C_s > 0$, one has*

$$\mathbb{E}_{\xi_k^{\text{all}}}[\|d(x_k, \tilde{w}_k, \xi_k)\|^2] \;\leq\; G_d, \tag{3.5}$$

where $G_d = 2\,(U_d + V_d)$, $U_d = L_{BSG}^2 C_s^2 (L_{LL}^2 C_w^2 + 4\bar{L}_{LL} C_e C_w + 4C_e^2)$, and $\bar{L}_{LL} > 0$ is the constant introduced in Assumption 3.8.

Proof. By considering the data $D_1 = D(x_k, w(x_k), \xi_k)$ and $D_2 = D(x_k, \tilde{w}_k, \xi_k)$ in Assumption 3.8, we obtain

$$\begin{aligned}\|d(x_k, w(x_k), \xi_k) - d(x_k, \tilde{w}_k, \xi_k)\| \;&\leq L_{BSG}\|D(x_k, w(x_k), \xi_k) - D(x_k, \tilde{w}_k, \xi_k)\| \\ &+\; L_{BSG}\|r_1 - r_2\|.\end{aligned} \tag{3.6}$$

From (3.6), inequality (3.3) of Assumption 3.8, and Assumption 3.11, we obtain

$$\|d(x_k, w(x_k), \xi_k) - d(x_k, \tilde{w}_k, \xi_k)\| \;\leq\; L_{BSG}(\bar{L}_{LL}\|w(x_k) - \tilde{w}_k\| + 2C_e\alpha_k). \tag{3.7}$$

By raising both sides of (3.7) to the second power, we have

$$\|d(x_k, w(x_k), \xi_k) - d(x_k, \tilde{w}_k, \xi_k)\|^2 \;\leq\; L_{BSG}^2(\bar{L}_{LL}^2\|w(x_k) - \tilde{w}_k\|^2 + 4\bar{L}_{LL}C_e\alpha_k\|w(x_k) - \tilde{w}_k\| + 4C_e^2\alpha_k^2).$$

Therefore, by taking expectations with respect to the distribution of ξ_k^{all}, considering Assumption 3.9, (3.4), and the bound on the stepsize sequence, and denoting $U_d = L_{BSG}^2 C_s^2 (\bar{L}_{LL}^2 C_w^2 + 4\bar{L}_{LL} C_e C_w + 4C_e^2)$, we obtain

$$\mathbb{E}_{\xi_k^{\text{all}}}[\|d(x_k, w(x_k), \xi_k) - d(x_k, \tilde{w}_k, \xi_k)\|^2] \;\leq\; U_d, \tag{3.8}$$

where we have used that $\mathbb{E}_{\xi_k^{\text{all}}}[\|w(x_k) - \tilde{w}_k\|^i] \;=\; \mathbb{E}_{\xi_k^{\text{S1}}}[\|w(x_k) - \tilde{w}_k\|^i]$, with $i \in \{1, 2\}$.

From Assumption 3.7 and (3.8), we can obtain the desired bound on the second moment of the approximate BSG direction by adding and subtracting $d(x_k, w(x_k), \xi_k)$ and considering that $\mathbb{E}_{\xi_k^{\text{all}}}[\|d(x_k, w(x_k), \xi_k)\|^2] = \mathbb{E}_{\xi_k}[\|d(x_k, w(x_k), \xi_k)\|^2]$. In particular, we have

$$\begin{aligned}\mathbb{E}_{\xi_k^{\text{all}}}[\|d(x_k, \tilde{w}_k, \xi_k)\|^2] \;&\leq\; 2\,\mathbb{E}_{\xi_k^{\text{all}}}[\|d(x_k, \tilde{w}_k, \xi_k) - d(x_k, w(x_k), \xi_k)\|^2] \\ &\quad +2\,\mathbb{E}_{\xi_k}[\|d(x_k, w(x_k), \xi_k)\|^2] \\ &\leq\; 2\,(U_d + V_d).\end{aligned}$$

□

Lemma 3.2 *Under Assumptions 3.1–3.6 and 3.8–3.11,*

$$\mathbb{E}_{\xi_k^{\text{all}}}[\| - \nabla f(x_k) - d(x_k, \tilde{w}_k, \xi_k)\|] \;\leq\; C_d\, \alpha_k, \tag{3.9}$$

where $C_d = L_{BSG}(L_{LL}C_w + C_D + 4C_e)$, and $L_{LL} > 0$ is a constant only dependent on the Lipschitz constants of the gradients, Hessians, and Jacobians of Assumptions 3.1 and 3.4.

Proof. By adding and subtracting the term $d(x_k, \tilde{w}_k)$ and using the triangle inequality, we have

$$\begin{aligned}\mathbb{E}_{\xi_k^{\text{all}}}[\|d(x_k, w(x_k)) - d(x_k, \tilde{w}_k, \xi_k)\|] \;\leq\; & \mathbb{E}_{\xi_k^{\text{all}}}[\|d(x_k, w(x_k)) - d(x_k, \tilde{w}_k)\|] && (3.10)\\ & + \mathbb{E}_{\xi_k^{\text{all}}}[\|d(x_k, \tilde{w}_k) - d(x_k, \tilde{w}_k, \xi_k)\|]. && (3.11)\end{aligned}$$

Now we derive a bound for the right-hand side in (3.10). By considering the data $D_1 = D(x_k, w(x_k))$ and $D_2 = D(x_k, \tilde{w}_k)$ in Assumption 3.8, and taking the expectation, we obtain

$$\mathbb{E}_{\xi_k^{\text{all}}}[\|d(x_k, w(x_k)) - d(x_k, \tilde{w}_k)\|] \;\leq\; L_{BSG}\, \mathbb{E}_{\xi_k^{\text{all}}}[\|D(x_k, w(x_k)) - D(x_k, \tilde{w}_k)\|] \;+\; 2L_{BSG}C_e\alpha_k, \tag{3.12}$$

where we have applied Assumption 3.11 on $\|(r_1)_k - (r_2)_k\|$.

Note that the right-hand side of (3.12) contains exact BLP gradients and Hessians (or Jacobians in the LL constrained case). Therefore, the Lipschitz continuity of those mappings (Assumptions 3.1 and 3.4) implies the existence of a constant $L_{LL} > 0$ such that

$$\|D(x_k, w(x_k)) - D(x_k, \tilde{w}_k)\| \;\leq\; L_{LL}\|w(x_k) - \tilde{w}_k\|. \tag{3.13}$$

Taking expectations with respect to the distribution of ξ_k^{all} on both sides of (3.13), we can write

$$\mathbb{E}_{\xi_k^{\text{all}}}[\|D(x_k, w(x_k)) - D(x_k, \tilde{w}_k)\|] \;\leq\; L_{LL}\, \mathbb{E}_{\xi_k^{\text{S1}}}[\|w(x_k) - \tilde{w}_k\|], \tag{3.14}$$

where we have used that $\mathbb{E}_{\xi_k^{\text{all}}}[\|w(x_k) - \tilde{w}_k\|] \;=\; \mathbb{E}_{\xi_k^{\text{S1}}}[\|w(x_k) - \tilde{w}_k\|]$. Therefore, from (3.12), (3.14), and (3.4), we obtain

$$\mathbb{E}_{\xi_k^{\text{all}}}[\|d(x_k, w(x_k)) - d(x_k, \tilde{w}_k)\|] \;\leq\; L_{BSG}\,(L_{LL}\, C_w \;+\; 2C_e)\alpha_k. \tag{3.15}$$

Now we derive a bound for (3.11). By considering the data $D_1 = D(x_k, \tilde{w}_k)$ and $D_2 = D(x_k, \tilde{w}_k, \xi_k)$ in Assumption 3.8 and applying Assumption 3.11, we have

$$\|d(x_k, \tilde{w}_k) - d(x_k, \tilde{w}_k, \xi_k)\| \;\leq\; L_{BSG}\|D(x_k, \tilde{w}_k) - D(x_k, \tilde{w}_k, \xi_k)\| \;+\; 2L_{BSG}C_e\alpha_k. \tag{3.16}$$

Taking expectations with respect to the distribution of ξ_k^{all} on both sides of (3.16), we obtain

$$\mathbb{E}_{\xi_k^{\text{all}}}[\|d(x_k, \tilde{w}_k) - d(x_k, \tilde{w}_k, \xi_k)\|] \;\leq\; L_{BSG}\, \mathbb{E}_{\xi_k^{\text{all}}}[\|D(x_k, \tilde{w}_k) - D(x_k, \tilde{w}_k, \xi_k)\|] \;+\; 2L_{BSG}C_e\alpha_k. \tag{3.17}$$

Hence, from (3.17) and Assumption 3.10, we obtain

$$\mathbb{E}_{\xi_k^{\text{all}}}[\|d(x_k, \tilde{w}_k) - d(x_k, \tilde{w}_k, \xi_k)\|] \;\leq\; L_{BSG}\,(C_D + 2C_e)\alpha_k. \tag{3.18}$$

The proof can be concluded from (3.10)–(3.11), (3.15), and (3.18). □

3.3 Smoothness of the true objective function

Our convergence theory requires smoothness of the true function f, which is given in Proposition 3.2 below. In both the LL unconstrained and constrained cases, the Lipschitz continuity of ∇f can be inferred from the Lipschitz continuity of $y(x)$, $w(x)$, and the gradients, Hessians, and Jacobians involved (along with the boundedness away from singularity of Hessian or KKT matrices). The proof is given in Appendix B, and again part of the proof has been reported in [13] and relies on elementary arguments.

Proposition 3.2 (Smoothness of f) *Under Assumptions 3.1–3.6, there exists a constant $L_{\nabla f} > 0$ such that the gradient ∇f is Lipschitz continuous in x, i.e.,*

$$\|\nabla f(x_1) - \nabla f(x_2)\| \;\leq\; L_{\nabla f}\|x_1 - x_2\| \quad \text{for all } (x_1, x_2) \in \mathbb{R}^n \times \mathbb{R}^n. \tag{3.19}$$

An important and well-known consequence that follows from (3.19) is

$$f(x) \;\leq\; f(\bar{x}) + \nabla f(\bar{x})^\top (x - \bar{x}) + \frac{1}{2} L_{\nabla f}\|x - \bar{x}\|^2 \;\text{ for all }\; (x, \bar{x}) \in \mathbb{R}^n \times \mathbb{R}^n. \tag{3.20}$$

4 Convergence rate of the BSG method

In this section, we extend the convergence theory of the SG method to the bilevel case when the stepsize is assumed to be decaying. The BLP objective function f is assumed to be nonconvex, strongly convex (leading to a $1/k$ sublinear convergence rate), or simply convex ($1/\sqrt{k}$ rate).

Using the notation introduced in (2.12)–(2.13), the convergence theory developed in this section covers both the LL unconstrained and constrained cases. The BSG method under consideration takes an inexact solution of the LL problem, for which the stochasticity is rigorously included in the analysis for the first time. Moreover, such a theory also applies when the BSG direction (2.12) is computed inexactly regardless of the approach used (see Subsection 2.5), thus leading to an analysis that is considerably more general than the ones proposed in the literature [24, 46].

4.1 Rate in the nonconvex case

In this section, the true objective function f is assumed to be possibly nonconvex. We now present two lemmas that will allow us to prove the convergence result. Such lemmas and the resulting theorem are based on the theory provided in [5], where the main differences lie in the use of the projection operator to handle the upper-level constraints $x \in X$ and, importantly, in how inexactly $d(x_k, \tilde{w}_k, \xi_k)$ approximates $-\nabla f(x_k)$ (see Lemmas 3.1 and 3.2). The first lemma is just a Taylor bound derived as a result of (3.19).

Lemma 4.1 *Under Assumptions 3.1–3.6, the iterates of Algorithm 1 satisfy the following inequality for all $k \in \mathbb{N}$*

$$\begin{aligned}\mathbb{E}_{\xi_k^{\text{all}}}\left[f(x_{k+1})\right] - f(x_k) \leq \alpha_k \left(P_X \nabla f(x_k)\right)^\top \mathbb{E}_{\xi_k^{\text{all}}}\left[d(x_k, \tilde{w}_k, \xi_k)\right] \\ + \frac{1}{2}\alpha_k^2 L_{\nabla f} \mathbb{E}_{\xi_k^{\text{all}}}\left[\|d(x_k, \tilde{w}_k, \xi_k)\|^2\right].\end{aligned} \tag{4.1}$$

Proof. From equation (3.20), the iterates generated by Algorithm 1 satisfy

$$f(x_{k+1}) - f(x_k) \leq \nabla f(x_k)^\top (x_{k+1} - x_k) + \frac{1}{2} L_{\nabla f} \|x_{k+1} - x_k\|^2.$$

Recalling that Algorithm 1 uses the update $x_{k+1} = P_X\left(x_k + \alpha_k d(x_k, \tilde{w}_k, \xi_k)\right)$, and since x_k is in the feasible region X, we know that $x_k = P_X(x_k)$, which yields

$$f(x_{k+1}) - f(x_k) \;\leq\; \alpha_k \nabla f(x_k)^\top P_X d(x_k, \tilde{w}_k, \xi_k) + \frac{1}{2} \alpha_k^2 L_{\nabla f} \|P_X d(x_k, \tilde{w}_k, \xi_k)\|^2.$$

Since P_X is an orthogonal projection, we know $P_X = P_X^\top = P_X^2$ and $\|P_X\| \leq 1$. Using this fact and taking expectations with respect to the distribution of ξ_k^{all}, we obtain (4.1). □

The following lemma further extends the result of Lemma 4.1 by using the inexactness of $d(x_k, \tilde{w}_k, \xi_k)$ and the bound on its second-order moment.

Lemma 4.2 *Under Assumptions 3.1–3.11, the iterates generated by Algorithm 1 satisfy the following inequality for all $k \in \mathbb{N}$:*

$$\mathbb{E}_{\xi_k^{\text{all}}}\left[f(x_{k+1})\right] - f(x_k) \leq -\alpha_k \|P_X \nabla f(x_k)\|^2 + \alpha_k^2 C_{\nabla f} C_d + \frac{1}{2} \alpha_k^2 L_{\nabla f} G_d, \tag{4.2}$$

where $C_{\nabla f} > 0$ is a bound on the norm of ∇f.

Proof. From inequality (4.1) and Lemma 3.1, we have

$$\mathbb{E}_{\xi_k^{\text{all}}}\left[f(x_{k+1})\right] - f(x_k) \leq \alpha_k \left(P_X \nabla f(x_k)\right)^\top \mathbb{E}_{\xi_k^{\text{all}}}\left[d(x_k, \tilde{w}_k, \xi_k)\right] + \frac{1}{2} \alpha_k^2 L_{\nabla f} G_d.$$

Adding and subtracting $\alpha_k \left(P_X \nabla f(x_k)\right)^\top \mathbb{E}_{\xi_k^{\text{all}}}\left[\nabla f(x_k)\right]$ to the right-hand side and simplifying, we obtain

$$\begin{aligned}\mathbb{E}_{\xi_k^{\text{all}}}\left[f(x_{k+1})\right] - f(x_k) \;\leq\; & \alpha_k \left(P_X \nabla f(x_k)\right)^\top \mathbb{E}_{\xi_k^{\text{all}}}\left[d(x_k, \tilde{w}_k, \xi_k) + \nabla f(x_k)\right] \\ & - \; \alpha_k \left(P_X \nabla f(x_k)\right)^\top \nabla f(x_k) + \frac{1}{2} \alpha_k^2 L_{\nabla f} G_d.\end{aligned}$$

Applying the properties of orthogonal projections along with the Cauchy-Schwarz and Jensen's inequalities, we obtain

$$\begin{aligned}\mathbb{E}_{\xi_k^{\text{all}}}\left[f(x_{k+1})\right] - f(x_k) \;\leq\; & \alpha_k \|P_X \nabla f(x_k)\| \, \mathbb{E}_{\xi_k^{\text{all}}}\left[\|d(x_k, \tilde{w}_k, \xi_k) + \nabla f(x_k)\|\right] \\ & - \; \alpha_k \|P_X \nabla f(x_k)\|^2 + \frac{1}{2} \alpha_k^2 L_{\nabla f} G_d.\end{aligned}$$

Assumptions 3.1, 3.3, 3.4, and 3.6 imply that there exists a constant $C_{\nabla f} > 0$ such that the gradients generated by the sequence of iterates $\{x_k\}_{k \geq 0}$ are bounded, i.e., $\|\nabla f(x_k)\| \;\leq\; C_{\nabla f}$. Finally, from the boundedness of ∇f and inequality (3.9) along with the properties of orthogonal projections, we obtain the desired result. □

We will now introduce the final assumptions that are needed for the convergence result of the nonconvex case. The first of these states that, for Algorithm 1 to converge, the sequence of function values must be bounded below by some minimum value.

Assumption 4.1 *The sequence $\{f(x_k)\}_{k\geq 0}$ is bounded below by f_{inf}.*

Lastly, we require the stepsize to be of decaying type.

Assumption 4.2 *The sequence of decaying stepsizes $\{\alpha_k\}_{k\geq 0}$ satisfies*

$$\sum_{k=0}^{\infty} \alpha_k = \infty \quad \textit{and} \quad \sum_{k=0}^{\infty} \alpha_k^2 < \infty.$$

We can now establish the convergence result for the nonconvex case. We use $\mathbb{E}[\cdot]$ to refer to the *total expectation* of f, namely, the expected value with respect to the joint distribution of all the random vectors ξ_k^{all}.

Theorem 4.3 *Under Assumptions 3.1–3.11 and 4.1, suppose that Algorithm 1 is run with a decaying stepsize sequence that satisfies Assumption 4.2. Then, with $A_K := \sum_{k=0}^{K} \alpha_k$,*

$$\lim_{K\to\infty} \mathbb{E}\left[\sum_{k=0}^{K} \alpha_k \|P_X \nabla f(x_k)\|^2\right] < \infty, \tag{4.3}$$

and therefore

$$\lim_{K\to\infty} \mathbb{E}\left[\frac{1}{A_K}\sum_{k=0}^{K} \alpha_k \|P_X \nabla f(x_k)\|^2\right] = 0. \tag{4.4}$$

Proof. The proof follows [5, Theorem 4.10] closely. Taking the total expectation of (4.2), we have

$$\mathbb{E}\left[f(x_{k+1})\right] - \mathbb{E}\left[f(x_k)\right] \;\leq\; -\alpha_k \mathbb{E}\left[\|P_X \nabla f(x_k)\|^2\right] + \alpha_k^2 C_{\nabla f} C_d + \frac{1}{2}\alpha_k^2 L_{\nabla f} G_d.$$

Summing both sides of this inequality for $k \in \{0, 1, ..., K\}$ and by Assumption 4.1, we have

$$\begin{aligned} f_{\text{inf}} - \mathbb{E}\left[f(x_0)\right] \;&\leq\; \mathbb{E}\left[f(x_{K+1})\right] - \mathbb{E}\left[f(x_0)\right] \\ &\leq\; -\sum_{k=0}^{K} \alpha_k \mathbb{E}\left[\|P_X \nabla f(x_k)\|^2\right] + C_{\nabla f} C_d \sum_{k=0}^{K} \alpha_k^2 + \frac{1}{2} L_{\nabla f} G_d \sum_{k=0}^{K} \alpha_k^2. \end{aligned}$$

Rearranging, we obtain

$$\sum_{k=0}^{K} \alpha_k \mathbb{E}\left[\|P_X \nabla f(x_k)\|^2\right] \;\leq\; \mathbb{E}\left[f(x_0)\right] - f_{\text{inf}} + C_{\nabla f} C_d \sum_{k=0}^{K} \alpha_k^2 + \frac{1}{2} L_{\nabla f} G_d \sum_{k=0}^{K} \alpha_k^2.$$

Assumption 4.2 implies that the right-hand side of this inequality converges to a finite limit when K increases, which proves (4.3). To obtain (4.4), we can divide by A_K as follows:

$$\frac{1}{A_K}\sum_{k=0}^{K} \alpha_k \mathbb{E}\left[\|P_X \nabla f(x_k)\|^2\right] \;\leq\; \frac{\mathbb{E}\left[f(x_0)\right] - f_{\text{inf}}}{A_K} + \frac{C_{\nabla f} C_d}{A_K} \sum_{k=0}^{K} \alpha_k^2 + \frac{L_{\nabla f} G_d}{2A_K} \sum_{k=0}^{K} \alpha_k^2.$$

Taking the limit as $K \to \infty$, and noting Assumption 4.2, we obtain the desired result. □

4.2 Rate in the strongly convex case

In this subsection, we present the convergence rate of the BSG method when f is assumed to be strongly convex. In practice, such a case occurs when the UL objective function f_u is strongly convex and $y(x)$ is an affine function in x. Hence, imposing strong convexity of f is a strong assumption in the sense of assuming in practice that the LL problem is a QP problem. Still, we cover this case for completeness of our convergence theory.

Along with (3.19), we also need the iterates to lie in a bounded set, which could be ensured by the boundedness of X in the BLP formulation.

Assumption 4.3 (Boundedness of the iterates) *The sequence of iterates $\{x_k\}_{k\geq 0}$ yielded by Algorithm 1 is contained in a bounded set.*

Assumption 4.3 implies that there exists a positive constant Θ such that, for any (k_1, k_2), we have

$$\|x_{k_1} - x_{k_2}\| \;\leq\; \Theta \;<\; \infty.$$

Finally, we assume that the true function f is strongly convex.

Assumption 4.4 (Strong convexity of f) *The function f is strongly convex, namely, there exists a constant $c > 0$ such that*

$$f(\bar{x}) \;\geq\; f(x) + \nabla f(x)^\top(\bar{x} - x) + \frac{c}{2}\|\bar{x} - x\|^2 \text{ for all } (\bar{x}, x) \in \mathbb{R}^n \times \mathbb{R}^n. \tag{4.5}$$

A well-known equivalent condition to (4.5) (see, e.g., [44]) is given by

$$(\nabla f(x) - \nabla f(\bar{x}))^\top(x - \bar{x}) \geq c\|x - \bar{x}\|^2 \text{ for all } (x, \bar{x}) \in \mathbb{R}^n \times \mathbb{R}^n. \tag{4.6}$$

Let x_* be the unique minimizer of f on X, which implies that $\nabla f(x_*)^\top(x - x_*) \geq 0$ for all $x \in X$. Therefore, if in (4.6) we choose $x = x_k$ and $\bar{x} = x_*$, we obtain

$$\nabla f(x_k)^\top(x_k - x_*) \geq c\|x_k - x_*\|^2. \tag{4.7}$$

The next theorem proves that under the assumption of strong convexity and decaying stepsize ($\sum_{k=0}^{\infty} \alpha_k = \infty$ and $\sum_{k=0}^{\infty} \alpha_k^2 < \infty$), the sequence of points yielded by Algorithm 1 generates a sequence of f values that decays sublinearly at the rate of $1/k$. The proof of this theorem is given in Appendix C.

Theorem 4.4 *Let Assumptions 3.1–3.11 and 4.3–4.4 hold and x_* be the unique minimizer of f on X. Consider the schema given by Algorithm 1 and assume a decaying step size sequence of the form $\alpha_k = \gamma/k$, where $\gamma \geq 1/(2c)$ is a positive constant. The sequence of iterates yielded by Algorithm 1 satisfies*

$$\begin{aligned}
\mathbb{E}[\|x_k - x_*\|^2] \quad &\leq \frac{\max\{2\,\gamma^2 M(2c\gamma - 1)^{-1}, \|x_0 - x_*\|^2\}}{k}, \\
\mathbb{E}[f(x_k)] - f(x_*) \;&\leq \frac{(L_{\nabla f}/2)\max\{2\,\gamma^2 M(2c\gamma - 1)^{-1}, \|x_0 - x_*\|^2\}}{k},
\end{aligned}$$

where $M = G_d + 2C_d\Theta$.

4.3 Rate in the convex case

In this subsection, we state the convergence rate of the BSG method assuming that f is convex and attains a minimizer x_*. The same comment about the lack of practicality applies to the convex case, i.e., that $y(x)$ would need to be affine, which considerably restricts the choice of LL problems.

Assumption 4.5 (Convexity of f) *Given $(\bar{y}, y) \in \mathbb{R}^m \times \mathbb{R}^m$, the (continuously differentiable) function f is convex in x, namely,*

$$f(\bar{x}) \ \geq \ f(x) + \nabla f(x)^\top (\bar{x} - x) \text{ for all } (\bar{x}, x) \in \mathbb{R}^n \times \mathbb{R}^n. \tag{4.8}$$

Moreover, f attains a minimizer.

The next theorem states that the BSG method exhibits a sublinear convergence rate of $1/\sqrt{k}$, which implies that the convergence is slower than in the strongly convex case (Theorem 4.4). The proof of this theorem is given in Appendix D.

Theorem 4.5 *Let Assumptions 3.1–3.11, 4.3, and 4.5 hold. Consider the schema given by Algorithm 1 and assume a decaying step size of the form $\alpha_k = \bar{\alpha}/\sqrt{k}$, with $\bar{\alpha} > 0$. Given a minimizer x_* of f, the sequence of iterates yielded by Algorithm 1 satisfies*

$$\min_{s=0,\ldots,k} \mathbb{E}[f(x_s)] - f(x_*) \ \leq \ \frac{\frac{\Theta^2}{2\bar{\alpha}} + \bar{\alpha}(G_d M + 2C_d M\Theta)}{\sqrt{k}}.$$

4.4 Imposing a bound on the distance from the LL optimal solution

In this subsection, we want to discuss a way to enforce Assumption 3.9 when using the stochastic gradient (SG) method to solve the LL problem at x_k. We focus on the LL unconstrained case. Given an initial point $\tilde{y}_k^0$ and a sequence of stepsizes $\{\beta_i\}$, such a SG method can be described as

$$\tilde{y}_k^{i+1} \ = \ \tilde{y}_k^i - \beta_i g_y^\ell(x_k, \tilde{y}_k^i, \xi_{k,i}^{\mathrm{S1}}), \quad i = 0, \ldots, i_k. \tag{4.9}$$

We start by introducing the sampling assumptions that are standard in the literature related to the SG method. First, suppose that the stochastic gradient $g_y^\ell(x_k, \tilde{y}_k, \xi_{k,i}^{\mathrm{S1}})$ is unbiased, i.e., $\mathbb{E}_{\xi_{k,i}^{\mathrm{S1}}}[g_y^\ell(x_k, \tilde{y}_k, \xi_{k,i}^{\mathrm{S1}})] = \nabla_y f_\ell(x_k, \tilde{y}_k)$, and there exists a positive constant $Q > 0$ such that $\mathbb{E}_{\xi_{k,i}^{\mathrm{S1}}}[\|g_y^\ell(x_k, \tilde{y}_k, \xi_{k,i}^{\mathrm{S1}})\|^2] \leq Q^2$. Also, suppose that f_ℓ is strongly convex in the y variables with constant μ with a unique minimizer $y(x_k)$.

Recalling that the convergence rate of the SG method (4.9) with decaying stepsize is $\mathcal{O}(1/\sqrt{i})$, and by choosing i_k equal to k^2, one guarantees the existence of a positive constant C_y such that Assumption 3.9 holds. In fact, by choosing a decaying step size sequence $\{\beta_i\}$ given by $\beta_i = \gamma/i$, where $\gamma \geq 1/(2\mu)$ is a positive constant, and under the classical assumptions stated in the previous paragraph, from [43, Equation (2.9)] it follows that the choice $i_k = k^2$ implies (with $\tilde{y}_k = \tilde{y}_k^{i_k+1}$)

$$\mathbb{E}_{\xi_k^{\mathrm{S1}}}[\|\tilde{y}_k - y(x_k)\|^2] \ \leq \ \frac{\max\{\gamma^2 Q(2\mu\gamma - 1)^{-1}, \|\tilde{y}_k^0 - y(x_k)\|^2\}}{k^2}.$$

Such a result also holds in the LL constrained case when the scheme (4.9) incorporates a projection onto $Y(x_k)$, as long as $Y(x_k)$ is a closed convex set. We refer the reader to the discussion in [31, Appendix B], which applies to bilevel problems with an LL strongly convex objective function and LL linear constraints. It is also possible to obtain an inequality like (3.9) for the LL constrained case when using a primal-dual stochastic method [65].

4.5 Imposing a bound on dynamic sampling

In this subsection, we want to mention a dynamic sampling strategy to enforce the inequality in Assumption 3.10 in both the LL unconstrained and constrained cases. For the sake of simplicity, we will omit the subscript k in this subsection. Such a dynamic sampling strategy allows reducing the level of noise by increasing the size of the batch. Recalling the definition of $D(x, w)$ given in (2.2) and (2.10), the definition of $D(x, w, \xi)$ given in (2.3) and (2.11), and the unified notation introduced in Subsection 2.3, let us denote

$$\begin{aligned} D(x, w) \quad &= (a(x, y), b(x, y), A(x, w), B(x, w)), \\ D(x, w, \xi) &= (a(x, y, \vartheta^u), b(x, y, \vartheta^u), A(x, w, \gamma), B(x, w, \gamma)), \end{aligned}$$

where γ is either ϑ^ℓ (in the LL unconstrained case) or ς^ℓ (in the LL constrained case). Let us assume that the stochastic estimates $a(x, y, \vartheta^u)$, $b(x, y, \vartheta^u)$, $A(x, w, \gamma)$, and $B(x, w, \gamma)$ are normally distributed with means $a(x, y)$, $b(x, y)$, $A(x, w)$, and $B(x, w)$, respectively, and variances σ_a^2, σ_b^2, σ_A^2, and σ_B^2, respectively. Such an assumption implies that the stochastic estimates in $D(x, w, \xi)$ are unbiased estimates of the corresponding true gradients, Hessians, and Jacobians in $D(x, w)$.

To increase the accuracy of the stochastic estimates in $D(x, w, \xi)$, we can choose larger batch sizes, which we denote by n_a, n_b, n_A, and n_B. Let $\bar{a}(x, y, \vartheta^u) = (1/n_a) \sum_{r=1}^{n_a} a(x, y, (\vartheta^u)_r)$ be the mini-batch stochastic estimate for $a(x, y)$, where $\{(\vartheta^u)_r\}_{r=1}^{n_a}$ are values sampled from the distribution of ϑ^u. It is known that (for details, see, for instance, [36, Section 5.3])

$$\mathbb{E}_{\xi_k^{\text{all}}}[\|a(x, y) - \bar{a}(x, y, \vartheta^u)\|] \;\le\; \frac{\sigma_a \sqrt{n}}{\sqrt{n_a}}.$$

One can repeat similar arguments for b, A, and B, and their corresponding mini-batch stochastic estimates $\bar{b}$, $\bar{A}$, and $\bar{B}$, respectively. Let us denote

$$\bar{D}(x, w, \xi) = (\bar{a}(x, y, \vartheta^u), \bar{b}(x, y, \vartheta^u), \bar{A}(x, w, \gamma), \bar{B}(x, w, \gamma)).$$

From the equivalence of norms, there exists a positive constant $\hat{C}$ such that

$$\begin{aligned} \|D(x, w) - \bar{D}(x, w, \xi)\| \;\le\; & \hat{C}(\|a(x, y) - \bar{a}(x, y, \vartheta^u)\| + \|b(x, y) - \bar{b}(x, y, \vartheta^u)\|) \\ & + \hat{C}(\|A(x, w) - \bar{A}(x, w, \gamma)\| + \|B(x, w) - \bar{B}(x, w, \gamma)\|). \end{aligned}$$

Taking expectations with respect to ξ_k^{all} on both sides, one obtains

$$\mathbb{E}_{\xi_k^{\text{all}}}[\|D(x, w) - \bar{D}(x, w, \xi)\|] \;\le\; \hat{C}\left(\frac{\sigma_a}{\sqrt{n_a}} + \frac{\sigma_b}{\sqrt{n_b}} + \frac{\sigma_A}{\sqrt{n_A}} + \frac{\sigma_B}{\sqrt{n_B}}\right)\sqrt{n}. \tag{4.10}$$

To guarantee that Assumption 3.10 holds, we need to choose mini-batch sizes n_a, n_b, n_A, and n_B and sample standard deviations σ_a, σ_b, σ_A, and σ_B such that the right-hand side in (4.10) is less than or equal to $C_D\, \alpha_k$. Therefore, when α_k decreases, the dynamic sampling strategy would increase the mini-batch sizes.

5 Numerical experiments

All code was written in Python and the experimental results were obtained on a desktop computer (32GB of RAM, Intel(R) Core(TM) i9-9900K processor running at 3.60GHz). We averaged all the results over 10 trials by using different random seeds.

5.1 Our practical BSG methods

A major difficulty in the adjoint formulas (1.2) and (2.8) is the use of second-order derivatives of f_ℓ and $\mathcal{L}_\ell$, respectively, and the need to solve the adjoint equation or use a truncated Neumann series, which prevents its application to large-scale ML application problems. We propose two approaches to get around this problem, which lead to two different practical versions of the BSG method, referred to as BSG-N-FD and BSG-1. In the numerical experiments considered for both LL unconstrained and constrained BLPs, we are mainly interested in testing these two practical implementations (Algorithms 2 and 3 in Subsections 5.1.1 and 5.1.2 below, respectively) as opposed to the standard BSG method (Algorithm 1). We will consider two types of test problems: synthetic quadratic bilevel problems and continual learning problems. Since these test problems do not have upper-level constraints (i.e., $X = \mathbb{R}^n$), we have omitted the use of the orthogonal projection operator P_X in all of the algorithms.

In the numerical experiments for the synthetic problems, we will also test the BSG method with stochastic Hessians, where the (negative) BSG direction $d(x_k, \tilde{y}_k, \xi_k)$ is calculated from (2.1) or (2.9). This version is referred to as BSG-H. In the LL unconstrained case, BSG-H applies the linear conjugate gradient (CG) method [45] to solve the adjoint system $H^\ell_{yy}(x_k, \tilde{y}_k, \vartheta^\ell_k)\lambda = g^u_y(x_k, \tilde{y}_k, \vartheta^u_k)$ until non-positive curvature is detected. In the LL constrained case, BSG-H solves the adjoint system $\mathcal{G}_v(x_k, \tilde{v}_k, \varsigma^\ell_k)\lambda = L\, g^u_y(x_k, \tilde{y}_k, \vartheta^u_k)$ by applying the GMRES method [52]. Note that we only include BSG-H in the experiments for the sake of comparison. For very large problems, one must use BSG-N-FD or BSG-1.

In the BSG-N-FD, BSG-1, and BSG-H versions of the BSG method for the LL unconstrained case, we will apply the SG method (4.9) to the LL problem for a certain budget i_k of iterations, obtaining an approximation $\tilde{y}_k$ to the LL optimal solution $y(x_k)$. To obtain an approximation $\tilde{w}_k$ to $w(x_k)$, given x_k, in the LL constrained case, BSG-N-FD, BSG-1, and BSG-H will first determine an approximation $\tilde{y}_k$ to $y(x_k)$ by minimizing the following exact penalty function over y

$$\Phi(x_k, y; \mu) = f_\ell(x_k, y) + \frac{1}{\mu}\sum_{i\in I} \max\{0, c_i(x_k, y)\} + \frac{1}{\mu}\sum_{i\in E} |c_i(x_k, y)|\,, \tag{5.1}$$

where μ is a penalty parameter and the functions c_i, with $i \in I \cup E$, are the LL constraints defined in Subsection 2.2. We recall that for sufficiently small and positive values of μ, the minimization of such an unconstrained problem will yield the optimal solution $y(x_k)$ of the constrained LL problem [45]. To minimize (5.1), which is a nonsmooth function, the stochastic subgradient method will be applied. Then, given x_k and $\tilde{y}_k$, to determine approximations $(\tilde{z}_I, \tilde{z}_E)$ to the optimal multipliers $(z_I(x_k), z_E(x_k))$, the linear CG method will be applied to solve the KKT system $G(x_k, (\tilde{y}_k, z_I, z_E)) = 0$ for the variables z_I and z_E, where G is the vector function introduced in Subsection 2.2.

For the solution of the LL problem, we consider an inexact scheme, denoted as inc. acc., which consists of obtaining $\tilde{y}_k$ by taking multiple steps of the stochastic gradient method applied to f_ℓ ($i_k \geq 1$, $\forall k$, in (4.9), for the LL unconstrained case) or stochastic subgradient method

applied to (5.1) (for the LL constrained case). In particular, the number of steps of the stochastic gradient/subgradient method increases by 1 every time the difference of the UL objective function between two consecutive iterations is less than a given threshold, thus leading to an increasing accuracy strategy. In such an inexact scheme, $\tilde{y}_k$ is determined by using the approximation $\tilde{y}_{k-1}$ obtained at the previous iteration as a starting point. In the ML community, iterative schemes with multiple LL steps, like the inc. acc. strategy above, are referred to as double-loop schemes [9, 29].

5.1.1 BSG-N-FD

Our first proposed method, BSG-N-FD, solves the adjoint system by using an iterative method where each Hessian vector product is approximated with a finite-difference (FD) scheme. In particular, in the LL unconstrained case, the adjoint equation $\nabla^2_{yy} f_\ell \, \lambda = \nabla_y f_u$ is solved for the adjoint variables λ by using the linear CG method, with $\nabla^2_{yy} f_\ell \, \lambda$ being approximated as follows:

$$\nabla^2_{yy} f_\ell(x_k, y_k)\lambda \;\approx\; \frac{\nabla_y f_\ell(x_k, y_k^+) - \nabla_y f_\ell(x_k, y_k^-)}{2\varepsilon}, \tag{5.2}$$

where $y_k^\pm \;=\; y_k \pm \varepsilon\lambda,$ with $\varepsilon > 0$. Then, the adjoint gradient is calculated from

$$\nabla f \;\approx\; \nabla_x f_u - \nabla^2_{xy} f_\ell \, \lambda, \tag{5.3}$$

where $\nabla^2_{xy} f_\ell \, \lambda$ is approximated using an additional FD scheme. In practice, we use a scaling parameter value of $\varepsilon = 0.1$.

In the LL constrained case, we can use FD schemes similar to the ones in the LL unconstrained case to approximate the Jacobian vector products when solving the adjoint equation and computing the adjoint gradient. In particular, the adjoint equation $\nabla_v G\lambda = L\nabla_y f_u$ is solved for the adjoint variables $\lambda = (\lambda_y, \lambda_I, \lambda_E)^\top$ by using the GMRES method, with $\nabla_v G\lambda$ being approximated as follows:

$$\nabla_v G\lambda \;\approx\; \begin{pmatrix} \overline{\nabla^2_{yy}\mathcal{L}_\ell\lambda_y} + (z_I^\top \circ \nabla_y c_I)\lambda_I + \nabla_y c_E \lambda_E \\ \nabla_y c_I^\top \lambda_y + C_I \lambda_I \\ \nabla_y c_E^\top \lambda_y \end{pmatrix}, \tag{5.4}$$

where

$$\overline{\nabla^2_{yy}\mathcal{L}_\ell\lambda_y} \;=\; \frac{\nabla_y \mathcal{L}_\ell(x_k, y_k^+, (z_I)_k, (z_E)_k) - \nabla_y \mathcal{L}_\ell(x_k, y_k^-, (z_I)_k, (z_E)_k)}{2\varepsilon}$$

and $y_k^\pm = y_k \pm \varepsilon\lambda_y$, with $\varepsilon > 0$. Then, the adjoint gradient is calculated from

$$\nabla f \;\approx\; \nabla_x f_u - \nabla_x G\lambda, \tag{5.5}$$

where $\nabla_x G\lambda$ is approximated using an additional FD scheme. In practice, what worked better for us was again $\varepsilon = 0.1$.

We denote the algorithm corresponding to this approach as BSG-N-FD, where the "N" stands for the Newton-type system given by the adjoint equation, and the "FD" for the finite-difference approximations we employ. Its schema is described in Algorithm 2. In the practical BSG-N-FD method, we will use the stochastic data defined by (2.3) and (2.11).

Algorithm 2 BSG-N-FD Method

Input: $(x_0, \tilde{w}_0)$, $\{\alpha_k\}_{k\geq 0} > 0$.

For $k = 0, 1, 2, \ldots$ **do**

Step 1. Obtain an approximation $\tilde{w}_k$ to the LL optimal solution $w(x_k)$.

Step 2. Compute $d(x_k, \tilde{w}_k, \xi_k)$ by drawing the stochastic gradients and/or Jacobians from (5.2)–(5.3) for the LL unconstrained case or (5.4)–(5.5) for the LL constrained case.

Step 3. Compute $x_{k+1} = x_k + \alpha_k\, d(x_k, \tilde{w}_k, \xi_k)$.

End do

5.1.2 BSG-1

Our second proposed method, BSG-1, approximates the second-order derivatives in the adjoint formulas (1.2) and (2.8) with outer products of the corresponding gradients, i.e.,

$$\nabla^2_{xy} f_\ell \;\simeq\; \nabla_x f_\ell \nabla_y f_\ell^\top \quad \text{and} \quad \nabla^2_{yy} f_\ell \;\simeq\; \nabla_y f_\ell \nabla_y f_\ell^\top, \tag{5.6}$$

$$\nabla^2_{yx} \mathcal{L}_\ell \;\simeq\; \nabla_y \mathcal{L}_\ell \nabla_x \mathcal{L}_\ell^\top \quad \text{and} \quad \nabla^2_{yy} \mathcal{L}_\ell \;\simeq\; \nabla_y \mathcal{L}_\ell \nabla_y \mathcal{L}_\ell^\top. \tag{5.7}$$

As mentioned in Subsection 1.3, such approximations are inspired by Gauss-Newton methods for nonlinear least-squares problems, where the Hessian matrix of the objective function $\sum_{i=1}^p (r_i - a_i)^2$ (in which each r_i is a scalar function and a_i a scalar) is approximated by $\sum_{i=1}^p \nabla r_i \nabla r_i^\top$, and also from the fact that the empirical risk of misclassification in ML is often a sum of nonnegative terms matching a function to a scalar which can then be considered in a least-squares fashion [4, 23]. In the numerical experiments, the rank-1 approximations are observed to perform well when the LL function f_ℓ has a Gauss-Newton structure, such as the binary cross-entropy loss function used for the continual learning instances (see Subsection 5.4).

In the LL unconstrained case, the resulting approximate adjoint equation $(\nabla_y f_\ell \nabla_y f_\ell^\top)\,\lambda = \nabla_y f_u$ is most likely infeasible, and we suggest solving it in the least-squares sense. One solution is $\lambda = \nabla_y f_u / (\nabla_y f_\ell^\top \nabla_y f_\ell)$. Plugging this and $\nabla^2_{xy} f_\ell \simeq \nabla_x f_\ell \nabla_y f_\ell^\top$ in the adjoint formula (1.2) gives rise to our practical BSG-1 calculation

$$\nabla f \;\simeq\; \nabla_x f_u - \frac{\nabla_y f_\ell^\top \nabla_y f_u}{\nabla_y f_\ell^\top \nabla_y f_\ell} \nabla_x f_\ell. \tag{5.8}$$

This approximate BSG allows us to use the adjoint formula without computing Hessians or even Hessian-vector products, which is prohibitively expensive for the large bilevel problems arising in ML applications.

In the LL constrained case, we can use the outer products (5.7) to approximate the second-order derivatives of $\mathcal{L}_\ell$ in the Jacobian matrices $\nabla_x G^\top$ and $\nabla_v G^\top$, introduced in (2.6), and obtain corresponding approximate Jacobians $\tilde{G}_x^\top$ and $\tilde{G}_v^\top$, respectively. The resulting approximate adjoint equation is given by $\tilde{G}_v \tilde{\lambda} = L \nabla_y f_u$, where L is the matrix used in (2.8), and can be solved by using an iterative method for non-symmetric linear systems. Plugging a solution $\tilde{\lambda}$ into $\nabla_x f_u - \tilde{G}_x\, \tilde{\lambda}$, we obtain the practical BSG-1 calculation

$$\nabla f \;\simeq\; \nabla_x f_u - \tilde{G}_x\, \tilde{\lambda}, \quad \text{where} \quad \tilde{G}_v \tilde{\lambda} = L \nabla_y f_u. \tag{5.9}$$

Both of these rank-1 approaches for the LL unconstrained and constrained cases will be referred to as BSG-1, the "1" standing for first-order rank-1 approximations of the Hessian and Jacobian matrices. In the practical BSG-1 method, we will use the stochastic data defined by (2.3) and (2.11).

Algorithm 3 BSG-1 Method

Input: $(x_0, \tilde{w}_0)$, $\{\alpha_k\}_{k\geq 0} > 0$.

For $k = 0, 1, 2, \ldots$ **do**

Step 1. Obtain an approximation $\tilde{w}_k$ to the LL optimal solution $w(x_k)$.

Step 2. Compute $d(x_k, \tilde{w}_k, \xi_k)$ by drawing the stochastic gradients and/or Jacobians from (5.8) for the LL unconstrained case or (5.9) for the LL constrained case.

Step 3. Compute $x_{k+1} = x_k + \alpha_k\, d(x_k, \tilde{w}_k, \xi_k)$.

End do

5.2 DARTS

DARTS was proposed in [34] for the solution of stochastic BLPs arising from NAS, and was briefly introduced in Section 1.2. Only the LL unconstrained case ($Y(x) = \mathbb{R}^m$) has been considered. To avoid the computation of the second-order derivatives in (1.5), DARTS approximates the matrix-vector product $\nabla^2_{xy} f_\ell(x_k, y_k)\nabla_y f_u(x_k, \tilde{y}_k)$ by a finite-difference scheme [34]:

$$\nabla^2_{xy} f_\ell(x_k, y_k)\nabla_y f_u(x_k, \tilde{y}_k) \;\approx\; \frac{\nabla_x f_\ell(x_k, y_k^+) - \nabla_x f_\ell(x_k, y_k^-)}{2\varepsilon},$$

where

$$y_k^\pm \;=\; y_k \pm \varepsilon \nabla_y f_u(x_k, \tilde{y}_k) \quad \text{with} \quad \varepsilon \;=\; 0.01/\|\nabla_y f_u(x_k, \tilde{y}_k)\|. \tag{5.10}$$

Algorithm 4 reports the schema of DARTS for the stochastic setting. In Step 1, a single step of SG (with fixed stepsize η) is applied to the LL problem to obtain an approximation $\tilde{y}_k$ to the LL optimal solution. Then, in Step 2, the UL variables are updated by moving along the "approximated" descent direction using a stepsize α_k.

Algorithm 4 Differentiable Architecture Search (DARTS)

Input: $(x_0, y_0) \in \mathbb{R}^n \times \mathbb{R}^m$, $\{\alpha_k\}_{k\geq 0} > 0$, $\eta > 0$.

For $k = 0, 1, 2, \ldots$ **do**

Step 1. Compute $\tilde{y}_k = y_k - \eta\, g_y^\ell(x_k, y_k, \vartheta_k^\ell)$.

Step 2. Compute $x_{k+1} = x_k - \alpha_k \big(g_x^u(x_k, \tilde{y}_k, \vartheta_k^u) - \frac{\eta}{2\varepsilon}(g_x^\ell(x_k, y_k^+, \vartheta_k^\ell) - g_x^\ell(x_k, y_k^-, \vartheta_k^\ell))\big)$, with $y_k^\pm$ and ε as in (5.10), with $g_y^u(x_k, \tilde{y}_k, \vartheta_k^u)$ instead of $\nabla_y f_u(x_k, \tilde{y}_k)$, and set $y_{k+1} = \tilde{y}_k$.

End do

5.3 Numerical results for synthetic quadratic bilevel problems

We first report results for a "synthetic" bilevel problem, where both levels are defined by quadratic objective functions. Given $h_1 \in \mathbb{R}^n$, $h_2 \in \mathbb{R}^m$, symmetric positive definite matri-

ces $H_2 \in \mathbb{R}^{n\times n}$ and $H_3 \in \mathbb{R}^{m\times m}$, and matrices $H_1 \in \mathbb{R}^{n\times m}$ and $H_4 \in \mathbb{R}^{m\times n}$, we consider the following problem

$$\begin{aligned} \min_{x\in\mathbb{R}^n} \; & f_u(x,y) \;=\; h_1^\top x + h_2^\top y + \frac{1}{2}x^\top H_1 y + \frac{1}{2}x^\top H_2 x \\ \text{s.t.} \quad & y \in \operatorname*{argmin}_{y\in Y(x)} \; f_\ell(x,y) \;=\; \frac{1}{2}y^\top H_3 y - y^\top H_4 x, \end{aligned} \tag{5.11}$$

where the set $Y(x)$ used for the numerical experiments will be specified in Subsections 5.3.1–5.3.3 below. In particular, Subsection 5.3.1 focuses on the LL unconstrained case and Subsections 5.3.2–5.3.3 address the LL constrained case.

For all of the algorithms, we used the best UL and LL fixed stepsizes (i.e., α^u and α^ℓ, respectively) found by performing a grid search over the set $\{10^{-s_u} \mid s_u \in \{\underline{s}_u, \ldots, \bar{s}_u\}\}$ for the UL stepsize and $\{10^{-s_\ell} \mid s_\ell \in \{\underline{s}_\ell, \ldots, \bar{s}_\ell\}\}$ for the LL stepsize. We used independent bounds $\underline{s}_u, \bar{s}_u, \underline{s}_\ell, \bar{s}_\ell$ for each algorithm with the goal of selecting stepsize values that capture their best performances on the different types of problems. Using a single arbitrary grid to test all of the algorithms would have led to stepsizes biased toward one set of algorithms over the others. The domain of possible values for the bounds $\underline{s}_u, \bar{s}_u, \underline{s}_\ell, \bar{s}_\ell$ was restricted to the set $\{1, \ldots, 8\}$. We chose the values of such bounds to include two to three consecutive values in the grid searches for both s_u and s_ℓ.

With the exception of DARTS in the LL unconstrained case, we utilized the LL inc. acc. strategy (introduced in Subsection 5.1) for the rest of the algorithms, with an f_u difference threshold for increasing the number of LL iterations equal to 10^{-1} (and a maximum limit of 30 LL iterations). In all the figures included in this subsection, we plot the true function f of the BLP ($f(x_k) = f_u(x_k, y(x_k))$). The number of UL iterations and running time were both used as metrics for the comparison of the algorithms. In the LL constrained case, in accordance with the procedure described in Subsection 5.1, we obtain approximate Lagrange multipliers at each iteration by solving the corresponding KKT system with the linear conjugate gradient method (with a maximum number of iterations equal to 3 and tolerance equal to 10^{-4}). The system in (5.9) is solved by using the GMRES method (with a maximum number of 3 iterations and tolerance equal to 10^{-4}).

5.3.1 Results for the LL unconstrained case

In the numerical experiments for the LL unconstrained version ($Y(x) = \mathbb{R}^m$) of problem (5.11), we considered a dimension of 300 at both the upper and lower levels (i.e., $n = m = 300$), with H_2 and H_3 randomly generated and H_1 and H_4 set equal to the identity matrix. Note that problem (5.11) is deterministic. To investigate the numerical performance of the stochastic methods considered in the experiments, we computed stochastic gradient and Hessian estimates by adding Gaussian noise with mean 0 to each corresponding deterministic gradient (i.e., $\nabla_x f_u$, $\nabla_y f_u$, $\nabla_x f_\ell$, $\nabla_y f_\ell$) and Hessian (i.e., $\nabla^2_{xy} f_\ell$, $\nabla^2_{yy} f_\ell$). In the stochastic case, the standard deviations for the stochastic estimates of the gradients and Hessians were set to 5 and 0.05, respectively.

In the experiments for this subsection, we compared BSG-N-FD, BSG-1, and BSG-H against DARTS and StocBiO, introduced in Subsection 1.2. Regarding the grid searches for the stepsizes α_u and α_ℓ, in the deterministic case, we used $\underline{s}_u = \underline{s}_\ell = 2$ and $\bar{s}_u = \bar{s}_\ell = 4$ for BSG-N-FD, BSG-H, BSG-1, and StocBiO, and $\underline{s}_u = 3$, $\bar{s}_u = 5$, $\underline{s}_\ell = 2$, and $\bar{s}_\ell = 4$ for DARTS. In the stochastic case, we used $\underline{s}_u = \underline{s}_\ell = 2$ and $\bar{s}_u = \bar{s}_\ell = 4$ for BSG-N-FD, BSG-1, and StocBiO,

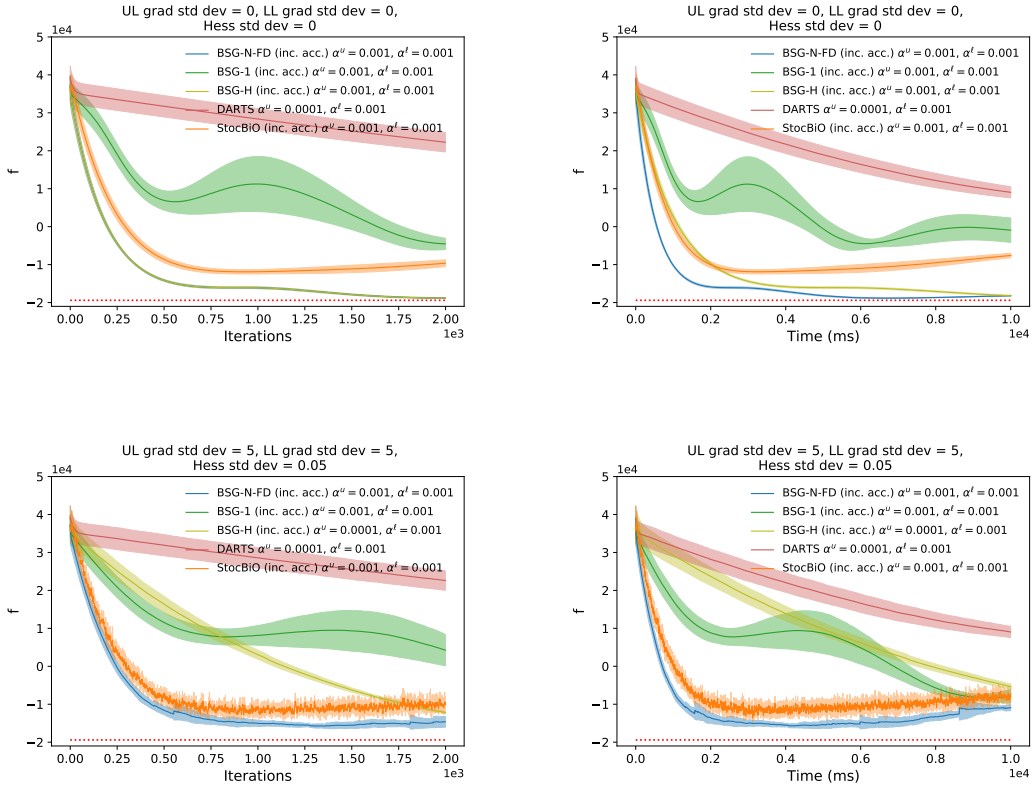

Figure 1: Numerical results of the BSG-N-FD, BSG-H, BSG-1, DARTS, and StocBiO algorithms on problem (5.11) for the LL unconstrained case in terms of both iterations and time (in milliseconds).

and $\underline{s}_u = 3$, $\bar{s}_u = 5$, $\underline{s}_\ell = 2$, and $\bar{s}_\ell = 4$ for BSG-H and DARTS. For StocBiO, we set the constant C_0 introduced in Subsection 1.2 to 0.05 and the parameter q introduced in Subsection 2.5 to 2, which led to the best results.

Starting with the deterministic results displayed in the top two plots of Figure 1, we can see that both BSG-N-FD and BSG-H (these methods completely overlap in the iterations plot) clearly outperform all the other methods in terms of both iterations and time, with StocBiO performing slightly worse. Note that compared to BSG-N-FD, BSG-H is less efficient in terms of time due to the high computational cost of computing Hessian matrices. As somewhat expected, due to the lack of Gauss-Newton structure in problem (5.11), BSG-1 performs worse than the other versions of the BSG method, but it is still able to yield a decrease in the true function and outperform DARTS, which has the worst performance out of the five methods. It bears mentioning that only BSG-N-FD and BSG-H are able to achieve the optimal value of the true function f (represented by the red dotted horizontal lines in all the plots of Figure 1) within the iteration and time limit used.

We will now focus on the results for the stochastic case displayed in the bottom two plots of

Figure 1. Firstly, BSG-N-FD still performs the best in terms of both iterations and time, closely followed by StocBiO. BSG-H is able to achieve a similar function value to those achieved by BSG-N-FD and StocBiO, but after several more iterations and much more time. The decrease in the performance of BSG-H can be explained by sample sizing of the stochastic Hessian matrices, and it is well known that Hessians require more samples than gradients in the presence of noise [5, Section 6.1.1]. In particular, BSG-H is very sensitive to the value of the Hessian standard deviation, and its performance significantly degrades for values much larger than 0.05. Finally, BSG-1 and DARTS seem to be relatively robust to the noise and exhibit similar performance to the deterministic case in terms of iterations. However, in terms of time, BSG-1 outperforms BSG-H and is even able to achieve a similar level of accuracy to BSG-N-FD and StocBiO.

5.3.2 Results for the LL linearly constrained case

In the experiments for the LL linearly constrained version of problem (5.11), the LL constraint set $Y(x) = Y$ was defined by the following $|I|$ linear inequality constraints in y

$$Y \;=\; \{y \mid Wy \;\leq\; s\}, \tag{5.12}$$

where $W \in \mathbb{R}^{|I|\times m}$ and $s \in \mathbb{R}^{|I|}$ were both randomly generated according to a uniform distribution, respectively. In this section, we focus on LL linear constraints in y because this allows us to compare the performance of the BSG algorithms developed in this paper against SIGD, which is only designed for handling these types of constraints (as mentioned in Subsection 1.2). We again considered a dimension of 300 at both the upper and lower levels (i.e., $n = m = 300$) along with $|I| = 50$ constraints, with H_1, H_2, H_3, and H_4 chosen in the same manner as in Subsection 5.3.1.

In the stochastic case, we added Gaussian noise with mean 0 to the gradients and Hessians as in Subsection 5.3.1 and also to the Jacobians $\nabla_x c_I$, $\nabla_x c_E$, $\nabla_y c_I$, and $\nabla_y c_E$. Note that we did not add noise to $\nabla^2_{yx} c_i$ and $\nabla^2_{yy} c_i$, with $i \in I \cup E$, because they are null matrices in the LL linearly constrained case. The value of the standard deviation was chosen from $\{0.005, 0.05\}$ (referred to as the "low" and "high" values, respectively) for the stochastic Hessian matrices, $\nabla^2_{xy} f_\ell$ and $\nabla^2_{yy} f_\ell$, and was set to 0.5 for all the other stochastic estimates. Using two different values for the standard deviation of the stochastic Hessian allows us to demonstrate the impact of different noisy Hessian estimates on the performance of BSG-H and SIGD, which use second-order derivatives. Regarding the stepsizes α_u and α_ℓ, in the deterministic case, we used $\underline{s}_u = 2$, $\bar{s}_u = 4$, $\underline{s}_\ell = 3$, and $\bar{s}_\ell = 5$ for BSG-N-FD and BSG-H and $\underline{s}_u = \underline{s}_\ell = 2$ and $\bar{s}_u = \bar{s}_\ell = 4$ for SIGD. For the stochastic case with Hessian standard deviation equal to 0.005, we used the same UL and LL stepsizes for SIGD as in the deterministic case. However, we changed to values of $\underline{s}_\ell = 7$ and $\bar{s}_\ell = 8$ for BSG-N-FD along with $\underline{s}_\ell = 3$ and $\bar{s}_\ell = 5$ for BSG-H. When the Hessian standard deviation was equal to 0.05, we used the same UL and LL stepsizes as in the deterministic case, with the exception of $\underline{s}_u = 5$ and $\bar{s}_u = 7$ for SIGD. We do not include any results for BSG-1 and for BSG-H when the Hessian standard deviation is 0.05 because we were unable to find stepsizes that allowed the algorithms to converge.

Starting with the deterministic results displayed in the top two plots of Figure 2, we can clearly see that both BSG-N-FD and BSG-H outperform SIGD in terms of iterations and time. In fact, BSG-N-FD and BSG-H yield the exact same performance in terms of iterations, resulting in overlapping lines.

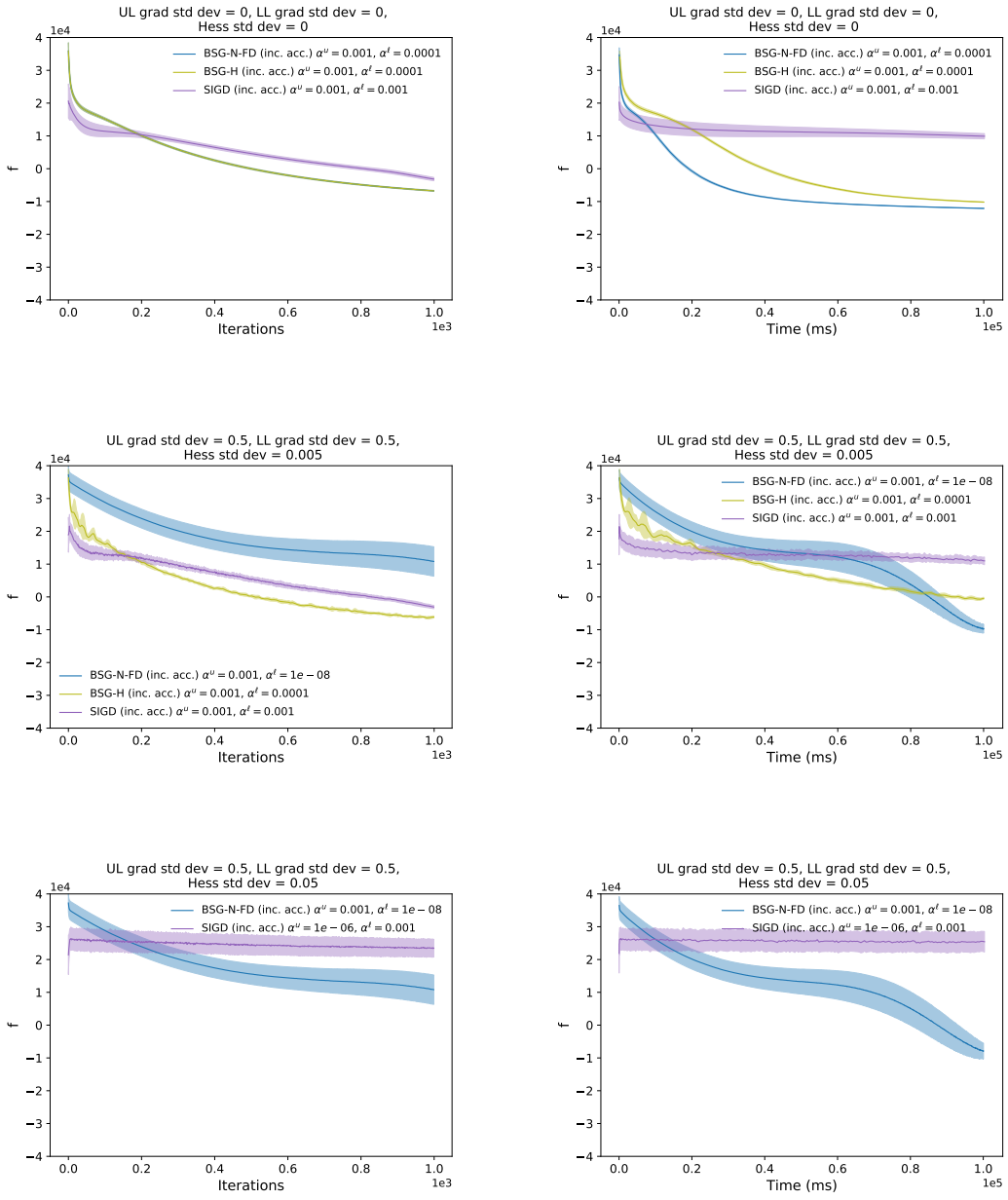

Figure 2: Numerical results of the BSG-N-FD, BSG-H, and SIGD algorithms on problem (5.11) with linear constraints in y defined by (5.12) in terms of both iterations and time (in milliseconds).

In the stochastic setting with low Hessian standard deviation displayed in the two middle plots of Figure 2, we can see that BSG-H is still able to outperform SIGD both in terms of iterations and time, despite its inferior performance compared to the deterministic case. Although BSG-N-FD has the worst performance here in terms of iterations, it is able to outperform both BSG-H and SIGD in the long run due to its superior efficiency in terms of time. Looking now at the setting with high Hessian standard deviation displayed in the two bottom plots of Figure 2, we can see that BSG-N-FD is clearly the superior method. Similar to the behavior of BSG-H, SIGD is no longer able to converge. By contrast, BSG-N-FD is less affected by the noise, yielding the best performance.

5.3.3 Results for the LL quadratically constrained case

In the experiments for the LL quadratically constrained version of problem (5.11), the LL constraint set $Y(x)$ was defined by the following $|I|$ quadratic inequality constraints

$$Y(x) \;=\; \begin{cases} y^\top Q_1^{(1)} y \;+\; x^\top Q_2^{(1)} y \;\leq\; s^{(1)}, \\ y^\top Q_1^{(2)} y \;+\; x^\top Q_2^{(2)} y \;\leq\; s^{(2)}, \\ \qquad\qquad \vdots \\ y^\top Q_1^{(|I|)} y \;+\; x^\top Q_2^{(|I|)} y \;\leq\; s^{(q)}, \end{cases} \tag{5.13}$$

where $Q_1^{(i)} \in \mathbb{R}^{m\times m}$, $Q_2^{(i)} \in \mathbb{R}^{n\times m}$, and $s^{(i)} \in \mathbb{R}$, for all $i \in \{1, 2, ..., |I|\}$, were all randomly generated according to a uniform distribution. We again considered a dimension of 300 at both the upper and lower levels (i.e., $n = m = 300$) along with $|I| = 5$ constraints, with H_1, H_2, H_3, and H_4 chosen in the same manner as in subsection 5.3.1.

We now present numerical results for the BSG methods developed in this paper on a deterministic and two stochastic versions of problem (5.11) with constraints defined by (5.13), again testing two different levels of Hessian noise (with standard deviation values chosen from $\{0.005, 0.5\}$) as in Subsection 5.3.2. To the best of our knowledge, there do not exist any other bilevel stochastic algorithms that can handle general nonlinear constraints in the LL problem, specifically quadratic constraints in this case, and as a result, the following numerical experiments are the first for this type of problem. We added noise to each gradient, Jacobian, and Hessian (including $\nabla^2_{xy} c_i$ and $\nabla^2_{yy} c_i$, for all $i \in I \cup E$), as described in Subsection 5.3.1. Regarding the stepsizes α_u and α_ℓ, we used $\underline{s}_u = 2$, $\bar{s}_u = 4$, $\underline{s}_\ell = 6$, and $\bar{s}_\ell = 8$ for both BSG-N-FD and BSG-H in both the deterministic and stochastic cases.

Starting with the deterministic results displayed in the top two plots of Figure 3, we can clearly see that BSG-N-FD and BSG-H have the same exact performance in terms of iterations and time. It would be expected that BSG-N-FD yields better results in terms of time due to the efficiency we noted in Subsection 5.3.2. In fact, this still holds true, except that this time the differences are not as substantial, and they would only become more visually apparent when allowing the algorithms to run for much longer.

In the stochastic setting with a low level of Hessian noise (the middle two plots), we notice that BSG-N-FD is impacted by the noise, while BSG-H is still able to retain almost the same behavior as in the deterministic case. Although BSG-H looks very favorable here, the stochastic setting with a high level of Hessian noise (the bottom two plots) shows similar results to the linearly constrained case in Subsection 5.3.2 (bottom two plots of Figure 2). Specifically, the

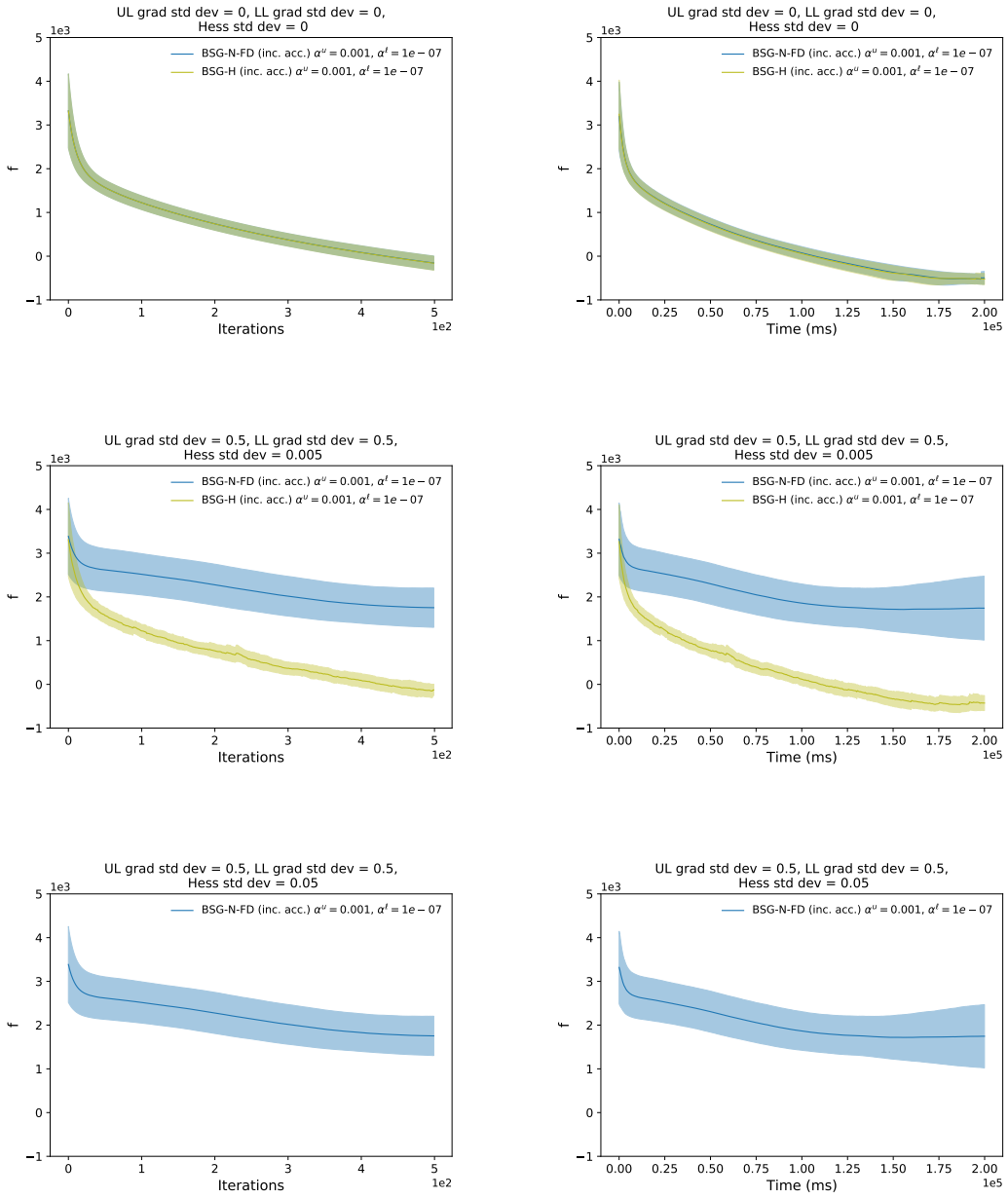

Figure 3: Numerical results of the BSG-N-FD and BSG-H algorithms on problem (5.11) with quadratic constraints defined by (5.13) in terms of both iterations and time (in milliseconds).

performance of BSG-N-FD remains unchanged, while we were not able to find stepsize values that allowed BSG-H to converge.

5.4 Continual learning

We are going to use instances of Continual Learning (CL) as practical stochastic bilevel problems to test the performance of BSG-N-FD, BSG-1, StocBiO, and DARTS. CL was briefly described in Section 1, and is now introduced in more detail. Let us denote a whole features/labels dataset by $\mathcal{D} = \{(\mathbf{u}_j, \mathbf{v}_j),\, j \in \{1, \ldots, N\}\}$, consisting of N pairs of a feature vector $\mathbf{u}_j$ and the corresponding true label $\mathbf{v}_j$. For any data point j, the classification is deemed correct if the right label is predicted. To evaluate the loss incurred when using the prediction function $\phi(\mathbf{u}; \theta)$, which in this section is supposed to be a DNN, we use a loss function $\ell(\phi(\mathbf{u}; \theta), \mathbf{v})$.

The goal of CL is to minimize the prediction error over a sequence of tasks that become available one at a time. Among the many different formulations proposed for CL, hierarchical objectives have been used in [47, 56]. In this section, we present an incremental setting where each task is available as a subset of samples, similar to the formulation in [47]. Given $t \in \{1, \ldots, T\}$, let $\mathcal{D}_t$ be the set of samples for a new task T_t, which can be split into a training set $D^t_{\rm tr}$ and a validation set $D^t_{\rm v}$. Moreover, let us split the parameters θ_t of the model $\phi(\mathbf{u}; \theta_t)$ into two subvectors, λ_t and δ_t, whose roles are to give us flexibility in minimizing the classification error on the training and validation data along the sequence of tasks. According to [21], when using a neural network as a prediction function, a reasonable strategy is to choose λ_t and δ_t as the vectors of weights in the hidden and output layers, respectively.

To solve the overall CL problem, one starts from the first task T_1 and, after an arbitrary number of iterations or amount of time, we include in the problem the second task T_2. One reiterates this procedure until all the tasks have been added to the problem. Let us now suppose that one has already added t tasks. At this stage, the goal of the UL and LL problems is to determine the values of λ_t and δ_t that ensure a small classification error on T_t and on all the previous tasks T_i, with $i < t$. To this end, the UL problem determines (λ_t, δ_t) by minimizing the prediction error on $D^t_{\rm val} = \cup_{i \le t} D^i_{\rm v}$, which is composed of the data sampled from the validation sets associated with the current and previous tasks. Similarly, the LL problem determines δ_t by minimizing the error on $D^t_{\rm train} = \cup_{i \le t} D^i_{\rm tr}$. Note that at each stage one solves a different problem since the objective functions of the UL and LL change as new tasks are included in the problem. The formulation of the problem solved at stage t, with $t \in \{1, \ldots, T\}$, can be written as follows:

$$\begin{array}{rl} \min\limits_{(\lambda_t,\, \delta_t)} & f_u(\lambda_t,\, \delta_t) \;=\; \dfrac{1}{|D^t_{\rm val}|} \displaystyle\sum_{(\mathbf{u},\mathbf{v}) \in\, D^t_{\rm val}} \ell(\phi(\mathbf{u}; \lambda_t, \delta_t), \mathbf{v}) \\ \text{s.t.} & \delta_t \in \operatorname*{argmin}\limits_{\delta_t} \; f_\ell(\lambda_t,\, \delta_t) \;=\; \dfrac{1}{|D^t_{\rm train}|} \displaystyle\sum_{(\mathbf{u},\mathbf{v}) \in\, D^t_{\rm train}} \ell(\phi(\mathbf{u}; \lambda_t, \delta_t), \mathbf{v}). \end{array} \tag{5.14}$$

We point out that the large dimension of the datasets usually considered in ML may prevent the use of the whole sets $D^i_{\rm tr}$ and $D^i_{\rm v}$ from previous tasks i's, where $i < t$ and t is the current task. In such cases, it may be necessary to resort to subsets $\bar{D}^i_{\rm tr} \subset D^i_{\rm tr}$ and $\bar{D}^i_{\rm v} \subset D^i_{\rm v}$, which we will not do in this paper given that our interest focuses on the solution of stochastic BLPs.

Once a new task is included in the problem, the classification accuracy of the DNN on the previous tasks tends to deteriorate, thus resulting in the well-studied phenomenon of *catastrophic forgetting* [26], which can be alleviated by adding LL inequality constraints to the lower level

of problem (5.14). Such inequality constraints are inspired by [37] and ensure that, at each stage, the current model outperforms the old model on all the previous tasks, thus preventing the deterioration of the classification accuracy when learning new tasks. In particular, for all $i < t$ and $t \geq 2$, we have

$$\sum_{(\mathbf{u},\mathbf{v})\in D^i_{\text{tr}}} \ell(\phi(\mathbf{u};\lambda_t,\delta_t),\mathbf{v}) - \sum_{(\mathbf{u},\mathbf{v})\in D^i_{\text{tr}}} \ell(\phi(\mathbf{u};\lambda_{t-1},\delta_{t-1}),\mathbf{v}) \;\leq\; 0. \tag{5.15}$$

We point out that similar constraints were also used in the bilevel formulation proposed in [56], where the violation of the constraints is penalized. Note that, in general, the constraints in (5.15) may be nonconvex when using a neural network as the prediction function $\phi(\mathbf{u};\lambda_t,\delta_t)$. However, our theory does encompass nonconvexity of the LL constraints as long as the LL SOSC is satisfied (see Assumption 3.5).

5.5 Results for continual learning instances

We now present numerical results comparing BSG-N-FD and BSG-1 against both DARTS and StocBiO on the CL problem (5.14) that was posed in Section 5.4. We also include numerical results on the CL problem with LL constraints defined by (5.15) for BSG-N-FD and BSG-1, as no other method applies in this case. In our implementation, we determine λ_t (the UL variables) on the current problem by starting from the parameter values found from the previous problem. However, since each consecutive task increases the output space of the DNN, we entirely re-initialize δ_t (the LL variables) at the start of each new task (when first applying an LL step) so that the model outputs are not biased from previous tasks.

In order to test our algorithm on a large-scale ML scenario, we chose the well-studied CIFAR-10 dataset [32], which consists of a total of 60,000 colored images (32×32) of 10 different classes (i.e., airplanes, automobiles, birds, cats, deer, dogs, frogs, horses, ships, and trucks). The dataset is split into a training set that consists of 50,000 images and a testing set that consists of 10,000 images; however, for our experiments we only used the first set of images. We used a subset of 40,000 images for training and the remaining 9,999 images for validation (since one of the images had an issue, we removed it from the dataset). We solved five problems (5.14) with an increasing number of tasks from 1 to 5, where the first task datasets (D^1_{val} and D^1_{train}) consist of only the images with class labels in $\{0,1\}$ (these correspond to airplanes and automobiles), the second task datasets (D^2_{val} and D^2_{train}) consist of the images with class labels in $\{0,1,2,3\}$ (airplanes, automobiles, birds, and cats), etc., until the final task datasets (D^5_{val} and D^5_{train}), which are the original training and validation sets and consist of all the class labels $\{0,1,...,9\}$. Further, we implemented a DNN with two convolutional layers, a max-pooling layer, and one linear fully-connected layer as our model. The network consisted of 19,392 and 163,840 weights in the hidden and output layers, respectively. For the UL and LL problems, we have used batch sizes equal to 0.05% and 0.01% of the sizes of the current task's validation and training datasets, respectively.

All of the algorithms, with the exception of DARTS in the LL unconstrained case, were run while using an increasing accuracy strategy in the LL problem with an f_u difference threshold for increasing the number of LL iterations equal to 10^{-2} (and a maximum limit of 30 LL iterations). The number of UL iterations and running time (in seconds) were both used as metrics for the comparison. As a loss function, we used the well-known binary cross-entropy loss. In all of the

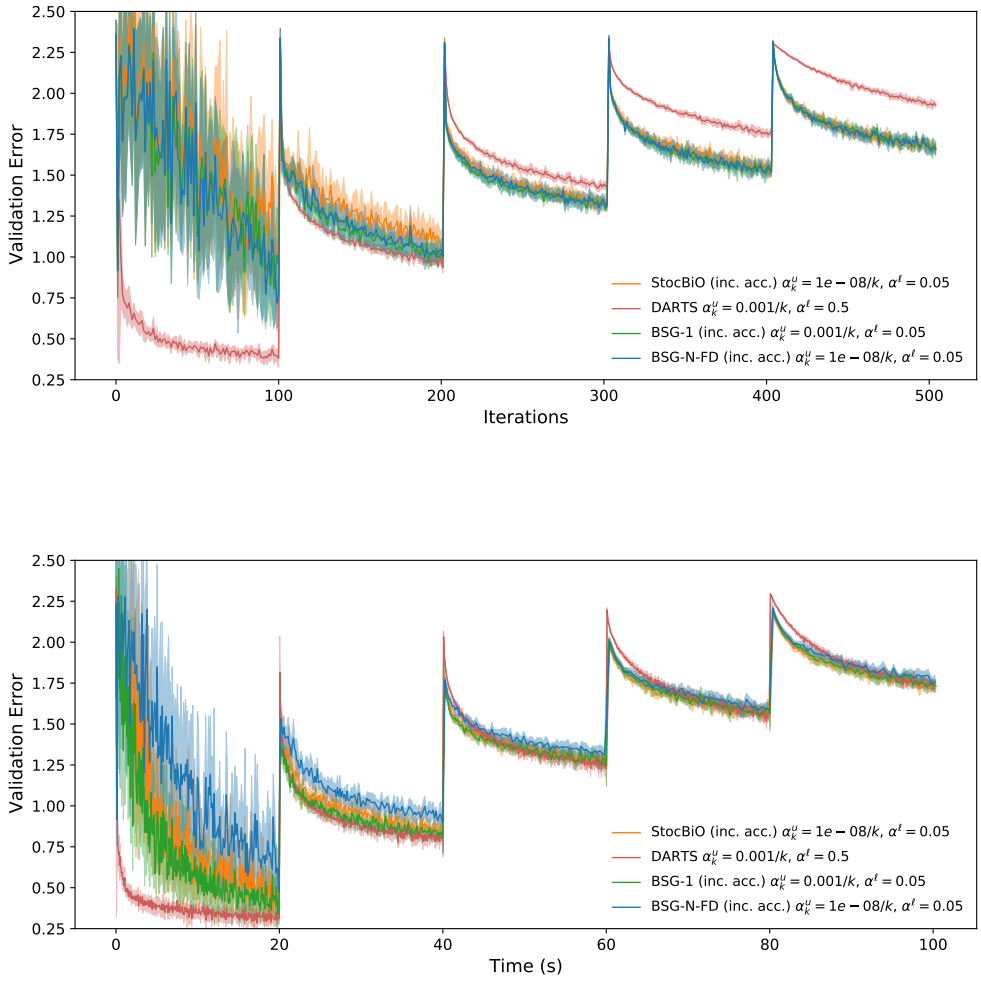

Figure 4: Comparison of BSG-N-FD, BSG-1, DARTS, and StocBiO on the CL problem (5.14) in terms of both iterations (top plot) and time (bottom plot, in seconds).

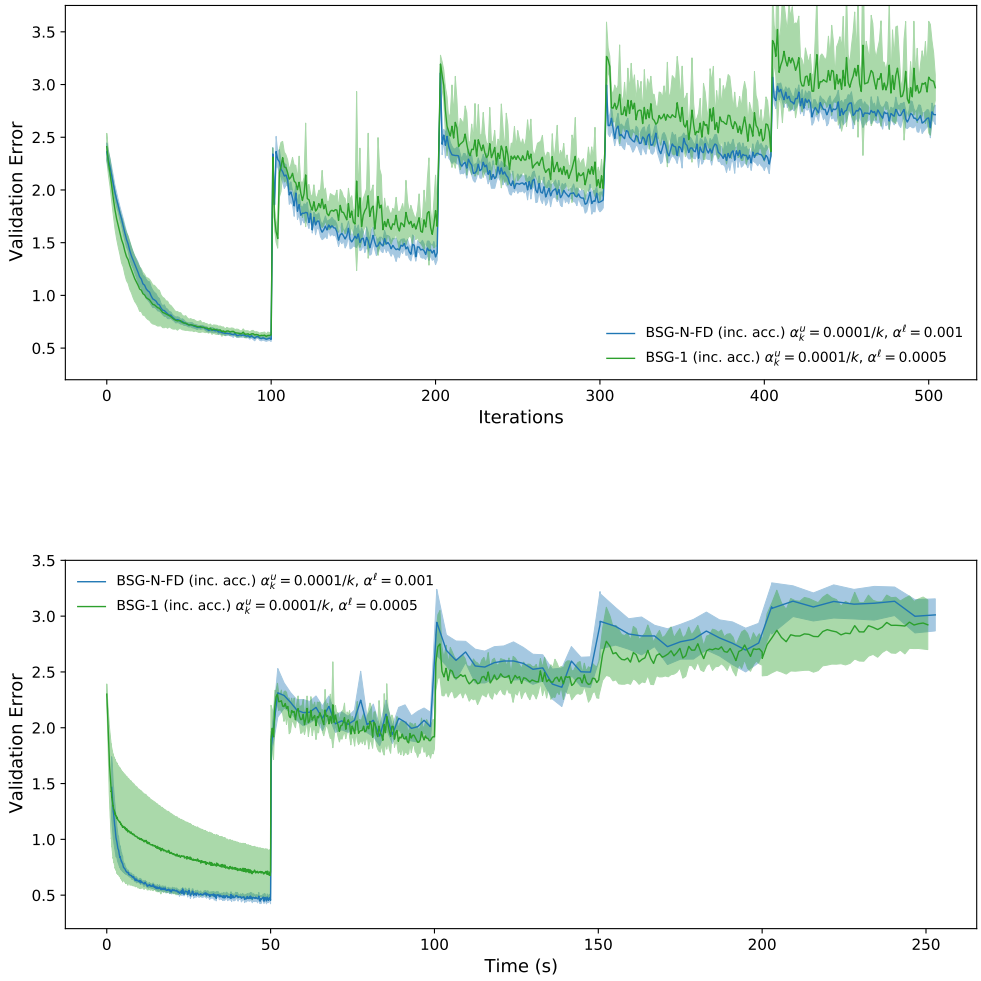

Figure 5: Comparison of BSG-N-FD and BSG-1 on the CL problem (5.14) with constraints (5.15) in terms of both iterations (top plot) and time (bottom plot, in seconds).

figures in this section, we plot the approximation f_u of the true function f, as it is typically done in bilevel ML [27, 38, 46, 59, 66].

The results for the unconstrained LL case are reported in Figure 4. We are not reporting BSG-H because of the extremely high computational cost of dealing with second-order derivatives given the choices of DNN and dataset. We compare four algorithms, BSG-N-FD and BSG-1 against DARTS and StocBiO, using the best UL decaying stepsize sequence $\{\alpha_k^u\}_{k\in\mathbb{N}}$ and the best LL fixed stepsize α^ℓ found for each algorithm. The step sizes for each algorithm were obtained by performing grid searches over the following sets: $\alpha^u \in \{5\cdot 10^{-3}/k, 10^{-3}/k, 5\cdot 10^{-4}/k\}$ for BSG-1, and DARTS and $\alpha^u \in \{10^{-7}/k, 10^{-8}/k\}$ for BSG-N-FD and StocBiO; $\alpha^\ell \in \{10^{-1}, 5\cdot 10^{-2}, 10^{-2}\}$ for BSG-N-FD, BSG-1, and StocBiO, $\alpha^\ell \in \{1, 5\cdot 10^{-1}, 10^{-1}\}$ for DARTS. Again, for StocBiO, we set the constant C_0 introduced in Subsection 1.2 to 0.05 and the parameter q introduced in Subsection 2.5 to 2, which led to the best results. By the nature of the CL problem, we expect to see five separate "jumps" in the validation error (the UL objective function) indicating the start of a new task. Among the four algorithms, BSG-N-FD, BSG-1, and StocBiO have similar performance in terms of iterations. In particular, they perform the best on all tasks excluding the first two, with StocBiO performing the worst on task two. On the first two tasks, DARTS performs the best in terms of both iterations and time, while BSG-N-FD, BSG-1, and StocBiO experience some initial noise. All methods seem to perform similarly on the last three tasks, except for DARTS, which seems to fall behind on the last two tasks.

Lastly, in Figure 5, we provide numerical results for BSG-N-FD and BSG-1 on the CL problem (5.14) when considering LL constraints (5.15). In accordance with the procedure described in Subsection 5.1 for the LL constrained case, approximate Lagrange multipliers are obtained at each iteration by solving the corresponding KKT system with the linear conjugate gradient method (with maximum number of iterations equal to 3 and tolerance equal to 10^{-4}). The system in (5.9) is solved by using the GMRES method with maximum number of iterations equal to 3 when running BSG-N-FD and 50 when running BSG-1, with a tolerance equal to 10^{-4}. In a similar manner for the CL unconstrained case, we chose the stepsizes for each algorithm by performing the following grid searches: $\alpha^u \in \{5\cdot 10^{-4}/k, 10^{-4}/k, 5\cdot 10^{-5}/k\}$ for both algorithms, $\alpha^\ell \in \{5\cdot 10^{-3}, 10^{-3}, 5\cdot 10^{-4}\}$ for BSG-N-FD, and $\alpha^\ell \in \{10^{-3}, 5\cdot 10^{-4}, 10^{-4}\}$ for BSG-1. Referring to Figure 5, BSG-N-FD performs the best in terms of iterations on all tasks except for the first, on which both algorithms perform similarly. In terms of time, BSG-1 yields slightly superior performance compared to BSG-N-FD besides the first task. It should be noted that both algorithms seem to plateau after the second task in terms of time as the number of iterations in each consecutive task decreases substantially. This is due to the amount of time allotted to each task. The results in Figure 5 demonstrate that our BSG methods are able to solve large-scale bilevel optimization problems with nonlinear constraints in the LL problem, and similar to Section 5.3.3, these results are the first of their kind for this type of problem, to the best of our knowledge.

6 Concluding remarks and future work

In this paper, we proposed a general framework for bilevel stochastic gradient (BSG) methods that applies to both the LL unconstrained and constrained cases, we provided a corresponding convergence theory that allows for any inexactness in the calculation of adjoint gradients and that also rigorously covers the inexact solution of the LL problem, and we introduced practi-

cal BSG methods for large-scale bilevel optimization problems (BSG-N-FD and BSG-1). The numerical results showed that BSG-N-FD, which is consistent with the theory, performs well on the synthetic quadratic bilevel problem. On the continual learning instances, BSG-N-FD has a similar performance to the practical algorithms BSG-1 and StocBiO in terms of iterations and is slightly outperformed in terms of time.

The results on the ML instances considered in this proposal suggest that our BSG methods have the potential to perform well on the unconstrained bilevel formulations of NAS, which in the literature are mostly still tackled by using DARTS when a continuous relaxation of the (discrete) search space is used [49]. We point out that using finite differences like in BSG-N-FD or rank-1 Hessian approximations like in BSG-1 is crucial to allow the application of the BSG method to NAS, which would not be possible otherwise due to the extreme dimensions of the resulting bilevel problems. Moreover, the fact that our BSG methods can solve bilevel optimization problems with constrained LL problems paves the way for the solution of new NAS formulations. In particular, one could think of including in the LL problem constraints that help the model avoid overfitting [48] or constraints that depend on the specific learning instances considered [54]. Also left for future work are variance reduction techniques, which can be incorporated into our BSG methods to ensure faster convergence, as already proposed in [8, 66].

Acknowledgments

This work is partially supported by the U.S. Air Force Office of Scientific Research (AFOSR) award FA9550-23-1-0217.

References

[1] J. F. Bard. *Practical Bilevel Optimization: Algorithms and Applications*. Springer Publishing Company, Incorporated, 1st edition, 2010.

[2] A. Beck. *First-Order Methods in Optimization*. SIAM-Society for Industrial and Applied Mathematics, 2017.

[3] K. P. Bennett, G. Kunapuli, J. Hu, and J. S. Pang. *Bilevel Optimization and Machine Learning*, pages 25–47. Springer Berlin Heidelberg, Berlin, Heidelberg, 2008.

[4] A. Botev, H. Ritter, and D. Barber. Practical Gauss-Newton optimisation for deep learning. In Doina Precup and Yee Whye Teh, editors, *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 557–565. PMLR, 06–11 Aug 2017.

[5] L. Bottou, F. E. Curtis, and J. Nocedal. Optimization methods for large-scale machine learning. 60:223–311, 2018.

[6] C. Chen, S. Zheng, X. Chen, E. Dong, X. Liu, H. Liu, and D. Dou. Generalized dataweighting via class-level gradient manipulation. In *Advances in Neural Information Processing Systems*, 2021.

[7] C. Chen, X. Chen, C. Ma, Z. Liu, and X. Liu. Gradient-based bi-level optimization for deep learning: A survey. *arXiv preprint arXiv:2207.11719*, 2022.

[8] T. Chen, Y. Sun, and W. Yin. A Single-Timescale Stochastic Bilevel Optimization Method. *arXiv e-prints*, art. arXiv:2102.04671, February 2021.

[9] T. Chen, Y. Sun, and W. Yin. Closing the gap: Tighter analysis of alternating stochastic gradient methods for bilevel problems. In M. Ranzato, A. Beygelzimer, Y. Dauphin, P.S. Liang, and J. Wortman Vaughan, editors, *Advances in Neural Information Processing Systems*, volume 34, pages 25294–25307. Curran Associates, Inc., 2021.

[10] K. L. Chung. On a stochastic approximation method. *Annals of Mathematical Statistics*, 25:463 – 483, 1954.

[11] B. Colson, P. Marcotte, and G. Savard. An overview of bilevel optimization. *Annals of Operations Research*, 153:235–256, 2007.

[12] N. Couellan and W. Wang. Bi-level stochastic gradient for large scale support vector machine. *Neurocomputing*, 153:300–308, 2015.

[13] N. Couellan and W. Wang. On the convergence of stochastic bi-level gradient methods. `http://www.optimization-online.org/DB\_HTML/2016/02/5323.html`, 2016.

[14] S. Dempe. *Foundations of bilevel programming*, volume 61 of *Nonconvex Optimization and its Applications*. Kluwer Academic Publishers, Dordrecht, 2002.

[15] S. Dempe and A. Zemkoho. *Bilevel Optimization: Advances and Next Challenges*. Springer International Publishing, 2020.

[16] S. Dhar, U. Kurup, and M. Shah. Stabilizing bi-Level hyperparameter optimization using Moreau-Yosida regularization. *arXiv e-prints*, art. arXiv:2007.13322, July 2020.

[17] Weinan E. A proposal on machine learning via dynamical systems. *Communications in Mathematics and Statistics*, 5:1–11, 02 2017.

[18] A. V. Fiacco. Sensitivity analysis for nonlinear programming using penalty methods. *Math. Programming*, 10:287–311, 1976.

[19] A. V. Fiacco. *Introduction to sensitivity and stability analysis in nonlinear programming*, volume 165 of *Mathematics in Science and Engineering*. Academic Press, Inc., Orlando, FL, 1983.

[20] A. V. Fiacco and G. P. McCormick. *Nonlinear programming: Sequential unconstrained minimization techniques*. John Wiley & Sons, Inc., New York-London-Sydney, 1968.

[21] L. Franceschi, P. Frasconi, S. Salzo, R. Grazzi, and M. Pontil. Bilevel programming for hyperparameter optimization and meta-learning. In Jennifer Dy and Andreas Krause, editors, *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 1568–1577. PMLR, 2018.

[22] R. M. French. Catastrophic forgetting in connectionist networks. *Trends in Cognitive Sciences*, 3:128–135, 1999.

[23] M. Gargiani, A. Zanelli, M. Diehl, and F. Hutter. On the promise of the stochastic generalized Gauss-Newton method for training DNNs. *arXiv e-prints*, June 2020.

[24] S. Ghadimi and M. Wang. Approximation Methods for Bilevel Programming. *arXiv e-prints*, art. arXiv:1802.02246, February 2018.

[25] I. Goodfellow, J. Shlens, and C. Szegedy. Explaining and harnessing adversarial examples. In *International Conference on Learning Representations*, 2015.

[26] I. J. Goodfellow, M. Mirza, D. Xiao, A. Courville, and Y. Bengio. An empirical investigation of catastrophic forgetting in gradient-based neural networks. *arXiv e-prints*, art. arXiv:1312.6211, December 2013.

[27] M. Hong, H. Wai, Z. Wang, and Z. Yang. A Two-Timescale Framework for Bilevel Optimization: Complexity Analysis and Application to Actor-Critic. *arXiv e-prints*, art. arXiv:2007.05170, July 2020.

[28] T. Hospedales, A. Antoniou, P. Micaelli, and A. Storkey. Meta-Learning in neural networks: A survey. *arXiv e-prints*, art. arXiv:2004.05439, April 2020.

[29] K. Ji, J. Yang, and Y. Liang. Bilevel optimization: Convergence analysis and enhanced design. In Marina Meila and Tong Zhang, editors, *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pages 4882–4892. PMLR, 18–24 Jul 2021.

[30] H. Jiang, Z. Chen, Y. Shi, B. Dai, and T. Zhao. Learning to defend by learning to attack. *arXiv e-prints*, art. arXiv:1811.01213, November 2018.

[31] P. Khanduri, I. Tsaknakis, Y. Zhang, J. Liu, S. Liu, J. Zhang, and M. Hong. Linearly constrained bilevel optimization: A smoothed implicit gradient approach. In *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pages 16291–16325. PMLR, 23–29 Jul 2023.

[32] A. Krizhevsky. Learning multiple layers of features from tiny images. pages 32–33, 2009. URL `https://www.cs.toronto.edu/~kriz/learning-features-2009-TR.pdf`.

[33] J. Kwon, D. Kwon, S. Wright, and R. Nowak. On Penalty Methods for Nonconvex Bilevel Optimization and First-Order Stochastic Approximation. *arXiv e-prints*, art. arXiv:2309.01753, September 2023.

[34] H. Liu, K. Simonyan, and Y. Yang. DARTS: Differentiable architecture search. *ArXiv*, arXiv:1806.09055, 2019.

[35] R. Liu, J. Gao, J. Zhang, D. Meng, and Z. Lin. Investigating bi-Level optimization for learning and vision from a unified perspective: A survey and beyond. *arXiv e-prints*, art. arXiv:2101.11517, January 2021.

[36] S. Liu and L. N. Vicente. The stochastic multi-gradient algorithm for multi-objective optimization and its application to supervised machine learning. *Annals of Operations Research*, pages 1–30.

[37] D. Lopez-Paz and M. Ranzato. Gradient episodic memory for continual learning. *arXiv e-prints*, art. arXiv:1706.08840, June 2017.

[38] J. Lorraine, P. Vicol, and D. Duvenaud. Optimizing Millions of Hyperparameters by Implicit Differentiation. *arXiv e-prints*, art. arXiv:1911.02590, November 2019.

[39] Y. Lu, A. Zhong, Q. Li, and B. Dong. Beyond Finite Layer Neural Networks: Bridging Deep Architectures and Numerical Differential Equations. *arXiv e-prints*, art. arXiv:1710.10121, October 2017.

[40] A. Madry, A. Makelov, L. Schmidt, D. Tsipras, and A. Vladu. Towards deep learning models resistant to adversarial attacks. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*. OpenReview.net, 2018.

[41] M. McCloskey and N. J. Cohen. Catastrophic interference in connectionist networks: The sequential learning problem. volume 24 of *Psychology of Learning and Motivation*, pages 109–165. Academic Press, 1989.

[42] G. P. McCormick. Optimality criteria in nonlinear programming. pages 27–38, Philadelphia, PA, USA, 1976. SIAM.

[43] A. Nemirovski, A. Juditsky, G. Lan, and A. Shapiro. Robust stochastic approximation approach to stochastic programming. *SIAM Journal on Optimization*, 19:1574–1609, 2009.

[44] Y. Nesterov. *Lectures on Convex Optimization*. Springer Publishing Company, Incorporated, 2nd edition, 2018. ISBN 3319915770.

[45] J. Nocedal and S. J. Wright. *Numerical Optimization*. Springer-Verlag, Berlin, second edition, 2006.

[46] F. Pedregosa. Hyperparameter optimization with approximate gradient. In Maria Florina Balcan and Kilian Q. Weinberger, editors, *Proceedings of The 33rd International Conference on Machine Learning*, volume 48 of *Proceedings of Machine Learning Research*, pages 737–746, New York, New York, USA, 20–22 Jun 2016. PMLR.

[47] Q. Pham, D. Sahoo, C. Liu, and S. C. H. Hoi. Bilevel continual learning, 2020.

[48] S. N. Ravi, T. Dinh, V. S. Lokhande, and V. Singh. Explicitly imposing constraints in deep networks via conditional gradients gives improved generalization and faster convergence. *Proceedings of the AAAI Conference on Artificial Intelligence*, 33(01):4772–4779, 2019.

[49] P. Ren, Y. Xiao, X. Chang, P. Huang, Z. Li, X. Chen, and X. Wang. A comprehensive survey of neural architecture search: Challenges and solutions. *ACM Comput. Surv.*, 54, 2021.

[50] H. Robbins and S. Monro. A stochastic approximation method. *Annals of Mathematical Statistics*, 22:400–407, 1951.

[51] W. Rudin. *Principles of mathematical analysis*. McGraw-Hill Book Company, Inc., New York-Toronto-London, 1953.

[52] Y. Saad and M. H. Schultz. GMRES: A generalized minimal residual algorithm for solving nonsymmetric linear systems. *SIAM Journal on Scientific and Statistical Computing*, 7: 856–869, 1986.

[53] J. Sacks. Asymptotic distribution of stochastic approximation procedures. *Annals of Mathematical Statistics*, 29:373 – 405, 1958.

[54] S. Sangalli, E. Erdil, A. Hoetker, O. Donati, and E. Konukoglu. Constrained optimization to train neural networks on critical and under-represented Classes. *arXiv e-prints*, art. arXiv:2102.12894, February 2021.

[55] G. Savard and J. Gauvin. The steepest descent direction for the nonlinear bilevel programming problem. *Operations Research Letters*, 15:265–272, 1994.

[56] A. Shaker, F. Alesiani, S. Yu, and W. Yin. Bilevel continual learning, 2020.

[57] H. Shen, Q. Xiao, and T. Chen. On Penalty-based Bilevel Gradient Descent Method. *arXiv e-prints*, art. arXiv:2302.05185, February 2023.

[58] A. Sinha, P. Malo, and K. Deb. A review on bilevel optimization: From classical to evolutionary approaches and applications. *IEEE Transactions on Evolutionary Computation*, 22:276–295, 2018.

[59] D. Sow, K. Ji, and Y. Liang. On the Convergence Theory for Hessian-Free Bilevel Algorithms. *arXiv e-prints*, 2021.

[60] H. Sun, W. Pu, X. Fu, T.H. Chang, and M. Hong. Learning to continuously optimize wireless resource in a dynamic environment: A bilevel optimization perspective. *IRE Transactions on Audio*, 70:1900–1917, 2022. ISSN 1053-587X.

[61] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. Goodfellow, and R. Fergus. Intriguing properties of neural networks. *arXiv e-prints*, art. arXiv:1312.6199, December 2013.

[62] I. Tsaknakis, P. Khanduri, and M. Hong. An implicit gradient-type method for linearly constrained bilevel problems. ICASSP, IEEE International Conference on Acoustics, Speech and Signal Processing - Proceedings, pages 5438–5442. Institute of Electrical and Electronics Engineers Inc., 2022.

[63] L. N. Vicente and P. H. Calamai. Bilevel and multilevel programming: A bibliography review. *Journal of Global Optimization*, 5:291–306, 1994.

[64] Q. Xiao, H. Shen, W. Yin, and T. Chen. Alternating projected sgd for equality-constrained bilevel optimization. In *Proceedings of The 26th International Conference on Artificial Intelligence and Statistics*, volume 206 of *Proceedings of Machine Learning Research*, pages 987–1023. PMLR, 25–27 Apr 2023.

[65] Y. Xu. Primal-dual stochastic gradient method for convex programs with many functional constraints. *SIAM Journal on Optimization*, 30(2):1664–1692, 2020.

[66] J. Yang, K. Ji, and Y. Liang. Provably Faster Algorithms for Bilevel Optimization. *arXiv e-prints*, art. arXiv:2106.04692, June 2021.

A Proposition 3.1

Proof. Let us first prove (3.2). There are two cases to consider: inexact adjoint system and truncated Neumann series. In both we will use the fact that when B_1 and B_2 are non-singular,

$$\|B_1^{-1} - B_2^{-1}\| \leq \|B_1^{-1}(B_1 - B_2)B_2^{-1}\| \leq \|B_1^{-1}\|\|B_2^{-1}\|\|B_1 - B_2\|. \tag{A.1}$$

1) Inexact adjoint system.
The approximate BSG direction is $d(D) = -(a - AB^{-1}\tilde{b})$, where $\tilde{b} = b + \tilde{r}$ and $\tilde{r}$ is the residual error due to the inexact solution of the ajoint equation (see Subsection 2.5). Now, we have

$$\|d(D_1) - d(D_2)\| = \| - a_1 + A_1B_1^{-1}\tilde{b}_1 + a_2 - A_2B_2^{-1}\tilde{b}_2\|.$$

Adding and subtracting $A_1B_1^{-1}\tilde{b}_2$ and using the triangle inequality, we obtain

$$\|d(D_1) - d(D_2)\| \leq \|a_1 - a_2\| + \|A_1B_1^{-1}\|\|\tilde{b}_1 - \tilde{b}_2\| + \|\tilde{b}_2\|\|A_1B_1^{-1} - A_2B_2^{-1}\|.$$

Adding and subtracting $A_2B_1^{-1}$ in the last norm on the right, we have

$$\begin{aligned}\|d(D_1) - d(D_2)\| \leq\ & \|a_1 - a_2\| + \|A_1\|\|B_1^{-1}\|\|\tilde{b}_1 - \tilde{b}_2\| \\ & + \|\tilde{b}_2\|\|A_2\|\|B_1^{-1} - B_2^{-1}\| + \|\tilde{b}_2\|\|B_1^{-1}\|\|A_1 - A_2\|.\end{aligned} \tag{A.2}$$

Let us define the positive constants $C_1 = \max\{CC_\ell,\ \bar{C}\bar{C}_\ell\}$, $C_2 = \max\{C^2C_\ell^2, \bar{C}^2\bar{C}_\ell^2\}$, $C_3 = \max\{CC_\ell^2, \bar{C}\bar{C}_\ell^2\}$, and $C_4 = \max\{C_\ell, \bar{C}_\ell\}$, where C, C_ℓ, $\bar{C}$, and $\bar{C}_\ell$ are the constants introduced in Remark 3.1. From the assumptions of Proposition 3.1 and (A.1), and setting $r_1 = \tilde{r}_1$ and $r_2 = \tilde{r}_2$, there exists $L = \max\{1, C_1, C_2 + C_3\|r_2\|, C_1 + C_4\|r_2\|\}$, such that

$$\begin{aligned}\|d(D_1) - d(D_2)\| &\leq \|a_1 - a_2\| + C_1(\|b_1 - b_2\| + \|r_1 - r_2\|) \\ &\quad + (C_2 + C_3\|r_2\|)\,\|B_1 - B_2\| + (C_1 + C_4\|r_2\|)\,\|A_1 - A_2\| \\ &\leq L(\|a_1 - a_2\| + \|b_1 - b_2\| + \|B_1 - B_2\| + \|A_1 - A_2\|) + L\|r_1 - r_2\|.\end{aligned} \tag{A.3}$$

From the equivalence of norms, there exists a positive constant $\tilde{C}$ such that the proof of Part 1) is completed with $L_{BSG} = L\max\{\tilde{C}, 1\}$.

2) Truncated Neumann series.
The approximate BSG direction is $d(D) = -(a - A\mathscr{B}b)$, where $\mathscr{B} = B^{-1} - \tilde{R}$ and $\tilde{R}$ is a residual matrix (see Subsection 2.5). By repeating the reasoning used to prove Part 1) until (A.2), we arrive at

$$\begin{aligned}\|d(D_1) - d(D_2)\| \leq\ & \|a_1 - a_2\| + \|A_1\|\,\|\mathscr{B}_1\|\,\|b_1 - b_2\| \\ & +\ \|b_2\|\,\|A_2\|\,\|\mathscr{B}_1 - \mathscr{B}_2\| + \|b_2\|\,\|\mathscr{B}_1\|\,\|A_1 - A_2\|\,.\end{aligned}$$

Let us define the positive constants $C_1 = \max\{CC_\ell, \bar{C}\bar{C}_\ell\}$, $C_2 = \max\{C, \bar{C}\}$, $C_3 = \max\{C^2C_\ell^2,$ $\bar{C}^2\bar{C}_\ell^2\}$, $C_4 = \max\{C^2, \bar{C}^2\}$, and $C_5 = \max\{C, \bar{C}\}$, where C, C_ℓ, $\bar{C}$, and $\bar{C}_\ell$ are again the constants introduced in Remark 3.1. Therefore, by using the same arguments as in Part 1), but now with $r_1 = \tilde{R}_1$ and $r_2 = \tilde{R}_2$, there exists $L = \max\{1, C_1 + C_2\|r_1\|, C_3, C_4, C_1 + C_5\|r_1\|\}$, such that

$$\begin{aligned}\|d(D_1) - d(D_2)\| &\leq \|a_1 - a_2\| + (C_1 + C_2\|r_1\|)\,\|b_1 - b_2\| \\ &\quad + C_3\|B_1 - B_2\| + C_4\|r_1 - r_2\| + (C_1 + C_5\|r_1\|)\,\|A_1 - A_2\| \\ &\leq L(\|a_1 - a_2\| + \|b_1 - b_2\| + \|B_1 - B_2\| + \|A_1 - A_2\|) + L\|r_1 - r_2\|.\end{aligned} \tag{A.4}$$

Again, from the equivalence of norms, there exists a positive constant $\tilde{C}$ such that the proof of Part 2) is completed with $L_{BSG} = L \max\{\tilde{C}, 1\}$.

We will now prove (3.3) for both the LL unconstrained and constrained cases. From the equivalence of norms, there exists a positive constant $\hat{C}$ such that

$$\|D_1 - D_2\| \;\leq\; \hat{C}(\|a_1 - a_2\| + \|b_1 - b_2\| + \|B_1 - B_2\| + \|A_1 - A_2\|),$$

where $D_1 = D(x, w(x), \xi)$, $D_2 = D(x, \tilde{w}, \xi)$, and (a_i, b_i, A_i, B_i) is the stochastic data in D_i, for $i \in \{1, 2\}$. From the assumption on the Lipschitz continuity of the stochastic gradients, Hessians, and Jacobians in D_1 and D_2 for all ξ, there exists a positive constant $\bar{L}_{LL}$ such that (3.3) is satisfied. □

B Proposition 3.2

Proof. We start by handling the LL unconstrained case. Taking the norm of equations (1.4) and (2.7) and from Remark 3.1, there exists a positive constant $L_\ell = CC_\ell$ such that $\|\nabla y(x)\| \leq L_\ell$ and $\|\nabla w(x)\| \leq L_\ell$. It is well known that these two inequalities imply that $y(x)$ and $w(x)$ are Lipschitz continuous in x with constant L_ℓ (see, e.g., [2, Chapter 5]). Therefore, one can write

$$\|y(x_1) - y(x_2)\| \;\leq\; L_\ell \|x_1 - x_2\| \quad \text{and} \quad \|w(x_1) - w(x_2)\| \;\leq\; L_\ell \|x_1 - x_2\|. \tag{B.1}$$

To prove (3.19) for the LL unconstrained case, using the derivation followed for the proof of Proposition 3.1 until (A.3)–(A.4) and considering $r_1 = r_2 = 0$, we have

$$\|\nabla f(x_1) - \nabla f(x_2)\| \;\leq\; L\,(\|a_1 - a_2\| + \|b_1 - b_2\| + \|B_1 - B_2\| + \|A_1 - A_2\|), \tag{B.2}$$

with $a_i = \nabla_x f_u(x_i, y(x_i))$, $b_i = \nabla_y f_u(x_i, y(x_i))$, $B_i = \nabla^2_{yy} f_\ell(x_i, y(x_i))$, and $A_i = \nabla^2_{xy} f_\ell(x_i, y(x_i))$, $i \in \{1, 2\}$.

By the Lipschitz continuity of $\nabla_x f_u$ in (x, y) due to Assumption 3.1, we have

$$\|a_1 - a_2\| \;=\; \|\nabla_x f_u(x_1, y(x_1)) - \nabla_x f_u(x_2, y(x_2))\| \;\leq\; L_1 \|(x_1 - x_2, y(x_1) - y(x_2))^\top\|,$$

where L_1 denotes the Lipschitz constant. Squaring both sides and using (B.1), we obtain

$$\|a_1 - a_2\|^2 \;\leq\; L_1^2(\|x_1 - x_2\|^2 + L_\ell^2 \|x_1 - x_2\|^2) \;=\; L_1^2(1 + L_\ell^2)\|x_1 - x_2\|^2.$$

Taking the square root of both sides yields $\|a_1 - a_2\| \leq L_a \|x_1 - x_2\|$, where $L_a = L_1(1 + L_\ell^2)^{\frac{1}{2}}$. Since $\nabla_y f_u$ is Lipschitz continuous in (x, y), after performing the same process as above, we will obtain the following bound: $\|\nabla_y f_u(x_1, y(x_1)) - \nabla_y f_u(x_2, y(x_2))\| \;\leq\; L_b \|x_1 - x_2\|$. Similarly, since $\nabla^2_{yy} f_\ell$ and $\nabla^2_{xy} f_\ell$ are Lipschitz continuous in (x, w), we have $\|\nabla^2_{yy} f_\ell(x_1, y(x_1)) - \nabla^2_{yy} f_\ell(x_2, y(x_2))\| \;\leq\; L_B \|x_1 - x_2\|$ and $\left\|\nabla^2_{xy} f_\ell(x_1, y(x_1)) - \nabla^2_{xy} f_\ell(x_2, y(x_2))\right\| \;\leq\; L_A \|x_1 - x_2\|$. Thus, substituting all of these into (B.2), we obtain

$$\|\nabla f(x_1) - \nabla f(x_2)\| \;\leq\; L_{\nabla f} \|x_1 - x_2\|,$$

where $L_{\nabla f} = L\,(L_a + L_b + L_A + L_B)$. This concludes the proof for the LL unconstrained case.

The proof for the LL constrained case follows very similar steps. However, given the structure of the adjoint gradient (2.8) in the constrained case, we must first establish the Lipschitz

continuity in x of $\nabla_v G$ and $\nabla_x G$ given in (2.6). At this point of the paper, this follows from already-seen arguments, which we will briefly summarize here to avoid repetition. The Lipschitz continuity of the Hessian terms $\nabla^2_{yy}\mathcal{L}_\ell$ and $\nabla^2_{yx}\mathcal{L}_\ell$ results from the Lipschitz continuity of the Hessians defining the problem, the Lipschitz continuity of the multipliers (B.1), the Lipschitz continuity of sums and products, and the boundedness of all terms by Assumption 3.4. The Lipschitz continuity of the terms $z_I \circ \nabla_x c_I^\top$ and $z_I \circ \nabla_y c_I^\top$ results from the Lipschitz continuity of the Jacobians, the Lipschitz continuity of the multipliers (B.1), the Lipschitz continuity of sums and products, and the boundedness of all terms. The remaining elements in (2.6) are constraint functions and their Jacobians, which are Lipschitz continuous per Assumption 3.4. □

C Theorem 4.4

Proof. For any $k \in \mathbb{N}$, we can write

$$\begin{aligned}
\mathbb{E}_{\xi_k^{\text{all}}}[\|x_{k+1} - x_*\|^2] &= \mathbb{E}_{\xi_k^{\text{all}}}[\|P_X(x_k + \alpha_k\, d(x_k, \tilde{w}_k, \xi_k)) - x_*\|^2] \\
&\leq \mathbb{E}_{\xi_k^{\text{all}}}[\|x_k + \alpha_k\, d(x_k, \tilde{w}_k, \xi_k) - x_*\|^2] \\
&= \|x_k - x_*\|^2 + \alpha_k^2 \mathbb{E}_{\xi_k^{\text{all}}}[\|d(x_k, \tilde{w}_k, \xi_k)\|^2] \\
&\quad + 2\alpha_k \mathbb{E}_{\xi_k^{\text{all}}}[d(x_k, \tilde{w}_k, \xi_k)]^\top (x_k - x_*).
\end{aligned}$$

Adding and subtracting the term $2\alpha_k(\mathbb{E}_{\xi_k^{\text{all}}}[d(x_k, w(x_k))])^\top (x_k - x_*)$, noting that $d(x_k, w(x_k)) = -\nabla f(x_k)$, and applying the Cauchy-Schwarz and Jensen's inequalities, we obtain

$$\begin{aligned}
\mathbb{E}_{\xi_k^{\text{all}}}[\|x_{k+1} - x_*\|^2] &\leq \|x_k - x_*\|^2 + \alpha_k^2 \mathbb{E}_{\xi_k^{\text{all}}}[\|d(x_k, \tilde{w}_k, \xi_k)\|^2] - 2\alpha_k \nabla f(x_k)^\top (x_k - x_*) \\
&\quad + 2\alpha_k \mathbb{E}_{\xi_k^{\text{all}}}[d(x_k, \tilde{w}_k, \xi_k) - d(x_k, w(x_k))]^\top (x_k - x_*) \\
&\leq \|x_k - x_*\|^2 + \alpha_k^2 \mathbb{E}_{\xi_k^{\text{all}}}[\|d(x_k, \tilde{w}_k, \xi_k)\|^2] - 2\alpha_k \nabla f(x_k)^\top (x_k - x_*) \\
&\quad + 2\alpha_k \mathbb{E}_{\xi_k^{\text{all}}}[\|d(x_k, \tilde{w}_k, \xi_k) - d(x_k, w(x_k))\|]\|x_k - x_*\|.
\end{aligned}$$

Then, by using Assumption 4.3 and inequalities (3.5), (3.9), and (4.7),

$$\mathbb{E}_{\xi_k^{\text{all}}}[\|x_{k+1} - x_*\|^2] \;\leq\; (1 - 2c\alpha_k)\|x_k - x_*\|^2 + (G_d + 2C_d\Theta)\,\alpha_k^2.$$

Denoting $M = G_d + 2C_d\Theta$ and taking the total expectation on both sides, one obtains

$$\mathbb{E}[\|x_{k+1} - x_*\|^2] \;\leq\; (1 - 2c\alpha_k)\mathbb{E}[\|x_k - x_*\|^2] + M\alpha_k^2.$$

Using $\alpha_k = \frac{\gamma}{k}$ for some constant $\gamma > \frac{1}{2c}$, it follows by induction [43, Eq. (2.9) and (2.10)] that

$$\mathbb{E}[\|x_k - x_*\|^2] \;\leq\; \frac{\max\{2\,\gamma^2 M(2c\gamma - 1)^{-1}, \|x_0 - x_*\|^2\}}{k}, \tag{C.1}$$

which proves the first result.

From (3.20), one obtains (see, e.g., [2, Lemma 5.7] and $P_X(x_k - x_*) = (x_k - x_*)$)

$$f(x_k) \;\leq\; f(x_*) + (P_X \nabla f(x_*))^\top (x_k - x_*) + \frac{1}{2} L_{\nabla f} \|x_k - x_*\|^2. \tag{C.2}$$

From (C.1) and (C.2), by taking the total expectation and recalling $P_X \nabla f(x_*) = 0$, one can obtain the optimality gap in terms of function values, yielding

$$\begin{aligned}\mathbb{E}[f(x_k)] - f(x_*) \;&\leq\; \frac{1}{2} L_{\nabla f} \mathbb{E}[\|x_k - x_*\|^2] \\ &\leq\; \frac{(L_{\nabla f}/2)\max\{2\,\gamma^2 M(2c\gamma - 1)^{-1}, \|x_0 - x_*\|^2\}}{k}.\end{aligned}$$

□

D Theorem 4.5

Proof. Assumption 4.5 implies that

$$\nabla f(x_k)^\top (x_k - x_*) \;\geq\; f(x_k) - f(x_*). \tag{D.1}$$

Repeating the same arguments that in the proof of Theorem 4.4 led to (C.1), but now using (D.1) instead, we obtain

$$\mathbb{E}_{\xi_k^{\text{all}}}[\|x_{k+1} - x_*\|^2] \;\leq\; \|x_k - x_*\|^2 + 2\alpha_k(f(x_*) - f(x_k)) + (G_d + 2C_d\Theta)\alpha_k^2.$$

Letting $M = G_d + 2C_d\Theta$, we have

$$\mathbb{E}_{\xi_k^{\text{all}}}\left[\|x_{k+1} - x_*\|^2\right] \;\leq\; \|x_k - x_*\|^2 + 2\alpha_k(f(x_*) - f(x_k)) + M\alpha_k^2.$$

Rearranging, taking total expectations, and dividing by α_k, we obtain

$$2(\mathbb{E}[f(x_k)] - f(x_*)) \;\leq\; \frac{\mathbb{E}\left[\|x_k - x_*\|^2\right]}{\alpha_k} - \frac{\mathbb{E}\left[\|x_{k+1} - x_*\|^2\right]}{\alpha_k} + M\alpha_k.$$

If we replace k by s and sum over $s = 0, 1, \ldots, k$, we obtain

$$\begin{aligned}2\sum_{s=0}^{k}(\mathbb{E}[f(x_s)] - f(x_*)) \;&\leq\; \sum_{s=0}^{k}\left(\frac{\mathbb{E}\left[\|x_s - x_*\|^2\right]}{\alpha_s} - \frac{\mathbb{E}\left[\|x_{s+1} - x_*\|^2\right]}{\alpha_s}\right) + M\sum_{s=0}^{k}\alpha_s \\ &=\; \frac{\mathbb{E}\left[\|x_0 - x_*\|^2\right]}{\alpha_0} + \sum_{s=1}^{k}\left(\frac{1}{\alpha_s} - \frac{1}{\alpha_{s-1}}\right)\mathbb{E}\left[\|x_s - x_*\|^2\right] + M\sum_{s=0}^{k}\alpha_s \\ &=\; \frac{\mathbb{E}\left[\|x_k - x_*\|^2\right]}{\alpha_k} + M\sum_{s=0}^{k}\alpha_s.\end{aligned}$$

Using Assumption 4.3 and repeating the same steps used in [36, Theorem 5.3], we can obtain the desired result. □