Applied Mathematics & Optimization (2026) 93:93
https://doi.org/10.1007/s00245-026-10451-x

Multi-iteration Stochastic Optimizers

André Carlon[1] · **Luis Espath**[2] · **Rafael Holdorf**[3] · **Raúl Tempone**[1,4,5]

Received: 14 March 2025 / Accepted: 10 May 2026 / Published online: 29 May 2026
© The Author(s) 2026

Abstract

We introduce Multi-Iteration Stochastic Optimizers, a novel class of first-order stochastic methods that control the relative L^2 error using successive control variates along the iteration path. By exploiting correlations between iterates, these control variates reduce the estimator's variance, making an accurate mean gradient estimation computationally affordable. Our approach centers on the Multi-Iteration stochastiC Estimator (MICE), which can be seamlessly coupled with any first-order stochastic optimizer due to its non-intrusive design. The algorithm adaptively selects which iterates to include in its index set. We provide both an error analysis of MICE and a convergence analysis for Multi-Iteration Stochastic Optimizers across various problem classes, including some non-convex cases. In the smooth, strongly convex setting, we demonstrate that to approximate a minimizer within a tolerance tol, SGD-MICE requires, on average, $O(tol^{-1})$ stochastic gradient evaluations, compared to $O(tol^{-1}\log(tol^{-1}))$ for SGD with adaptive batch sizes. In numerical experiments, SGD-MICE achieved the desired tolerance with fewer than 3% of the gradient evaluations required by adaptive batch SGD. Additionally, MICE offers a straightforward stopping criterion based on the gradient norm, validated through consistency tests. To assess its efficiency, we present examples using both SGD-MICE and Adam-MICE, including a stochastic adaptation of the Rosenbrock function and logistic regression on various datasets. Compared to SGD, SAG, SAGA, SVRG, and SARAH, our approach consistently reduces the gradient sampling cost without the need for extensive parameter tuning.

✉ André Carlon
carlon@uq.rwth-aachen.de

1 Department of Mathematics, RWTH Aachen University, Pontdriesch 14-16, 52062 Aachen, Germany

2 School of Mathematical Sciences, University of Nottingham, NG7 2RD, Nottingham, UK

3 School of Engineering, Federal University of Santa Catarina, Rua João Pio Duarte da Silva, Florianópolis, SC 88040-970, Brazil

4 Computer, Electrical and Mathematical Sciences & Engineering Division, King Abdullah University of Science & Technology, 23955-6900 Thuwal, Saudi Arabia

5 Alexander von Humboldt Professor in Mathematics for Uncertainty Quantification, RWTH Aachen University, Aachen, Germany

Keywords Stochastic optimization · Monte Carlo · Multilevel Monte Carlo · Variance reduction · Control variates · Machine learning

Mathematics Subject Classification 62L20 · 65K05 · 90C15 · 65C05

1 Introduction

We focus on the stochastic optimization problem of minimizing the objective function $\mathbb{E}\left[f(\boldsymbol{\xi},\boldsymbol{\theta})|\boldsymbol{\xi}\right]$, where f is a given real-valued function, $\boldsymbol{\xi}$ is the design variable vector, $\boldsymbol{\theta}$ is a random vector, and $\mathbb{E}\left[\cdot|\boldsymbol{\xi}\right]$ is the expectation conditioned on $\boldsymbol{\xi}$. Stochastic optimization problems [1–3] are relevant to different fields, such as machine learning [4], Stochastic Optimal Control [5, 6], Computational Finance [7–9], Economics [10], Insurance [11], Communication Networks [12], Queues and Supply Chains [13], and Bayesian Optimal Design of Experiments [14, 15], among many others.

In the same spirit and inspired by the work by Heinrich [16] and Giles [17] on *Multilevel Monte Carlo* methods, we propose the *Multi-Iteration stochastiC Estimator*—MICE—to obtain a computationally efficient approximation of the mean gradient at iteration k, $\nabla_{\boldsymbol{\xi}}\mathbb{E}\left[f(\boldsymbol{\xi},\boldsymbol{\theta})|\boldsymbol{\xi}=\boldsymbol{\xi}_k\right]$, which may be coupled with any first order stochastic optimizer in a non-intrusive fashion. Combining MICE with any stochastic optimizer furnishes *Multi-Iteration Stochastic Optimizers*, a novel class of efficient and robust stochastic optimizers. In this class of stochastic optimizers, the mean gradient estimator's relative variance is controlled using successive control variates based on previous iterations' available information. This procedure results in a more accurate yet cost-effective estimation of the mean gradient. In approximating the mean gradient, MICE constructs an index set of iterations and performs control variates for every pair of nested elements of this index set. As the stochastic optimization evolves, we increase the number of samples along the index set while keeping the previously sampled gradients, i.e., we use gradient information from previous iterations to reduce the variance in the current gradient estimate, which is a crucial feature to make MICE competitive. We design MICE to achieve a given relative error for the mean gradient with minimum additional gradient sampling cost. Indeed, in the MICE index set constructed along the stochastic optimization path $\{\boldsymbol{\xi}_\ell\}_{\ell=0}^k$, our generic optimizer optimally decides whether to drop a particular iteration ℓ out of the index set or restart it to reduce the total optimization work. Moreover, it can decide if it is advantageous, from the computational work perspective, to clip the index set at some point ℓ, discarding iterations before ℓ. Since we control the gradients' error using an estimate of the gradient norm, we propose a resampling technique to get a gradient norm estimate, reducing the effect of sampling error and resulting in robust optimizers. We note in passing that MICE can be adjusted to the case of finite populations; see (2), for optimization problems arising in supervised machine learning.

Generally speaking, in first-order stochastic optimization algorithms that produce convergent iterates, the mean gradient converges to zero as the number of iterations, k, goes to infinity, that is $\left\|\mathbb{E}\left[\nabla_{\boldsymbol{\xi}} f(\boldsymbol{\xi}_k,\boldsymbol{\theta})\right]\right\| \to 0$; however, the gradient covariance, $\mathbb{C}[\nabla_{\boldsymbol{\xi}} f(\boldsymbol{\xi}_k,\boldsymbol{\theta}), \nabla_{\boldsymbol{\xi}} f(\boldsymbol{\xi}_k,\boldsymbol{\theta})]$, does not converge to zero. Thus, to ensure convergence of the iterates $\boldsymbol{\xi}_k$, in the literature it is customary to use decreasing step-size (learn-

ing rate) schedules, reducing the effect of the statistical error in the gradient onto the iterates $\boldsymbol{\xi}_k$. However, this approach also results in sublinear convergence rates [18]. Another approach to deal with the gradient's statistical error is to increase the sample sizes (batch sizes) while keeping the step-size fixed, thus avoiding worsening the convergence. Byrd et al. [19] propose to adaptively increase the sample sizes to guarantee that the trace of the covariance matrix of the mean gradient is proportional to its norm. This approach forces the statistical error to decrease as fast as the gradient norm. Balles et al. [20] use a similar approach; however, instead of setting a parameter to control the statistical error, they set a step-size and find the parameter that guarantees the desired convergence. Bollapragada et al. [21] propose yet another approach to control the variance of gradient estimates in stochastic optimization, which they call the inner product test. Their approach ensures that descent directions are generated sufficiently often.

Relatedly, for certain over-parameterized (interpolating) models it is possible to establish convergence of stochastic methods with constant step-size under the *strong growth condition* [22], which bounds the second moment of the (raw) stochastic gradient by a multiple of $\|\nabla F(\boldsymbol{\xi})\|^2$. This assumption is conceptually different from our approach: rather than postulating a property of the intrinsic gradient noise, we adaptively allocate samples so that the `MICE` mean-gradient estimator satisfies a relative accuracy requirement with user-chosen tolerance ϵ in L^2.

Instead of increasing the sample size, some methods rely on using control variates with respect to previously sampled gradients to reduce the variance in current iterations and thus be able to keep a fixed step-size. Pioneering ideas of control variates in stochastic optimization, by Johnson & Zhang [23], profit on an accurate mean gradient estimation at the initial guess $\boldsymbol{\xi}_0$, $\nabla_{\boldsymbol{\xi}}\mathbb{E}\left[f(\boldsymbol{\xi},\boldsymbol{\theta})|\boldsymbol{\xi}=\boldsymbol{\xi}_0\right]$, to update and compute, via single control variates, an inexpensive and accurate version of the mean gradient at the iteration k, $\nabla_{\boldsymbol{\xi}}\mathbb{E}\left[f(\boldsymbol{\xi},\boldsymbol{\theta})|\boldsymbol{\xi}=\boldsymbol{\xi}_k\right]$. Instead of doing control variates with respect to one starting full-gradient, `SARAH`, by Nguyen et al. [24], computes an estimate of the gradient at the current iteration by using control variates with respect to the last iteration. An 'inexact' version of `SARAH` is presented in [25], where `SARAH` is generalized to the minimization of expectations. In the spirit of successive control variates, `SPIDER` by Fang et al. [26] uses control variates between subsequent iterations; however, it employs the *normalized* gradient descent instead of plain gradient descent. In a different approach, `SAGA`, by Defazio et al. [27], keeps in the memory the last gradient $\nabla_{\boldsymbol{\xi}} f$ observed for each data point and computes $\nabla_{\boldsymbol{\xi}}\mathbb{E}\left[f(\boldsymbol{\xi}_k,\boldsymbol{\theta})|\boldsymbol{\xi}_k\right]$ using control variates with respect to the average of this memory. Lastly, many algorithms try to 'adapt' the initial batch size of the index set of batches using predefined rules, such as exponential or polynomial growth, as presented by Friedlander & Schmidt [28], or based on statistical bounds as discussed by De et al. [29] and Ji et al. [30], to mention a few.

Although our proposed *Multi-Iteration Stochastic Optimizers* share similarities with `SVRG` [23], `SARAH`, and `SPIDER`, our stochastic optimizers distinctly control the relative variance in gradient estimates. We achieve this control by sampling the entire index set of iterations, optimally distributing the samples to minimize the gradient sampling cost. While the previously mentioned methods are devised for finite sum minimization, `MICE` can tackle both finite sum and expectation minimization. More-

over, we provide additional flexibility by including dropping, restart, and clipping operations in the MICE index set updates.

For strongly-convex and L-smooth objective functions, Polyak, in his book [31, Theorem 5, pg 102], shows a convergence rate in the presence of random relative noise. The theorem states a linear (geometric) convergence cq^k in terms of the number of iterations. However, the dependency on the relative noise level, ϵ, of the constants c and q is not made explicit. This work presents the explicit form of these constants and their dependency on ϵ. Using this, we can estimate the total average computational work in stochastic gradient evaluations and optimize it with respect to the controllable relative noise ϵ. Finally, we conclude that to generate an iterate $\boldsymbol{\xi}_k$ such that $\left\|\nabla_{\boldsymbol{\xi}} F(\boldsymbol{\xi}_k)\right\|^2 < tol$, SGD-MICE requires, on average, $\mathcal{O}(tol^{-1})$ stochastic gradient evaluations, while SGD with adaptive batch sizes requires the larger $\mathcal{O}(tol^{-1}\log(tol^{-1}))$, correspondingly, as we establish formally in Sect. 3.2 (Corollaries 5 and 6). While the reuse of previous data causes the MICE estimator to be conditionally biased, we present an analysis for the conditional bias and characterize the L^2 error, including bias and statistical error, which is controlled to achieve convergence of SGD-MICE.

Since MICE is non-intrusive and designed for both continuous and discrete random variables, it can be coupled with most available optimizers with ease. For instance, we couple MICE with SGD [32] and Adam [33], showing the robustness of our approach. The Adam algorithm by Kingma & Ba [33] does not exploit control variates techniques for variance reduction. Instead, it reduces the gradient estimator's variance based on iterate history by adaptive estimates of lower-order moments, behaving similarly to a filter. Thus, the coupling Adam-MICE profits from the information available in the optimizer path in more than one way. More generally, MICE can also be coupled with momentum schemes (e.g., heavy-ball or Nesterov-type momentum) by replacing the stochastic gradient oracle with the MICE estimator. Finally, the reader is referred to the books by Spall [34] and Shapiro, Dentcheva, and Ruszczyński [35] for comprehensive overviews on stochastic optimization.

To assess MICE's applicability, we numerically minimize expectations of continuous and discrete random variables using analytical functions and logistic regression models. Also, we compare SGD-MICE with SVRG, SARAH, SAG, and SAGA in training the logistic regression model with datasets with different sizes and numbers of features.

1.1 Optimization of Expectations and Stochastic Optimizers

To state the stochastic optimization problem, let $\boldsymbol{\xi}$ be the design variable in dimension $d_{\boldsymbol{\xi}}$ and $\boldsymbol{\theta}$ a vector-valued random variable in dimension $d_{\boldsymbol{\theta}}$, whose probability distribution π may depend on $\boldsymbol{\xi}$. Throughout this work we assume that we can produce as many independent identically distributed samples from π as needed. Here, $\mathbb{E}\left[\cdot|\boldsymbol{\xi}\right]$ and $\mathbb{V}\left[\cdot|\boldsymbol{\xi}\right]$ are respectively the expectation and variance operators conditioned on $\boldsymbol{\xi}$. Aiming at optimizing expectations on $\boldsymbol{\xi}$, we state our problem as follows. Find $\boldsymbol{\xi}^*$ such that

$$\boldsymbol{\xi}^* = \underset{\boldsymbol{\xi}\in\mathbb{R}^{d_{\boldsymbol{\xi}}}}{\arg\min}\, \mathbb{E}\left[f(\boldsymbol{\xi},\boldsymbol{\theta})\right], \tag{1}$$

where $f\colon \mathbb{R}^{d_{\boldsymbol{\xi}}} \times \mathbb{R}^{d_{\boldsymbol{\theta}}} \to \mathbb{R}$. Through what follows, let the objective function in our problem be denoted by $F(\boldsymbol{\xi}') := \mathbb{E}\left[f(\boldsymbol{\xi}, \boldsymbol{\theta}) | \boldsymbol{\xi} = \boldsymbol{\xi}'\right]$. In general, function F might not have a unique minimizer, in which case we define Ξ^* as the set of all $\boldsymbol{\xi}^*$ satisfying (1). The case of minimizing a finite sum of functions is of special interest given its importance for training machine learning models in empirical risk minimization tasks,

$$\boldsymbol{\xi}^* = \arg\min_{\boldsymbol{\xi} \in \mathbb{R}^{d_{\boldsymbol{\xi}}}} \frac{1}{N} \sum_{n=1}^{N} f(\boldsymbol{\xi}, \boldsymbol{\theta}_n), \tag{2}$$

where N is usually a large number. Note that the finite sum case is a special case of the expectation minimization, i.e., let $\boldsymbol{\theta}$ be a random variable with probability mass function

$$\mathbb{P}(\boldsymbol{\theta} = \boldsymbol{\theta}_n) = \frac{1}{N}. \tag{3}$$

We recall that throughout this work we assume the sampling distribution π of $\boldsymbol{\theta}$ is independent of $\boldsymbol{\xi}$ (see Assumption 1 and Remark 8); in the finite-sum setting this corresponds to sampling from a fixed population.

In minimizing (1) with respect to the design variable $\boldsymbol{\xi} \in \mathbb{R}^{d_{\boldsymbol{\xi}}}$, SGD is constructed with the following updating rule

$$\boldsymbol{\xi}_{k+1} = \boldsymbol{\xi}_k - \eta_k \boldsymbol{\upsilon}_k, \tag{4}$$

where $\eta_k > 0$ is the step-size at iteration k and $\boldsymbol{\upsilon}_k$ is an estimator of the gradient of F at $\boldsymbol{\xi}_k$. For instance, an unbiased estimator $\boldsymbol{\upsilon}_k$ of the gradient of F at $\boldsymbol{\xi}$ at the iteration k may be constructed by means of a Monte Carlo estimator, namely

$$\nabla_{\boldsymbol{\xi}} F(\boldsymbol{\xi}_k) = \mathbb{E}\left[\nabla_{\boldsymbol{\xi}} f(\boldsymbol{\xi}, \boldsymbol{\theta}) | \boldsymbol{\xi} = \boldsymbol{\xi}_k\right] \approx \boldsymbol{\upsilon}_k := \frac{1}{M} \sum_{\alpha \in \mathcal{I}} \nabla_{\boldsymbol{\xi}} f(\boldsymbol{\xi}_k, \boldsymbol{\theta}_\alpha), \tag{5}$$

with M independent and identically distributed (iid) random variables $\boldsymbol{\theta}_\alpha \sim \pi$ given $\boldsymbol{\xi}_k$, $\alpha \in \mathcal{I}$, with $\mathcal{I}$ being an index set with cardinality $M := |\mathcal{I}|$. Bear in mind that an estimator of the type (5) is, in fact, a random variable and its use in optimization algorithms gives rise to the so-called *Stochastic Optimizers*. The challenge of computing the gradient of F in an affordable and accurate manner motivated the design of several gradient estimators.

For the sake of brevity, the following review on control variates techniques for stochastic optimization is not comprehensive. To motivate our approach, we recall the control variates proposed by Johnson & Zhang [23] (and similarly, by Defazio et al. [27]) for the optimization of a function defined by a finite sum of functions. The idea of control variates is to add and subtract the same quantity, that is, for any $\boldsymbol{\xi}_0$,

$$\nabla_{\boldsymbol{\xi}} F(\boldsymbol{\xi}_k) = \mathbb{E}\left[\nabla_{\boldsymbol{\xi}} f(\boldsymbol{\xi}, \boldsymbol{\theta}) - \nabla_{\boldsymbol{\xi}} f(\boldsymbol{\xi}_0, \boldsymbol{\theta}) | \boldsymbol{\xi} = \boldsymbol{\xi}_k\right] + \mathbb{E}\left[\nabla_{\boldsymbol{\xi}} f(\boldsymbol{\xi}_0, \boldsymbol{\theta})\right], \tag{6}$$

rendering the following sample-based version

$$\nabla_{\boldsymbol{\xi}} F(\boldsymbol{\xi}_k) \approx \frac{1}{M_k} \sum_{\alpha \in \mathcal{I}_k} (\nabla_{\boldsymbol{\xi}} f(\boldsymbol{\xi}_k, \boldsymbol{\theta}_\alpha) - \nabla_{\boldsymbol{\xi}} f(\boldsymbol{\xi}_0, \boldsymbol{\theta}_\alpha)) + \frac{1}{M_0 - M_k} \sum_{\alpha \in \mathcal{I}_0 \setminus \mathcal{I}_k} \nabla_{\boldsymbol{\xi}} f(\boldsymbol{\xi}_0, \boldsymbol{\theta}_\alpha), \tag{7}$$

where $M_0 \gg M_k$ and $\boldsymbol{\theta}_\alpha$ are iid samples from the π distribution, which does not depend on $\boldsymbol{\xi}$ in their setting. In the original work by Johnson and Zhang [23], M_0 is the total population and $M_k = 1$. Later, Nitanda [36] and Konečnỳ et al. [37] also used the total populations M_0 at $\boldsymbol{\xi}_0$, but with $M_k = 2, 4, 8 \ldots$, to study the efficiency of the algorithm. Additionally, the work [23] restarts the algorithm after a pre-established number of iterations by setting $\boldsymbol{\xi}_0 \leftarrow \boldsymbol{\xi}_k$. The efficiency of this algorithm relies on the correlation between the components of the gradients $\nabla_{\boldsymbol{\xi}} F(\boldsymbol{\xi}_0)$ and $\nabla_{\boldsymbol{\xi}} F(\boldsymbol{\xi}_k)$. If this correlation is high, the variance of the mean gradient estimator (7) is reduced.

1.2 Paper Outline

The remainder of this work is as follows. In Sect. 1, we describe the stochastic optimization problem, classical stochastic optimization methods and motivate variance reduction in this context. In Sect. 2, we construct the `MICE` statistical estimator (Sect. 2.1); analyze its error (Sect. 2.2); compute the optimal number of samples for the current index set (Sect. 2.3); present the operators used to build `MICE`'s index set and derive a work-based criterion to choose one (Sect. 2.4). In Sect. 3, we present a convergence analysis of L^2 error-controlled `SGD`, which includes `SGD-MICE`, showing these converge polynomially for general L-smooth problems, and exponentially if the objective function is gradient-dominated (Sect. 3.1). In Sect. 3.2 we present gradient sampling cost analyses for `SGD-MICE` and `SGD-A` (`SGD` with adaptive increase in the sample sizes) on expectation minimization (Sect. 3.2.1) and finite sum minimization Sect. 3.2.2). In Sect. 4, practical matters related to implementation of the `MICE` estimator are discussed. In Sect. 5, to assess the efficiency of *Multi-Iteration Stochastic Optimizers*, we present some numerical examples, ranging from analytical functions to the training of a logistic regression model over datasets of size up to 11×10^6. In Appendix A, are presented detailed pseudocodes for the *Multi-Iteration Stochastic Optimizers* used in this work.

2 Multi-iteration Stochastic Optimizers

2.1 Multi-iteration Gradient Estimator

We now construct an efficient estimator of the mean gradient at the current iteration k, $\nabla_{\boldsymbol{\xi}} F(\boldsymbol{\xi}_k) = \mathbb{E}\left[\nabla_{\boldsymbol{\xi}} f(\boldsymbol{\xi}_k, \boldsymbol{\theta}) | \boldsymbol{\xi}_k\right]$, which we name *Multi-Iteration stochastiC Estimator*—`MICE`. Profiting from available information already computed in previous iterations, `MICE` uses multiple control variates between pairs of, possibly non-consecutive, iterations along the optimization path to approximate the mean gradient at iteration k. Bearing in mind that stochastic optimization algorithms, in a broad

sense, create an L^2 convergent path where $\mathbb{E}\left[\left\|\boldsymbol{\xi}_k - \boldsymbol{\xi}_\ell\right\|^2\right] \to 0$ as $\ell, k \to \infty$, the gradients evaluated at $\boldsymbol{\xi}_\ell$ and $\boldsymbol{\xi}_k$ should become more and more correlated for $k, \ell \to \infty$. In this scenario, control variates with respect to previous iterations become more efficient, in the sense that one needs fewer and fewer new samples to accurately estimate the mean gradient.

To motivate our MICE estimator, first consider the following special case:

Remark 1 (*Add-only special case*) A simple and special case of MICE is the *add-only* case:

$$\nabla_{\boldsymbol{\xi}} \mathcal{F}_k = \frac{1}{M_{0,k}} \sum_{\alpha \in \mathcal{I}_{0,k}} \nabla_{\boldsymbol{\xi}} f(\boldsymbol{\xi}_0, \boldsymbol{\theta}_\alpha) + \sum_{\ell=1}^{k} \frac{1}{M_{\ell,k}} \sum_{\alpha \in \mathcal{I}_{\ell,k}} \left(\nabla_{\boldsymbol{\xi}} f(\boldsymbol{\xi}_\ell, \boldsymbol{\theta}_\alpha) - \nabla_{\boldsymbol{\xi}} f(\boldsymbol{\xi}_{\ell-1}, \boldsymbol{\theta}_\alpha)\right), \tag{8}$$

which is identical to SARAH but with *cumulative sampling*. Note that the sample sizes $M_{\ell,k}$ change both with ℓ and k.

Now, we generalize our estimator with an adaptive selection of which iterates $\boldsymbol{\xi}_\ell$ we want to consider in the MICE estimator.

First let us establish some notation. Let $\mathcal{L}_k$ be an index set, such that, $\mathcal{L}_k \subset \{0, \ldots, k\}$, where k is the current iteration and $k \in \mathcal{L}_k$. This index set is $\mathcal{L}_0 = \{0\}$ at the initial iteration, $k = 0$, and for later iterations it contains the indices of the iterations MICE uses to reduce the computational work at the current iteration, $k > 0$, via control variates. For the special case in Remark 1 the index set is $\mathcal{L}_k = \{0, 1, \ldots, k\}$. Next, for any $\min\{\mathcal{L}_k\} < \ell \in \mathcal{L}_k$, let $p_k(\ell)$ be the element *previous* to ℓ in $\mathcal{L}_k$,

$$p_k(\ell) \coloneqq \max\{\ell' \in \mathcal{L}_k \colon \ell' < \ell\}. \tag{9}$$

Then, the mean gradient at $\boldsymbol{\xi}_k$ conditioned on the sequence of random iterates, $\boldsymbol{\xi}$, indexed by the set $\mathcal{L}_k$ can be decomposed as

$$\nabla_{\boldsymbol{\xi}} F(\boldsymbol{\xi}_k) = \mathbb{E}\left[\nabla_{\boldsymbol{\xi}} f(\boldsymbol{\xi}_k, \boldsymbol{\theta}) | \{\boldsymbol{\xi}_\ell\}_{\ell \in \mathcal{L}_k}\right] = \sum_{\ell \in \mathcal{L}_k} \boldsymbol{\mu}_{\ell,k}, \qquad \boldsymbol{\mu}_{\ell,k} \coloneqq \mathbb{E}\left[\Delta_{\ell,k} | \boldsymbol{\xi}_\ell, \boldsymbol{\xi}_{p_k(\ell)}\right], \tag{10}$$

with the gradient difference notation

$$\Delta_{\ell,k} \coloneqq \begin{cases} \nabla_{\boldsymbol{\xi}} f(\boldsymbol{\xi}_\ell, \boldsymbol{\theta}) - \nabla_{\boldsymbol{\xi}} f(\boldsymbol{\xi}_{p_k(\ell)}, \boldsymbol{\theta}), & \text{if } \ell > \min\{\mathcal{L}_k\}, \\ \nabla_{\boldsymbol{\xi}} f(\boldsymbol{\xi}_\ell, \boldsymbol{\theta}), & \text{if } \ell = \min\{\mathcal{L}_k\}. \end{cases} \tag{11}$$

Thus, the conditional mean $\boldsymbol{\mu}_{\ell,k}$ defined in (10) is simply

$$\boldsymbol{\mu}_{\ell,k} = \begin{cases} \nabla_{\boldsymbol{\xi}} F(\boldsymbol{\xi}_\ell) - \nabla_{\boldsymbol{\xi}} F(\boldsymbol{\xi}_{p_k(\ell)}), & \text{if } \ell > \min\{\mathcal{L}_k\}, \\ \nabla_{\boldsymbol{\xi}} F(\boldsymbol{\xi}_\ell), & \text{if } \ell = \min\{\mathcal{L}_k\}. \end{cases} \tag{12}$$

For readability, we make the assumption that the distribution of $\boldsymbol{\theta}$ does not depend on $\boldsymbol{\xi}$. Observe that this assumption is more general than it may seem, see the discussion on Remark 8.

Assumption 1 (Simplified probability distribution of $\boldsymbol{\theta}$) The probability distribution of $\boldsymbol{\theta}$, π, does not depend on $\boldsymbol{\xi}$.

Now we are ready to introduce the MICE gradient estimator.

Definition 1 (MICE *gradient estimator*) Given an index set $\mathcal{L}_k$ such that $k \in \mathcal{L}_k \subset \{0, \ldots, k\}$ and positive integer numbers $\{M_{\ell,k}\}_{\ell \in \mathcal{L}_k}$, we define the MICE gradient estimator for $\nabla_{\boldsymbol{\xi}} F(\boldsymbol{\xi}_k)$ at iteration k as

$$\nabla_{\boldsymbol{\xi}} \mathcal{F}_k = \sum_{\ell \in \mathcal{L}_k} \hat{\boldsymbol{\mu}}_{\ell,k}, \qquad \hat{\boldsymbol{\mu}}_{\ell,k} \coloneqq \frac{1}{M_{\ell,k}} \sum_{\alpha \in \mathcal{I}_{\ell,k}} \Delta_{\ell,k,\alpha}, \tag{13}$$

where, for each index $\ell \in \mathcal{L}_k$, the set of samples, $\mathcal{I}_{\ell,k}$, has cardinality $M_{\ell,k}$. Finally, denote as before the difference to the previous gradient as

$$\Delta_{\ell,k,\alpha} \coloneqq \begin{cases} \nabla_{\boldsymbol{\xi}} f(\boldsymbol{\xi}_\ell, \boldsymbol{\theta}_\alpha) - \nabla_{\boldsymbol{\xi}} f(\boldsymbol{\xi}_{p_k(\ell)}, \boldsymbol{\theta}_\alpha), \text{ if } \ell > \min\{\mathcal{L}_k\}, \\ \nabla_{\boldsymbol{\xi}} f(\boldsymbol{\xi}_\ell, \boldsymbol{\theta}_\alpha), \text{ if } \ell = \min\{\mathcal{L}_k\}. \end{cases} \tag{14}$$

For each $\ell \in \mathcal{L}_k$, we might increase the sample sizes $M_{\ell,k}$ with respect to $M_{\ell,k-1}$, hence the dependence on both ℓ and k on the notation. Definition 1 allows us to manipulate the MICE index set to improve its efficiency; one can pick which ℓ to keep in $\mathcal{L}_k$. For example, $\mathcal{L}_k = \{0, k\}$ furnishes an SVRG-like index set, $\mathcal{L}_k = \{0, 1, ..., k\}$ furnishes a SARAH-like index set, and $\mathcal{L}_k = \{k\}$ results in SGD. A description of these baseline variance reduction methods and their connection to MICE are presented in Appendix B, C, D. The construction of the index set $\mathcal{L}_k$ is discussed in Sect. 2.4.

Remark 2 (*Cumulative sampling in* MICE) As the stochastic optimization progresses, new additional samples of $\boldsymbol{\theta}$ are taken and others, already available from previous iterations, are reused to compute the MICE estimator at the current iteration,

$$\nabla_{\boldsymbol{\xi}} \mathcal{F}_k = \underbrace{\sum_{\ell \in \mathcal{L}_k \cap \mathcal{L}_{k-1}} \frac{M_{\ell,k-1}}{M_{\ell,k}} \hat{\boldsymbol{\mu}}_{\ell,k-1}}_{\text{sunken cost}} + \underbrace{\sum_{\ell \in \mathcal{L}_k \cap \mathcal{L}_{k-1}} \frac{1}{M_{\ell,k}} \sum_{\alpha \in \mathcal{I}_{\ell,k} \setminus \mathcal{I}_{\ell,k-1}} \Delta_{\ell,k,\alpha} + \hat{\boldsymbol{\mu}}_{k,k}}_{\text{Additional MICE cost incurred at iteration } k}. \tag{15}$$

This sampling procedure is defined by the couples $(M_{\ell,k}, \boldsymbol{\xi}_\ell)_{\ell \in \mathcal{L}_k}$, making $\boldsymbol{\xi}_{k+1}$ a deterministic function of all the samples in the index set $\mathcal{L}_k$.

Remark 3 (*Conditional bias of the* MICE *estimator*) As described in Remark 2, MICE reuses samples generated at previous iterations, and because the iterate path is itself a function of past samples, the estimator $\nabla \mathcal{F}_k$ is generally *not* unbiased when conditioning on the iterate history $\left\{\boldsymbol{\xi}_{\ell'}\right\}_{\ell' \in \mathcal{L}_k}$. In other words, the conditional bias

$$\boldsymbol{b}_k \coloneqq \mathbb{E}\left[\nabla \mathcal{F}_k \mid \left\{\boldsymbol{\xi}_{\ell'}\right\}_{\ell' \in \mathcal{L}_k} \right] - \nabla_{\boldsymbol{\xi}} F(\boldsymbol{\xi}_k) \tag{16}$$

may be nonzero even though $\nabla \mathcal{F}_k$ is built from unbiased per-sample gradients. Accordingly, the squared L^2 error admits the decomposition

$$\mathbb{E}\left[\left\|\nabla \mathcal{F}_k - \nabla_{\boldsymbol{\xi}} F(\boldsymbol{\xi}_k)\right\|^2\right] = \underbrace{\mathbb{E}\left[\left\|\nabla \mathcal{F}_k - \mathbb{E}\left[\nabla \mathcal{F}_k \mid \{\boldsymbol{\xi}_{\ell'}\}_{\ell' \in \mathcal{L}_k}\right]\right\|^2\right]}_{\text{statistical error}} + \underbrace{\mathbb{E}\left[\|\boldsymbol{b}_k\|^2\right]}_{\text{bias contribution}}. \tag{17}$$

In Appendix B, C, D we derive an explicit expression for $\boldsymbol{b}_k$ (and $\mathbb{E}\left[\|\boldsymbol{b}_k\|^2\right]$) in terms of the previously computed level estimators. Moreover, restarting the index set $\mathcal{L}_k$ resets the conditional bias.

Remark 4 (*About* `MICE` *and* `MLMC`) Note that `MICE` resembles the estimator obtained in the *Multilevel Monte Carlo* method—`MLMC` [17, 38, 39]. For instance, if $\mathcal{L}_k = \{0, 1, \ldots, k\}$, `MICE` reads

$$\nabla_{\boldsymbol{\xi}} \mathcal{F}_k = \frac{1}{M_{0,k}} \sum_{\alpha \in \mathcal{I}_{0,k}} \nabla_{\boldsymbol{\xi}} f(\boldsymbol{\xi}_0, \boldsymbol{\theta}_\alpha) + \sum_{\ell=1}^{k} \frac{1}{M_{\ell,k}} \sum_{\alpha \in \mathcal{I}_{\ell,k}} \nabla_{\boldsymbol{\xi}} f(\boldsymbol{\xi}_\ell, \boldsymbol{\theta}_\alpha) - \nabla_{\boldsymbol{\xi}} f(\boldsymbol{\xi}_{\ell-1}, \boldsymbol{\theta}_\alpha). \tag{18}$$

Indeed, we may think that in `MICE`, the iterations play the same role as the levels of approximation in `MLMC`. However, there are several major differences with `MLMC`, namely *i*) `MICE` exploits sunk cost of previous computations, computing afresh only what is necessary to have enough accuracy on the current iteration *ii*) there is dependence in `MICE` across iterations and *iii*) in `MICE`, the sample cost for the gradients is the same in different iterations while in `MLMC` one usually has higher cost per sample for deeper, more accurate levels.

Indeed, assuming the availability of a convergent hierarchy of approximations and following the `MLMC` lines, the work [40] proposed and analyzed multilevel stochastic approximation algorithms, essentially recovering the classical error bounds for multilevel Monte Carlo approximations in this more complex context. In a similar `MLMC` hierarchical approximation framework, the work by Yang, Wang, and Fang [41] proposed a stochastic gradient algorithm for solving optimization problems with nested expectations as objective functions. Last, the combination of `MICE` and the `MLMC` ideas like those in [40] and [41] is thus a natural research avenue to pursue.

2.2 `MICE` Estimator Mean Squared Error

To determine the optimal number of samples per iteration $\ell \in \mathcal{L}_k$, we begin by defining the square of the error, $\mathcal{E}$, as the squared L^2-distance between `MICE` estimator (13) and the true gradient conditioned on the iterates generated up to k, which leads to

$$(\mathcal{E}_k)^2 \coloneqq \mathbb{E}\left[\left\|\nabla_{\boldsymbol{\xi}} \mathcal{F}_k - \nabla_{\boldsymbol{\xi}} F(\boldsymbol{\xi}_k)\right\|^2 \,\middle|\, \{\boldsymbol{\xi}_{\ell'}\}_{\ell' \in \mathcal{L}_k}\right]. \tag{19}$$

Next we prove that the expected L^2 error of the MICE estimator is identical to the expectation of the contribution of the statistical error of each element of the index set. Before we start, let's prove the following Lemma.

Lemma 1 *Let $\hat{\boldsymbol{\mu}}_{\ell,k}$, as defined in (13), be generated by a multi-iteration stochastic optimizer using MICE as a gradient estimator. Then, for $j \neq \ell$,*

$$\mathbb{E}\left[\langle \hat{\boldsymbol{\mu}}_{\ell,k} - \boldsymbol{\mu}_{\ell,k}, \hat{\boldsymbol{\mu}}_{j,k} - \boldsymbol{\mu}_{j,k} \rangle\right] = 0. \tag{20}$$

Proof First, let us assume $j > \ell$ without loss of generality. Note that the samples of $\boldsymbol{\theta}$ used to compute $\hat{\boldsymbol{\mu}}_{\ell,k}$ and $\hat{\boldsymbol{\mu}}_{j,k}$ are independent. However, the iterates $\{\boldsymbol{\xi}_m\}_{m=\ell+1}^{j}$ depend on $\hat{\boldsymbol{\mu}}_{\ell,k}$, thus, $\hat{\boldsymbol{\mu}}_{\ell,k}$ and $\hat{\boldsymbol{\mu}}_{j,k}$ are not independent. To prove Lemma 1, let us use the law of total expectation to write the expectation above as the expectation of an expectation conditioned on $\left\{\boldsymbol{\xi}_{\ell'}\right\}_{\ell' \in \mathcal{L}_j}$. Since $\hat{\boldsymbol{\mu}}_{\ell,k}$ and $\hat{\boldsymbol{\mu}}_{j,k}$ are then conditionally independent,

$$\begin{aligned}
&\mathbb{E}\left[\langle \hat{\boldsymbol{\mu}}_{\ell,k} - \boldsymbol{\mu}_{\ell,k}, \hat{\boldsymbol{\mu}}_{j,k} - \boldsymbol{\mu}_{j,k} \rangle\right] \\
&\quad = \mathbb{E}\left[\mathbb{E}\left[\langle \hat{\boldsymbol{\mu}}_{\ell,k} - \boldsymbol{\mu}_{\ell,k}, \hat{\boldsymbol{\mu}}_{j,k} - \boldsymbol{\mu}_{j,k} \rangle \,\middle|\, \left\{\boldsymbol{\xi}_{\ell'}\right\}_{\ell' \in \mathcal{L}_j}\right]\right] \\
&\quad = \mathbb{E}\left[\left\langle \underbrace{\mathbb{E}\left[\hat{\boldsymbol{\mu}}_{\ell,k} - \boldsymbol{\mu}_{\ell,k} \,\middle|\, \left\{\boldsymbol{\xi}_{\ell'}\right\}_{\ell' \in \mathcal{L}_j}\right]}_{\neq 0}, \underbrace{\mathbb{E}\left[\hat{\boldsymbol{\mu}}_{j,k} - \boldsymbol{\mu}_{j,k} \,\middle|\, \left\{\boldsymbol{\xi}_{\ell'}\right\}_{\ell' \in \mathcal{L}_j}\right]}_{=0} \right\rangle\right],
\end{aligned} \tag{21}$$

concluding the proof. □

Although the level estimators are not independent unconditionally (since later iterates depend on earlier samples), Lemma 1 shows that the cross terms vanish after conditioning on the iterate history: the samples used at different levels are independent and each level error has zero conditional mean, hence $\mathbb{E}\left[\langle \hat{\boldsymbol{\mu}}_{\ell,k} - \boldsymbol{\mu}_{\ell,k}, \hat{\boldsymbol{\mu}}_{j,k} - \boldsymbol{\mu}_{j,k} \rangle\right] = 0$ for $j \neq \ell$.

Let $\Delta_{\ell,k}^{(i)}$ be the i-th component of the $d_{\boldsymbol{\xi}}$ dimensional vector $\Delta_{\ell,k}$. Then, we define

$$V_{\ell,k} \coloneqq \sum_{i=1}^{d_{\boldsymbol{\xi}}} \mathbb{V}\left[\Delta_{\ell,k}^{(i)} \middle| \{\boldsymbol{\xi}_{\ell'}\}_{\ell' \in \mathcal{L}_k}\right]. \tag{22}$$

Lemma 2 (Expected squared L^2 error of the MICE estimator for expectation minimization) *The expected mean squared error of the MICE estimator is given by*

$$\mathbb{E}\left[(\mathcal{E}_k)^2\right] = \mathbb{E}\left[\sum_{\ell \in \mathcal{L}_k} \frac{V_{\ell,k}}{M_{\ell,k}}\right], \tag{23}$$

where $V_{\ell,k}$ is as in (22).

Proof The mean squared error of the MICE estimator is

$$\mathbb{E}\left[\left\|\nabla\mathcal{F}_k - \nabla_{\boldsymbol{\xi}} F(\boldsymbol{\xi}_k)\right\|^2\right] = \mathbb{E}\left[\left\|\sum_{\ell\in\mathcal{L}_k}(\hat{\boldsymbol{\mu}}_{\ell,k} - \boldsymbol{\mu}_{\ell,k})\right\|^2\right]$$
$$= \mathbb{E}\left[\sum_{\ell\in\mathcal{L}_k}\left\|\hat{\boldsymbol{\mu}}_{\ell,k} - \boldsymbol{\mu}_{\ell,k}\right\|^2\right]$$
$$+ 2\sum_{\ell\in\mathcal{L}_k}\sum_{j\in\mathcal{L}_k:j>\ell}\mathbb{E}\left[\left\langle\hat{\boldsymbol{\mu}}_{\ell,k} - \boldsymbol{\mu}_{\ell,k}, \hat{\boldsymbol{\mu}}_{j,k} - \boldsymbol{\mu}_{j,k}\right\rangle\right]. \quad (24)$$

Thus, using Lemma (1) and the law of total expectation,

$$\mathbb{E}\left[\left\|\nabla\mathcal{F}_k - \nabla_{\boldsymbol{\xi}} F(\boldsymbol{\xi}_k)\right\|^2\right] = \mathbb{E}\left[\sum_{\ell\in\mathcal{L}_k}\mathbb{E}\left[\left\|\hat{\boldsymbol{\mu}}_{\ell,k} - \boldsymbol{\mu}_{\ell,k}\right\|^2 \,\middle|\, \{\boldsymbol{\xi}_{\ell'}\}_{\ell'\in\mathcal{L}_\ell}\right]\right]. \quad (25)$$

Since

$$\mathbb{E}\left[\left\|\hat{\boldsymbol{\mu}}_{\ell,k} - \boldsymbol{\mu}_{\ell,k}\right\|^2 \,\middle|\, \{\boldsymbol{\xi}_{\ell'}\}_{\ell'\in\mathcal{L}_\ell}\right]$$
$$= \mathbb{E}\left[\left\|\frac{1}{M_{\ell,k}}\sum_{\alpha\in\mathcal{I}_{\ell,k}}\Delta_{\ell,k,\alpha} - \mathbb{E}\left[\Delta_{\ell,k} \,\middle|\, \boldsymbol{\xi}_\ell, \boldsymbol{\xi}_{p_k(\ell)}\right]\right\|^2 \,\middle|\, \boldsymbol{\xi}_\ell, \boldsymbol{\xi}_{p_k(\ell)}\right] \quad (26)$$
$$= \frac{\sum_{i=1}^{d_{\boldsymbol{\xi}}}\mathbb{V}\left[\Delta_{\ell,k}^{(i)} \,\middle|\, \{\boldsymbol{\xi}_{\ell'}\}_{\ell'\in\mathcal{L}_k}\right]}{M_{\ell,k}}, \quad (27)$$

using (22) concludes the proof. □

Remark 5 (*Expected squared L^2 error of the* MICE *estimator for finite sum minimization*) When minimizing a finite sum of functions as in (2), we sample the random variables $\boldsymbol{\theta}$ without replacement. Thus, the variance of the estimator should account for the ratio between the actual number of samples $M_{\ell,k}$ used in the estimator and the total population N [42, Section 3.7]. In this case, the error analysis is identical to the expectation minimization case up to (26), except in this case we include the correction factor $(N - M_{\ell,k})N^{-1}$ in the sample variance due to the finite population having size N, resulting in

$$\mathbb{E}\left[(\mathcal{E}_k)^2\right] = \mathbb{E}\left[\sum_{\ell\in\mathcal{L}_k}\frac{V_{\ell,k}}{M_{\ell,k}}\left(\frac{N - M_{\ell,k}}{N}\right)\right]. \quad (28)$$

Note that, in practice, the terms $V_{\ell,k}$ are computed using sample approximations for each $\ell \in \mathcal{L}_k$. In the convergence analysis in Sect. 3, we assume that they are computed exactly. The squared L^2 error of MICE can be decomposed in bias and statistical error, which are analyzed in Appendix B, C, D.

2.3 Multi-iteration Optimal Setting for Gradient Error Control

First, let the gradient sampling cost and the total MICE work be defined as follows. The number of gradient evaluations is 1 for $\Delta_{\ell,k,\alpha}$ when $\ell = \min\{\mathcal{L}_k\}$ and 2 otherwise. For this reason, we define the auxiliary index function

$$\mathbb{1}_{\underline{\mathcal{L}_k}}(\ell) \coloneqq \begin{cases} 0 & \text{if} \quad \ell = \min\{\mathcal{L}_k\}, \\ 1 & \text{otherwise} \end{cases}, \tag{29}$$

and define the gradient sampling cost in number of gradient evaluations as

$$\mathcal{C}(\{M_{\ell,k}\}_{\ell\in\mathcal{L}_k}) \coloneqq \sum_{\ell\in\mathcal{L}_k} (1 + \mathbb{1}_{\underline{\mathcal{L}_k}}(\ell)) M_{\ell,k}. \tag{30}$$

Motivated by the analysis of SGD-MICE in Sect. 3, here we choose the number of samples for the index set $\mathcal{L}_k$ by approximate minimization of the gradient sampling cost (30) subject to a given tolerance $\epsilon > 0$ on the relative error in the mean gradient approximation, that is

$$\begin{aligned} \{M^*_{\ell,k}\}_{\ell\in\mathcal{L}_k} &= \underset{\{M_{\ell,k}\}_{\ell\in\mathcal{L}_k}}{\arg\min}\, \mathcal{C}\left(\{M_{\ell,k}\}_{\ell\in\mathcal{L}_k}\right) \\ \text{subject to} \quad & (\mathcal{E}_k)^2 \leq \epsilon^2 \left\|\nabla_{\boldsymbol{\xi}} F(\boldsymbol{\xi}_k)\right\|^2. \end{aligned} \tag{31}$$

2.3.1 Expectation Minimization

As a consequence of (31) and Lemma 2, we define the sample sizes as the solution of the following constrained optimization problem,

$$\begin{aligned} \{M^*_{\ell,k}\}_{\ell\in\mathcal{L}_k} &= \underset{\{M_{\ell,k}\}_{\ell\in\mathcal{L}_k}}{\arg\min}\, \mathcal{C}\left(\{M_{\ell,k}\}_{\ell\in\mathcal{L}_k}\right) \\ &\text{subject to} \quad \sum_{\ell\in\mathcal{L}_k} \frac{V_{\ell,k}}{M_{\ell,k}} \leq \epsilon^2 \left\|\nabla_{\boldsymbol{\xi}} F(\boldsymbol{\xi}_k)\right\|^2. \end{aligned} \tag{32}$$

An approximate integer-valued solution based on Lagrangian relaxation to problem (32) is

$$M^*_{\ell,k} = \left\lceil \frac{1}{\epsilon^2 \left\|\nabla_{\boldsymbol{\xi}} F(\boldsymbol{\xi}_k)\right\|^2} \left(\sum_{\ell'\in\mathcal{L}_k} \sqrt{V_{\ell',k}(1 + \mathbb{1}_{\underline{\mathcal{L}_k}}(\ell'))} \right) \sqrt{\frac{V_{\ell,k}}{(1 + \mathbb{1}_{\underline{\mathcal{L}_k}}(\ell))}} \right\rceil, \quad \forall \ell \in \mathcal{L}_k. \tag{33}$$

In general, in considering the cost of computing new gradients at the iteration k, the expenditure already carried out up to the iteration $k-1$ is sunk cost and must not be included, as described in Remark 2, that is, one should only consider the incremental cost of going from $k-1$ to k. Moreover, in the variance constraint of problem (32), since we do not have access to the norm of the mean gradient, $\left\|\nabla_{\boldsymbol{\xi}} F(\boldsymbol{\xi}_k)\right\|$, we use

a resampling technique combined with the MICE estimator as an approximation; see Remark 9.

2.3.2 Finite Sum Minimization

In view of Remark 5, we define the sample sizes for MICE as the solution of the following optimization problem,

$$\text{find } \{M^*_{\ell,k}\} = \underset{\{M_{\ell,k}\}_{\ell\in\mathcal{L}_k}}{\arg\min}\ \mathcal{C}\left(\{M_{\ell,k}\}_{\ell\in\mathcal{L}_k}\right) \tag{34}$$
$$\text{subject to } \begin{cases} \sum_{\ell\in\mathcal{L}_k} \frac{V_{\ell,k}}{M_{\ell,k}} - \frac{V_{\ell,k}}{N} \le \epsilon^2 \left\|\nabla_{\boldsymbol{\xi}} F\left(\boldsymbol{\xi}_k\right)\right\|^2 \\ M_{min} \le M_{\ell,k} \le N \quad \forall \ell \in \mathcal{L}_k. \end{cases}$$

This problem does not have a closed form solution, but can be solved in an iterative process by noting that any ℓ such that $M_{\ell,k} = N$ does not contribute to the error of the estimator. Then, letting

$$\mathcal{G}_k = \{\ell \in \mathcal{L}_k : M_\ell < N\}, \tag{35}$$

we derive a closed form solution for the sample sizes as

$$M^*_{\ell,k} = \left\lceil \frac{\sum_{\ell'\in\mathcal{G}_k} \sqrt{(1+\mathbb{1}_{\underline{\mathcal{L}_{\ell'}}}(k))V_{\ell,k}}}{\epsilon^2 \left\|\nabla_{\boldsymbol{\xi}} F\left(\boldsymbol{\xi}_k\right)\right\|^2 + N^{-1}\sum_{\ell''\in\mathcal{G}_k} V_{\ell'',k}} \sqrt{\frac{V_{\ell,k}}{(1+\mathbb{1}_{\underline{\mathcal{L}_k}}(\ell))}} \right\rceil. \tag{36}$$

However, it is not possible to know directly the set $\mathcal{G}_k$. So, we initialize $\mathcal{G}_k = \mathcal{L}_k$ and iteratively remove elements that do not satisfy the condition $M_\ell < N$ as presented in Algorithm 1.

2.4 Optimal Index Set Operators

As for the construction of the MICE index set at iteration k, that is, $\mathcal{L}_k$, from the previous one, $\mathcal{L}_{k-1}$, we use one of the following index set operators:

Definition 2 (*Construction of the index set* $\mathcal{L}_k$) For $k = 0$, let $\mathcal{L}_0 = \{0\}$. If $k \ge 1$, After this step, there are four possible cases to finish the construction of $\mathcal{L}_k$:

`Add`:	$\mathcal{L}_k$	$\leftarrow \mathcal{L}_k^{\text{add}}$	$\leftarrow \mathcal{L}_{k-1} \cup \{k\}$
`Drop`:	$\mathcal{L}_k$	$\leftarrow \mathcal{L}_k^{\text{drop}}$	$\leftarrow \mathcal{L}_{k-1} \cup \{k\} \setminus \{k-1\}$
`Restart`:	$\mathcal{L}_k$	$\leftarrow \mathcal{L}_k^{\text{rest}}$	$\leftarrow \{k\}$
`Clip` at ℓ^*:	$\mathcal{L}_k$	$\leftarrow \mathcal{L}_k^{\text{clip},\ell^*}$	$\leftarrow \mathcal{L}_{k-1} \cup \{k\} \setminus \{\ell \in \mathcal{L}_{k-1} : \ell < \ell^*\}$

The `Add` operator simply adds k to the current index set. The `Drop` operator does the same but also removes $k-1$ from the index set. As the name suggests, `Restart` resets the index set at the current iterate. Finally, `Clip` adds k to the current index

Algorithm 1 Computing sample size of SGD-MICE for the finite sum case.

$\mathcal{G}_k \leftarrow \mathcal{L}_k$
Set $M_{\ell,k}$ using (36) for all $\ell \in \mathcal{G}_k$
while any $\{\ell \in \mathcal{G}_k : M_{\ell,k} \geq N\}$ **do**
  for $\ell \in \{\ell \in \mathcal{G}_k : M_{\ell,k} \geq N\}$ **do**
    $M_{\ell,k} \leftarrow N$
    $\mathcal{G}_k \leftarrow \mathcal{G}_k \setminus \{\ell\}$
  end for
  Set $M_{\ell,k}$ using (36) for all $\ell \in \mathcal{G}_k$
end while
Return $\{\lceil M_{\ell,k} \rceil\}_{\ell \in \mathcal{L}_k}$

set and removes all components previous to j. For more details, see Sect. 4 for an algorithmic description.

In the previous section, the sample sizes for each element of the index set are chosen as to minimize the gradient sampling cost while satisfying a relative error constraint. However, to pick one of the operators to update the index set, we must use the work including the overhead of aggregating the index set elements. Let the gradient sampling cost increment at iteration k be

$$\Delta\mathcal{C}_k(\mathcal{L}) = \sum_{\ell \in \mathcal{L} \cap \mathcal{L}_{k-1}} (1 + \mathbb{1}_{\underline{\mathcal{L}_k}}(\ell))(M^*_{\ell,k} - M_{\ell,k-1}) + (1 + \mathbb{1}_{\underline{\mathcal{L}_k}}(k))M^*_{k,k}, \tag{37}$$

with $M^*_{\ell,k}$ as in (33) or Algorithm 1.

The total work of a MICE evaluation is then the sum of the cost of sampling the gradients and the cost of aggregating the gradients as

$$\mathcal{W}\left(\{M_{\ell,k}\}_{\ell \in \mathcal{L}_k}\right) := \mathcal{C}(\{M_{\ell,k}\}_{\ell \in \mathcal{L}_k})C_\nabla + |\mathcal{L}_k|C_{\text{aggr}}, \tag{38}$$

where C_∇ is the work of sampling $\nabla_{\boldsymbol{\xi}} f$ and C_{aggr} is the work of averaging the $\Delta_{\ell,k}$ to construct $\mathcal{F}_k$. Then, the work done in iteration k to update MICE is

$$\Delta\mathcal{W}(\mathcal{L}) := \Delta\mathcal{C}_k(\mathcal{L})C_\nabla + |\mathcal{L}|C_{\text{aggr}}. \tag{39}$$

We choose the index set operator for iteration k by a greedy work-based policy. We then accept drop/restart operations if they are not significantly more expensive than

continuing, and accept clipping if it strictly reduces the work:

$$\begin{aligned}
&\text{Initialize} \quad \mathcal{L}_k \leftarrow \mathcal{L}_k^{\text{add}},\\
&\text{(Add) Let } \Delta\mathcal{W}_{\text{add}} \coloneqq \Delta\mathcal{W}_k(\mathcal{L}_k^{\text{add}})\\
&\text{(Drop) if} \quad \Delta\mathcal{W}_k(\mathcal{L}_k^{\text{drop}}) \le (1+\delta_{\text{drop}})\,\Delta\mathcal{W}_{\text{add}} \quad \text{then set} \quad \mathcal{L}_k \leftarrow \mathcal{L}_k^{\text{drop}},\\
&\text{(Restart) if} \quad \Delta\mathcal{W}_k(\mathcal{L}_k^{\text{rest}}) < (1+\delta_{\text{rest}})\,\Delta\mathcal{W}_k(\mathcal{L}_k) \quad \text{then set} \quad \mathcal{L}_k \leftarrow \mathcal{L}_k^{\text{rest}},\\
&\text{(Clip) if} \quad \min_{\ell\in\mathcal{L}_{k-1}} \Delta\mathcal{W}_k(\mathcal{L}_k^{\text{clip},\ell}) < \Delta\mathcal{W}_k(\mathcal{L}_k) \quad \text{then clip at the minimizing } \ell,
\end{aligned} \tag{40}$$

and otherwise we keep $\mathcal{L}_k$ as selected by the checks above (initialized as $\mathcal{L}_k^{\text{add}}$). Here $\mathcal{L}_k^{\text{clip},\ell}$ is discussed in more detail in Sect. 2.4.3, and $\delta_{\text{drop}}, \delta_{\text{rest}} \ge 0$ are slack parameters: increasing them makes the corresponding operator more likely to be selected, which tends to keep the index set smaller and reduce `MICE`'s overhead. In our experiments we use values in $[0, 1]$.

2.4.1 Dropping Iterations of the `MICE` Index Set

Given our estimator's stochastic nature, at the current iteration k, we may wonder if the iteration $k-1$ should be kept or dropped out from the `MICE` index set since it may not reduce the computational work. The procedure we follow here draws directly from an idea introduced by Giles [39] for the `MLMC` method. Although the numerical approach is the same, we construct the algorithm in a greedy manner. We only check the case of dropping the previous iteration in the current index set. In this approach, we never drop the initial iteration $\min\{\mathcal{L}_k\}$.

2.4.2 Restarting the `MICE` Index Set

As we verified in the previous section on whether we should keep the iteration $\ell = k-1$ in the `MICE` index set, we also may wonder if restarting the estimator may be less expensive than updating it. Usually, in the literature of control variates techniques for stochastic optimization, the restart step is performed after a fixed number of iterations; see, for instance, [23, 36, 37]. Moreover, restarting the index set resets the conditional bias discussed in Remark 3.

2.4.3 Clipping the `MICE` Index Set

In some cases, it may be advantageous to discard only some initial iterates indices out of the index set instead of the whole index set. We refer to this procedure as clipping the index set. We propose two different approaches to decide when and where to clip the index set.

Clipping "A" $\mathcal{L}_k^{\text{clip},\ell^*}$ is as in Definition 2 with

$$\ell^* = \operatorname*{arg\,min}_{\ell\in\mathcal{L}_{k-1}} \Delta\mathcal{W}_k(\mathcal{L}_k^{\text{clip},\ell}). \tag{41}$$

This clipping technique can be applied in both the continuous and discrete cases.
Clipping "B" This technique is simpler but can only be used in the finite sum case. It consists in clipping $\mathcal{L}_{k-1}$ at $\ell^* = \max\{\ell \in \mathcal{L}_{k-1} : M_{\ell,k-1} = N\}$.

Clipping "A" adds an extra computation overhead when calculating $M_{\ell,k}$ for each $\ell \in \mathcal{L}_k$ each iteration k. Thus, in the finite sum case, we suggest using Clipping "B". Clipping shortens the index set, thus possibly reducing the general overhead of `MICE`. Moreover, clipping the index set may reduce the frequency of restarts and the bias of the `MICE` estimator.

3 `SGD-MICE` Convergence and Gradient Sampling Cost Analysis

In this section, we will analyze the convergence of stochastic gradient methods with fixed step size as

$$\boldsymbol{\xi}_{k+1} = \boldsymbol{\xi}_k - \eta \boldsymbol{v}_k, \tag{42}$$

with gradient estimates controlled as

$$\mathbb{E}\left[\left\|\boldsymbol{v}_k - \nabla_{\boldsymbol{\xi}} F(\boldsymbol{\xi}_k)\right\|^2\right] \leq \epsilon^2 \mathbb{E}\left[\left\|\nabla_{\boldsymbol{\xi}} F(\boldsymbol{\xi}_k)\right\|^2\right]. \tag{43}$$

In special, we are interested in `SGD-MICE`, where $\boldsymbol{v}_k = \nabla_{\boldsymbol{\xi}} \mathcal{F}_k$ as defined in (13), and `SGD-A`, where $\boldsymbol{v}_k = M_k^{-1} \sum_{i=1}^{M_k} \nabla_{\boldsymbol{\xi}} f(\boldsymbol{\xi}_k, \boldsymbol{\theta}_i)$. Here, `SGD-A` is SGD where the sample sizes are increased to control the statistical error condition in (43) and can be seen as a special case of `SGD-MICE` where `Restart` is used every iteration. For `MICE`, this condition is satisfied by the choice of the sample sizes in Sect. 2.3.

Let us lay some assumptions.

Assumption 2 (Lipschitz continuous gradient) If the gradient of $F\colon \mathbb{R}^{d_{\boldsymbol{\xi}}} \mapsto \mathbb{R}$ is Lipschitz continuous, then, for some $L > 0$,

$$\left\|\nabla_{\boldsymbol{\xi}} F(\boldsymbol{x}) - \nabla_{\boldsymbol{\xi}} F(\boldsymbol{y})\right\| \leq L \left\|\boldsymbol{y} - \boldsymbol{x}\right\|, \qquad \forall \boldsymbol{x}, \boldsymbol{y} \in \mathbb{R}^{d_{\boldsymbol{\xi}}}. \tag{44}$$

Assumption 3 (Convexity) If F is convex, then,

$$F(\boldsymbol{y}) \geq F(\boldsymbol{x}) + \left\langle \nabla_{\boldsymbol{\xi}} F(\boldsymbol{x}), \boldsymbol{y} - \boldsymbol{x} \right\rangle, \qquad \forall \boldsymbol{x}, \boldsymbol{y} \in \mathbb{R}^{d_{\boldsymbol{\xi}}}. \tag{45}$$

Assumption 4 (Strong convexity) If F is μ-strongly convex, then, for some $\mu > 0$,

$$F(\boldsymbol{y}) \geq F(\boldsymbol{x}) + \langle \nabla_{\boldsymbol{\xi}} F(\boldsymbol{x}), \boldsymbol{y} - \boldsymbol{x} \rangle + \frac{\mu}{2} \left\|\boldsymbol{y} - \boldsymbol{x}\right\|^2, \qquad \forall \boldsymbol{x}, \boldsymbol{y} \in \mathbb{R}^{d_{\boldsymbol{\xi}}}. \tag{46}$$

Assumption 5 (Polyak–Łojasiewicz) If F is gradient dominated, it satisfies the Polyak–Łojasiewicz inequality

$$\frac{1}{2} \left\|\nabla_{\boldsymbol{\xi}} F(\boldsymbol{x})\right\|^2 \geq \mu (F(\boldsymbol{x}) - F^*), \qquad \forall \boldsymbol{x} \in \mathbb{R}^{d_{\boldsymbol{\xi}}}, \tag{47}$$

for a constant $\mu > 0$, where F^* is the minimum value of F.

Assumption 5 is weaker than Assumption 4, holding even for some non-convex problems [43].

3.1 Optimization Convergence Analysis

Proposition 1 (Local convergence in expectation of gradient-controlled SGD on L-smooth problems) *Let $F : \mathbb{R}^{d_\xi} \to \mathbb{R}$ be a differentiable function satisfying Assumption 2 with constant $L > 0$. Then, SGD methods with relative gradient error control $\epsilon < 1$ in the L^2-norm sense and step-size $\eta = 1/L$ reduce the optimality gap in expectation as*

$$\mathbb{E}\left[F(\boldsymbol{\xi}_{k+1})\right] \leq \mathbb{E}\left[F(\boldsymbol{\xi}_k)\right] - \left(\frac{1-\epsilon^2}{2L}\right)\mathbb{E}\left[\left\|\nabla_{\boldsymbol{\xi}} F(\boldsymbol{\xi}_k)\right\|^2\right]. \tag{48}$$

Proof Let $\boldsymbol{e}_k \coloneqq \boldsymbol{v}_k - \nabla_{\boldsymbol{\xi}} F(\boldsymbol{\xi}_k)$. From L-smoothness,

$$F(\boldsymbol{\xi}_{k+1}) \leq F(\boldsymbol{\xi}_k) - \eta\left\langle \nabla_{\boldsymbol{\xi}} F(\boldsymbol{\xi}_k), \nabla_{\boldsymbol{\xi}} F(\boldsymbol{\xi}_k) + \boldsymbol{e}_k\right\rangle + \frac{L\eta^2}{2}\left\|\nabla_{\boldsymbol{\xi}} F(\boldsymbol{\xi}_k) + \boldsymbol{e}_k\right\|^2 \tag{49}$$

$$= F(\boldsymbol{\xi}_k) + \left(\frac{L\eta^2}{2} - \eta\right)\left\|\nabla_{\boldsymbol{\xi}} F(\boldsymbol{\xi}_k)\right\|^2 + (L\eta^2 - \eta)\left\langle \nabla_{\boldsymbol{\xi}} F(\boldsymbol{\xi}_k), \boldsymbol{e}_k\right\rangle + \frac{L\eta^2}{2}\left\|\boldsymbol{e}_k\right\|^2. \tag{50}$$

Taking expectation on both sides and then using the Cauchy–Schwarz inequality,

$$\mathbb{E}\left[F(\boldsymbol{\xi}_{k+1})\right] \leq \mathbb{E}\left[F(\boldsymbol{\xi}_k)\right] + \left(\frac{L\eta^2}{2} - \eta\right)\mathbb{E}\left[\left\|\nabla_{\boldsymbol{\xi}} F(\boldsymbol{\xi}_k)\right\|^2\right] + |L\eta^2 - \eta|\sqrt{\mathbb{E}\left[\left\|\nabla_{\boldsymbol{\xi}} F(\boldsymbol{\xi}_k)\right\|^2\right]\mathbb{E}\left[\left\|\boldsymbol{e}_k\right\|^2\right]} + \frac{L\eta^2}{2}\mathbb{E}\left[\left\|\boldsymbol{e}_k\right\|^2\right]. \tag{51}$$

$$\leq \mathbb{E}\left[F(\boldsymbol{\xi}_k)\right] + \left(\frac{L\eta^2}{2} - \eta + \epsilon|L\eta^2 - \eta| + \epsilon^2\frac{L\eta^2}{2}\right)\mathbb{E}\left[\left\|\nabla_{\boldsymbol{\xi}} F(\boldsymbol{\xi}_k)\right\|^2\right], \tag{52}$$

where (43) is used to get the last inequality. Here, the step size that minimizes the term inside the parenthesis is $\eta = 1/L$. Substituting the step size in the equation above and taking full expectation on both sides concludes the proof. □

If the function F is also unimodal, as in the case of F satisfying Assumptions 3 or 5, then the convergence presented in Proposition 1 is also global, i.e., $\mathbb{E}\left[F(\boldsymbol{\xi}_{k+1}) - F(\boldsymbol{\xi}^*)\right] \to 0$.

Proposition 2 (Global convergence of gradient-controlled SGD in gradient-dominated problems) *Let all Assumptions of Proposition 1 be satisfied. Moreover, let F satisfy*

Assumption 5 with constant $\mu > 0$. Then, gradient-controlled SGD with step-size $\eta = 1/L$ converges linearly,

$$\mathbb{E}\left[F(\boldsymbol{\xi}_{k+1}) - F(\boldsymbol{\xi}^*)\right] \leq \left(1 - (1-\epsilon^2)\frac{\mu}{L}\right)^{k+1} \mathbb{E}\left[F(\boldsymbol{\xi}_0) - F(\boldsymbol{\xi}^*)\right]. \tag{53}$$

Proof From (48), using Assumption 5 and unrolling the recursion concludes the proof. □

Corollary 1 *If conditions of Proposition 2 hold and F also satisfies Assumption 3, the squared L^2-norm of the gradient of the objective function is bounded as*

$$\mathbb{E}\left[\left\|\nabla_{\boldsymbol{\xi}} F(\boldsymbol{\xi}_{k+1})\right\|^2\right] \leq 2L\left(1 - (1-\epsilon^2)\frac{\mu}{L}\right)^{k+1} \mathbb{E}\left[F(\boldsymbol{\xi}_0) - F(\boldsymbol{\xi}^*)\right]. \tag{54}$$

Proof From [44, Theorem 2.1.5], if F is convex and L-smooth,

$$\left\|\nabla_{\boldsymbol{\xi}} F(\boldsymbol{\xi})\right\|^2 \leq 2L(F(\boldsymbol{\xi}) - F(\boldsymbol{\xi}^*)). \tag{55}$$

Substituting this inequality for $\boldsymbol{\xi}_{k+1}$ into (53) finishes the proof. □

Corollary 2 *If conditions of Proposition 2 are satisfied and F also satisfies Assumption 4,*

$$\mathbb{E}\left[\left\|\boldsymbol{\xi}_{k+1} - \boldsymbol{\xi}^*\right\|^2\right] \leq \frac{2}{\mu}\left(1 - (1-\epsilon^2)\frac{\mu}{L}\right)^{k+1} \mathbb{E}\left[F(\boldsymbol{\xi}_0) - F(\boldsymbol{\xi}^*)\right]. \tag{56}$$

Proof From the definition of strong-convexity in Assumption 4,

$$\left\|\boldsymbol{\xi} - \boldsymbol{\xi}^*\right\|^2 \leq \frac{2}{\mu}(F(\boldsymbol{\xi}) - F(\boldsymbol{\xi}^*)). \tag{57}$$

Substituting into (53) finishes the proof. □

3.1.1 High-Probability Convergence in the PL Regime

We complement the expectation-based results above with a high-probability statement. The proof separates into two steps: (*i*) show linear convergence on any event where the relative error holds uniformly in k and (*ii*) show that, under a simple tail assumption (Assumption 6), such an event has probability at least $1 - \delta$ in an add-only MICE regime (Appendix B, C, D).

Assumption 6 (Coordinatewise sub-Gaussian level increments) In an add-only regime, assume that each level $\ell \in \mathbb{N}$ is anchored to a fixed iterate pair between restarts and its contribution to $\nabla\mathcal{F}_k$ is a sample mean of i.i.d. centered increments $Z_{\ell,1}, Z_{\ell,2}, \ldots \in \mathbb{R}^{d_\xi}$ with $\mathbb{E}[Z_{\ell,i}] = 0$. Assume that for each coordinate $j \in \{1, \ldots, d_\xi\}$ there exists $\sigma_\ell^2 \geq 0$ such that for all $\lambda \in \mathbb{R}$,

$$\mathbb{E}\left[\exp\left(\lambda(Z_{\ell,i})_j\right)\right] \leq \exp\left(\frac{\lambda^2\sigma_\ell^2}{2}\right). \tag{58}$$

Theorem 1 (Linear convergence on a uniform relative error event) *Assume F satisfies Assumptions 2 and 5 with constants $L > 0$ and $\mu > 0$. Fix $\eta > 0$ and $\epsilon \in [0, 1)$. Define*

$$c(\eta, \epsilon) := \eta(1-\epsilon) - \frac{L\eta^2}{2}(1+\epsilon)^2. \tag{59}$$

Assume $c(\eta, \epsilon) > 0$ and $2\mu\, c(\eta, \epsilon) < 1$. Let Ω be any event on which the realized gradient estimation errors satisfy

$$\left\| \boldsymbol{v}_k - \nabla_{\boldsymbol{\xi}} F(\boldsymbol{\xi}_k) \right\| \le \epsilon \left\| \nabla_{\boldsymbol{\xi}} F(\boldsymbol{\xi}_k) \right\|, \qquad \forall\, k \ge 0. \tag{60}$$

Then, on Ω, the iterates of SGD with update $\boldsymbol{\xi}_{k+1} = \boldsymbol{\xi}_k - \eta \boldsymbol{v}_k$ satisfy, for all $k \ge 0$,

$$F(\boldsymbol{\xi}_k) - F(\boldsymbol{\xi}^*) \le r_{\mathrm{hp}}^k \big(F(\boldsymbol{\xi}_0) - F(\boldsymbol{\xi}^*) \big), \qquad r_{\mathrm{hp}} := 1 - 2\mu\, c(\eta, \epsilon) \in (0, 1). \tag{61}$$

Proof Work on Ω and fix $k \ge 0$. From L-smoothness and the update $\boldsymbol{\xi}_{k+1} = \boldsymbol{\xi}_k - \eta(\nabla_{\boldsymbol{\xi}} F(\boldsymbol{\xi}_k) + (\boldsymbol{v}_k - \nabla_{\boldsymbol{\xi}} F(\boldsymbol{\xi}_k)))$, we have

$$\begin{aligned} F(\boldsymbol{\xi}_{k+1}) &\le F(\boldsymbol{\xi}_k) - \eta \langle \nabla_{\boldsymbol{\xi}} F(\boldsymbol{\xi}_k), \boldsymbol{v}_k \rangle + \frac{L\eta^2}{2} \|\boldsymbol{v}_k\|^2 \\ &\le F(\boldsymbol{\xi}_k) - \eta \left\| \nabla_{\boldsymbol{\xi}} F(\boldsymbol{\xi}_k) \right\|^2 + \eta \left\| \nabla_{\boldsymbol{\xi}} F(\boldsymbol{\xi}_k) \right\| \left\| \boldsymbol{v}_k - \nabla_{\boldsymbol{\xi}} F(\boldsymbol{\xi}_k) \right\| \\ &\quad + \frac{L\eta^2}{2} \big(\left\| \nabla_{\boldsymbol{\xi}} F(\boldsymbol{\xi}_k) \right\| \\ &\quad + \left\| \boldsymbol{v}_k - \nabla_{\boldsymbol{\xi}} F(\boldsymbol{\xi}_k) \right\| \big)^2 \\ &\le F(\boldsymbol{\xi}_k) - c(\eta, \epsilon) \left\| \nabla_{\boldsymbol{\xi}} F(\boldsymbol{\xi}_k) \right\|^2. \end{aligned} \tag{62}$$

where in the last step we used (60). Applying the PL inequality and unrolling the recursion yields (61). □

Corollary 3 (High-probability linear convergence for add-only MICE) *Assume the conditions of Theorem 1. Consider SGD-MICE in an add-only regime and assume Assumption 6. Assume further that, conditional on the iterate history, the sample sets used at distinct levels are independent, and that the sample sizes $\{M_{\ell,k}\}_{\ell \in \mathcal{L}_k}$ are chosen before drawing the iteration-k samples. Fix a summable schedule $(\delta_k)_{k \ge 0}$ with $\sum_{k \ge 0} \delta_k \le \delta$. If the (predictable) sample sizes satisfy the scalar constraint*

$$\sum_{\ell \in \mathcal{L}_k} \frac{\sigma_\ell^2}{M_{\ell,k}} \le \frac{\epsilon^2}{2 d_\xi \, \log(2 d_\xi / \delta_k)} \left\| \nabla_{\boldsymbol{\xi}} F(\boldsymbol{\xi}_k) \right\|^2, \qquad \forall\, k \ge 0, \tag{63}$$

then, with probability at least $1 - \delta$, the bound (61) holds for all $k \ge 0$.

Proof Define the gradient estimation error $e_k := \boldsymbol{v}_k - \nabla F(\boldsymbol{\xi}_k)$. Under Assumption 6, the conditional independence across levels, and the predictability of the sample

sizes, Lemma 3 (in Appendix 1Appendix B, C, D) yields an event $\Omega_\delta^{\mathrm{MICE}}$ such that $\mathbb{P}[\Omega_\delta^{\mathrm{MICE}}] \geq 1-\delta$ and, on $\Omega_\delta^{\mathrm{MICE}}$, for all $k \geq 0$,

$$\|e_k\| \leq \sqrt{2d_\xi \log\Big(\frac{2d_\xi}{\delta_k}\Big)} \left(\sum_{\ell \in \mathcal{L}_k} \frac{\sigma_\ell^2}{M_{\ell,k}}\right)^{1/2}. \tag{64}$$

(The summability $\sum_k \delta_k \leq \delta$ is used in Lemma 3 via a union bound over k.)

Assume now that the sample sizes satisfy (63). Substituting this bound into (64) gives, on $\Omega_\delta^{\mathrm{MICE}}$, for all $k \geq 0$,

$$\|e_k\| \leq \epsilon \left\|\nabla F(\boldsymbol{\xi}_k)\right\|. \tag{65}$$

Hence the uniform relative-error condition (60) of Theorem 1 holds on $\Omega_\delta^{\mathrm{MICE}}$. Applying Theorem 1 concludes that (61) holds for all $k \geq 0$ with probability at least $1-\delta$. □

The high-probability result above is intentionally stated for the add-only regime and for sample sizes $M_{\ell,k}$ chosen predictably with respect to the past. Extending it to fully adaptive index-set operations (Drop/Restart/Clip) and within-iteration stopping rules is left for future work.

Having established convergence rates for gradient-controlled SGD methods, we now quantify the total number of gradient evaluations required to achieve a given tolerance, comparing `SGD-MICE` with adaptive batch-size approaches.

3.2 Gradient Sampling Cost Analysis

To analyze the gradient sampling cost, we focus on the analysis in expectation with L^2 control on the error. A discussion on cost in high probability is presented after Corollary 5 in Remark 6.

Assuming the assumptions of Proposition 2 hold, the optimality gap converges with rate $r := 1-(1-\epsilon^2)\mu/L$. Then, we have the following inequalities that will be used throughout this section,

$$\frac{1}{\log(r)} \leq \frac{1}{1-r} = \frac{\kappa}{1-\epsilon^2}, \tag{66}$$

where $\kappa = L/\mu$. Moreover,

$$\frac{1}{1-\sqrt{r}} \leq \frac{2\kappa}{1-\epsilon^2}. \tag{67}$$

For the sake of simplicity and given the cumulative nature of the computational gradient sampling cost in `MICE`, we analyze the total gradient sampling cost on a set of iterations $\{\boldsymbol{\xi}_\ell\}_{\ell=0}^{k^*}$ converging to $\boldsymbol{\xi}^*$ as per Proposition 2. Observe that in this

simplified setting, the number of iterations required to stop the iteration, $k^* = k^*(tol)$, and both the sequences $(\boldsymbol{\xi}_\ell)$ and $(M_{\ell,k})$ are still random. Indeed, we define

$$k^* = \min\{k \geq 0 \colon \left\|\nabla_{\boldsymbol{\xi}} F(\boldsymbol{\xi}_k)\right\|^2 \leq tol\}. \tag{68}$$

Corollary 4 (Number of iterations) *If the assumptions of Corollary 1 hold then, letting*

$$k_1 := \frac{\log(tol^{-1} 2L\mathbb{E}\left[F(\boldsymbol{\xi}_0) - F(\boldsymbol{\xi}^*)\right])}{\log(1/r)}, \tag{69}$$

we have

$$\mathbb{P}\left[k^* \geq k\right] \leq \begin{cases} 1, \text{ if } k < k_1 \\ r^{k-k_1} \text{ otherwise.} \end{cases} \tag{70}$$

Moreover, we have

$$\mathbb{E}\left[k^*\right] \leq \frac{1}{1-r} + \max\left\{0, \frac{\log(tol^{-1} 2L\mathbb{E}\left[F(\boldsymbol{\xi}_0) - F(\boldsymbol{\xi}^*)\right])}{\log(1/r)}\right\}. \tag{71}$$

Proof First observe that

$$\mathbb{P}\left[k^* \geq k\right] \leq \mathbb{P}\left[\left\|\nabla_{\boldsymbol{\xi}} F(\boldsymbol{\xi}_k)\right\|^2 \geq tol\right]. \tag{72}$$

Then apply Markov's inequality and the exponential convergence in L^2-norm presented in Corollary 1, yielding

$$\mathbb{P}\left[k^* \geq k\right] \leq \min\left\{1, tol^{-1} 2Lr^k \mathbb{E}\left[F(\boldsymbol{\xi}_0) - F(\boldsymbol{\xi}^*)\right]\right\}. \tag{73}$$

The result (70) follows then directly. To show (71), simply use (73) and that

$$\mathbb{E}\left[k^*\right] = \sum_{k \geq 0} \mathbb{P}\left[k^* \geq k\right] \leq \max\{0, k_1\} + \frac{1}{1-r}. \tag{74}$$

□

The expected value of k^* can be bounded using (66) as

$$\mathbb{E}\left[k^*\right] \leq \max\left\{0, \frac{\kappa}{1-\epsilon^2} \log(tol^{-1} 2L\mathbb{E}\left[F(\boldsymbol{\xi}_0) - F(\boldsymbol{\xi}^*)\right])\right\} + \frac{\kappa}{1-\epsilon^2}. \tag{75}$$

Assumption 7 (Bound on second moments of gradient differences)

$$\mathbb{E}\left[\left\|\nabla_{\boldsymbol{\xi}} f(\boldsymbol{x}, \boldsymbol{\theta}) - \nabla_{\boldsymbol{\xi}} f(\boldsymbol{y}, \boldsymbol{\theta})\right\|^2 \,\middle|\, \boldsymbol{x}, \boldsymbol{y}\right] \leq \sigma^2 \left\|\nabla_{\boldsymbol{\xi}} F(\boldsymbol{x}) - \nabla_{\boldsymbol{\xi}} F(\boldsymbol{y})\right\|^2. \tag{76}$$

Assumption 7 is a convenient way to control the second moments of gradient differences by differences of true gradients, and it holds for many smooth models with light-tailed gradient noise. However, in heavy-tailed regimes (or when gradients are contaminated by occasional large outliers), second moments may be very large or even infinite, so Assumption 7 may fail or yield overly conservative bounds. In such cases, a practical safeguard is to clip large gradients or gradient differences (not to be confused with our `Clip` operator), which limits the influence of outliers at the cost of introducing a controlled bias.

If f satisfies Assumption 7 for $\ell > 0$,

$$V_{\ell,k} \leq \mathbb{E}\left[\left\| \nabla_{\boldsymbol{\xi}} f(\boldsymbol{\xi}_\ell, \boldsymbol{\theta}) - \nabla_{\boldsymbol{\xi}} f(\boldsymbol{\xi}_{p_k(\ell)}, \boldsymbol{\theta}) \right\|^2 \,\middle|\, \boldsymbol{\xi}_\ell, \boldsymbol{\xi}_{p_k(\ell)} \right] \tag{77}$$

$$\leq \sigma^2 \left\| \nabla_{\boldsymbol{\xi}} F(\boldsymbol{\xi}_\ell) - \nabla_{\boldsymbol{\xi}} F(\boldsymbol{\xi}_{p_k(\ell)}) \right\|^2. \tag{78}$$

For $\ell = 0$,

$$\sqrt{V_{0,k}} \leq \sqrt{\mathbb{E}\left[\left\| \nabla_{\boldsymbol{\xi}} f(\boldsymbol{\xi}_0, \boldsymbol{\theta}) \right\|^2 \,\middle|\, \boldsymbol{\xi}_0 \right]} \tag{79}$$

$$= \sqrt{\mathbb{E}\left[\left\| \nabla_{\boldsymbol{\xi}} f(\boldsymbol{\xi}_0, \boldsymbol{\theta}) - \nabla_{\boldsymbol{\xi}} f(\boldsymbol{\xi}^*, \boldsymbol{\theta}) + \nabla_{\boldsymbol{\xi}} f(\boldsymbol{\xi}^*, \boldsymbol{\theta}) \right\|^2 \,\middle|\, \boldsymbol{\xi}_0 \right]} \tag{80}$$

$$\leq \sqrt{\mathbb{E}\left[\left\| \nabla_{\boldsymbol{\xi}} f(\boldsymbol{\xi}_0, \boldsymbol{\theta}) - \nabla_{\boldsymbol{\xi}} f(\boldsymbol{\xi}^*, \boldsymbol{\theta}) \right\|^2 \,\middle|\, \boldsymbol{\xi}_0 \right]} + \underbrace{\sqrt{\mathbb{E}\left[\left\| \nabla_{\boldsymbol{\xi}} f(\boldsymbol{\xi}^*, \boldsymbol{\theta}) \right\|^2 \right]}}_{\sqrt{V_*}} \tag{81}$$

$$\leq \sigma \left\| \nabla_{\boldsymbol{\xi}} F(\boldsymbol{\xi}_0) \right\| + \sqrt{V_*}. \tag{82}$$

Let the total gradient sampling cost to reach iteration $k' + 1$ be

$$\mathcal{C}_{k'} = \sum_{k=0}^{k'} \Delta\mathcal{C}_k(\mathcal{L}_k), \tag{83}$$

where $\Delta\mathcal{C}$ is defined as in (37). In this section, we present limited analyses of `SGD-MICE` to reach k^* where we assume only the `Add` operator is used, thus, using the equation above,

$$\mathcal{C}_{k^*-1} = \sum_{k=0}^{k^*-1} \sum_{\ell=0}^{k} (1 + \mathbb{1}_{\underline{\mathcal{L}_k}}(\ell))(M_{\ell,k} - M_{\ell,k-1}) \tag{84}$$

$$= \sum_{\ell=0}^{k^*-1} (1 + \mathbb{1}_{\underline{\mathcal{L}_{k^*-1}}}(\ell)) M_{\ell,k^*-1}. \tag{85}$$

As will be shown in Sect. 5, the other index set operators, `Drop`, `Restart`, and `Clip` greatly improve the convergence of `SGD-MICE`. As a consequence, these analyses considering only the `Add` operator are pessimistic.

3.2.1 Expectation Minimization Problems

Corollary 5 (Expected gradient sampling cost of SGD-MICE with linear convergence) *Let the Assumptions of Corollary 1 and Assumption 7 hold. Moreover, let k^* be the smallest k such that $\left\|\nabla_{\boldsymbol{\xi}} F(\boldsymbol{\xi}_k)\right\|^2 < tol$ and all sample sizes at the last iteration be larger than M_{min}. Then, the expected number of gradient evaluations needed to generate $\boldsymbol{\xi}_{k^*}$ is*

$$\mathbb{E}\left[\mathcal{C}_{k^*-1}\right] \leq \epsilon^{-2} tol^{-1} \left(4\sigma\sqrt{L\mathbb{E}\left[F(\boldsymbol{\xi}_0) - F(\boldsymbol{\xi}^*)\right]}\left(\frac{2\kappa}{1-\epsilon^2}\right) + \sqrt{V_*}\right)^2 + 2M_{min}\left(\max\left\{0, \frac{\kappa}{1-\epsilon^2}\log(tol^{-1} 2\, L\mathbb{E}\left[F(\boldsymbol{\xi}_0) - F(\boldsymbol{\xi}^*)\right])\right\} + \frac{\kappa}{1-\epsilon^2}\right). \quad (86)$$

Moreover, the relative gradient error that minimizes the expected gradient sampling cost is $\epsilon = \sqrt{1/3}$.

Proof We know that $k^* - 1$ iterations are needed to generate $\boldsymbol{\xi}_{k^*}$. Thus, the whole optimization cost is

$$\mathcal{C}_{k^*-1} \leq \epsilon^{-2}\left\|\nabla_{\boldsymbol{\xi}} F(\boldsymbol{\xi}_{k^*-1})\right\|^{-2}\left(\sum_{\ell'\in\mathcal{L}_{k^*-1}}\sqrt{V_{\ell',k^*-1}(1+\mathbb{1}_{\underline{\mathcal{L}_{k^*-1}}}(\ell'))}\right)^2 + \sum_{\ell'\in\mathcal{L}_{k^*-1}}(1+\mathbb{1}_{\underline{\mathcal{L}_{k^*-1}}}(\ell'))M_{min} \quad (87)$$

$$\leq \epsilon^{-2} tol^{-1}\left(\sum_{\ell'\in\mathcal{L}_{k^*-1}}\sqrt{V_{\ell',k^*-1}(1+\mathbb{1}_{\underline{\mathcal{L}_{k^*-1}}}(\ell'))}\right)^2 + 2|\mathcal{L}_{k^*-1}|M_{min}. \quad (88)$$

Let us analyze the following sum

$$\sum_{\ell'\in\mathcal{L}_{k^*-1}}\sqrt{V_{\ell',k^*-1}(1+\mathbb{1}_{\underline{\mathcal{L}_{k^*-1}}}(\ell'))} = \sqrt{V_{0,k}} + \sqrt{2}\sum_{1\leq\ell'\leq k^*-1}\sqrt{V_{\ell',k}} \quad (89)$$

$$\leq \sigma\left\|\nabla_{\boldsymbol{\xi}} F(\boldsymbol{\xi}_0)\right\| + \sqrt{V_*} + \sqrt{2}\sigma\sum_{1\leq\ell'\leq k^*-1}\left\|\nabla_{\boldsymbol{\xi}} F(\boldsymbol{\xi}_{\ell'}) - \nabla_{\boldsymbol{\xi}} F(\boldsymbol{\xi}_{p_{k^*-1}(\ell')})\right\| \quad (90)$$

$$\leq \sigma\left\|\nabla_{\boldsymbol{\xi}} F(\boldsymbol{\xi}_0)\right\| + \sqrt{V_*} + \sqrt{2}\sigma\sum_{1\leq\ell'\leq k^*-1}\left\|\nabla_{\boldsymbol{\xi}} F(\boldsymbol{\xi}_{\ell'})\right\| + \left\|\nabla_{\boldsymbol{\xi}} F(\boldsymbol{\xi}_{p_{k^*-1}(\ell')})\right\| \quad (91)$$

$$\leq 2\sqrt{2}\sigma\sum_{0\leq\ell'\leq k^*-1}\left\|\nabla_{\boldsymbol{\xi}} F(\boldsymbol{\xi}_{\ell'})\right\| + \sqrt{V_*}. \quad (92)$$

Taking expectation of the summation above squared,

$$\mathbb{E}\left[\left(\sum_{\ell'\in\mathcal{L}_{k^*-1}}\sqrt{V_{\ell',k^*-1}(1+\mathbb{1}_{\underline{\mathcal{L}_{k^*-1}}}(\ell'))}\right)^2\right]$$

$$\leq 8\,\sigma^2\sum_{\ell'\in\mathcal{L}_{k^*-1}}\sum_{\ell\in\mathcal{L}_{k^*-1}}\mathbb{E}\left[\left\|\nabla_{\boldsymbol{\xi}}F(\boldsymbol{\xi}_{\ell'})\right\|\left\|\nabla_{\boldsymbol{\xi}}F(\boldsymbol{\xi}_{\ell})\right\|\right] \tag{93}$$

$$+4\sigma\sqrt{2V_*}\sum_{\ell''\in\mathcal{L}_{k^*-1}}\mathbb{E}\left[\left\|\nabla_{\boldsymbol{\xi}}F(\boldsymbol{\xi}_{\ell''})\right\|\right]+V_*$$

$$\leq 8\,\sigma^2\sum_{\ell'\in\mathcal{L}_{k^*-1}}\sum_{\ell\in\mathcal{L}_{k^*-1}}\sqrt{\mathbb{E}\left[\left\|\nabla_{\boldsymbol{\xi}}F(\boldsymbol{\xi}_{\ell'})\right\|^2\right]\mathbb{E}\left[\left\|\nabla_{\boldsymbol{\xi}}F(\boldsymbol{\xi}_{\ell})\right\|^2\right]} \tag{94}$$

$$+4\sigma\sqrt{2V_*}\sum_{\ell''\in\mathcal{L}_{k^*-1}}\sqrt{\mathbb{E}\left[\left\|\nabla_{\boldsymbol{\xi}}F(\boldsymbol{\xi}_{\ell''})\right\|^2\right]}+V_*$$

$$=\left(2\sqrt{2}\sigma\sum_{\ell\in\mathcal{L}_{k^*-1}}\sqrt{\mathbb{E}\left[\left\|\nabla_{\boldsymbol{\xi}}F(\boldsymbol{\xi}_{\ell})\right\|^2\right]}+\sqrt{V_*}\right)^2 \tag{95}$$

$$\leq\left(4\sigma\sqrt{L\mathbb{E}\left[F(\boldsymbol{\xi}_0)-F(\boldsymbol{\xi}^*)\right]}\left(\sum_{\ell\in\mathcal{L}_{k^*-1}}r^{\ell/2}\right)+\sqrt{V_*}\right)^2 \tag{96}$$

$$\leq\left(4\sigma\sqrt{L\mathbb{E}\left[F(\boldsymbol{\xi}_0)-F(\boldsymbol{\xi}^*)\right]}\left(\frac{1}{1-\sqrt{r}}\right)+\sqrt{V_*}\right)^2. \tag{97}$$

Substituting back to the expected cost,

$$\mathbb{E}\left[\mathcal{C}_{k^*-1}\right]\leq\epsilon^{-2}tol^{-1}\left(4\sigma\sqrt{L\mathbb{E}\left[F(\boldsymbol{\xi}_0)-F(\boldsymbol{\xi}^*)\right]}\left(\frac{1}{1-\sqrt{r}}\right)+\sqrt{V_*}\right)^2+2\mathbb{E}\left[k^*\right]M_{min}. \tag{98}$$

Substituting the expected number of iterations from Corollary 4 and using (67) results in (86).

Since the term $(1/(1-\sqrt{r}))^2$ is $\mathcal{O}\left((1-\epsilon^2)^{-2}\kappa^2\right)$, it dominates convergence as $\kappa\to\infty$, thus the expected work of SGD-MICE without restart or dropping is $\mathcal{O}\left(\epsilon^{-2}(1-\epsilon^2)^{-2}\kappa^2 tol^{-1}\right)$. Therefore, the relative gradient error that minimizes the total gradient sampling cost is $\epsilon=\sqrt{1/3}$. □

Remark 6 (*Expected vs. high-probability sampling-cost bounds*) The results in Sect. 3.2 bound the gradient sampling cost in *expectation*, under the mean-square relative error control (43). In contrast, the high-probability PL result in Sect. 3.1.1 requires the stronger condition that the realized error satisfies $\|e_k\|\leq\epsilon\|\nabla F(\boldsymbol{\xi}_k)\|$ *uniformly in k* with probability at least $1-\delta$. Under Assumption 6 and the conditional independence structure in Appendix B, C, D, this uniform control is obtained by imposing the

variance-sum constraint (63), which inflates the required sample sizes by an explicit factor $\log(2d_\xi/\delta_k)$. With a standard summable schedule, for example $\delta_k \propto \delta/(k+1)^2$, one has $\log(2d_\xi/\delta_k) = \log(2d_\xi/\delta) + \mathcal{O}(\log(k+1))$. Since $k^*(tol) = \mathcal{O}(\log(1/tol))$ in the PL regime, the corresponding high-probability sampling-cost bounds preserve the leading tol^{-1} scaling, at the cost of logarithmic dependence on δ (and at most a mild $\log\log(1/tol)$ correction through the schedule). Concretely, Appendix B, C, D builds a single "good" event $\Omega_\delta^{\mathrm{MICE}}$ via a union bound over iterations (Lemma 3), shows that the relative error control holds uniformly on that event (Corollary 11), and then derives explicit bounds on the iteration complexity (Lemma 4) and on the total gradient sampling cost (Corollary 12).

Corollary 6 (Expected gradient sampling cost of SGD-A) *If Assumptions of Corollary 1 hold and Assumption 7 also holds, SGD-A generates an iterate $\boldsymbol{\xi}_{k^*}$ satisfying $\left\|\nabla_{\boldsymbol{\xi}} F(\boldsymbol{\xi}_{k^*})\right\|^2 \le tol$ with an expected gradient sampling cost*

$$\mathbb{E}\left[\mathcal{C}_{k^*-1}\right] \le \left(\frac{3(\sigma^2+1)}{\epsilon^2} + \frac{2V_*}{\epsilon^2 tol} + M_{min}\right) \times \left(\max\left\{0, \frac{\kappa}{1-\epsilon^2}\log\left(tol^{-1}2L\mathbb{E}\left[F(\boldsymbol{\xi}_0) - F(\boldsymbol{\xi}^*)\right]\right)\right\} + \frac{\kappa}{1-\epsilon^2}\right). \tag{99}$$

Proof Let the gradient sampling cost of SGD-A be

$$\mathcal{C}_{k^*-1} = \sum_{k=0}^{k^*-1} M_{k,k}. \tag{100}$$

The sample sizes are

$$M_{k,k} \le \frac{V_{k,k}}{\epsilon^2 \left\|\nabla_{\boldsymbol{\xi}} F(\boldsymbol{\xi}_k)\right\|^2} + M_{min} \tag{101}$$

We can bound $V_{k,k}$ as

$$V_\ell = \mathbb{E}\left[\left\|\nabla_{\boldsymbol{\xi}} f(\boldsymbol{\xi}_\ell, \boldsymbol{\theta}) - \nabla_{\boldsymbol{\xi}} F(\boldsymbol{\xi}_\ell)\right\|^2 \,\middle|\, \boldsymbol{\xi}_\ell\right] \tag{102}$$

$$\le 2\mathbb{E}\left[\left\|\nabla_{\boldsymbol{\xi}} f(\boldsymbol{\xi}_\ell, \boldsymbol{\theta}) - \nabla_{\boldsymbol{\xi}} f(\boldsymbol{\xi}^*, \boldsymbol{\theta})\right\|^2 \,\middle|\, \boldsymbol{\xi}_\ell\right] + 2\mathbb{E}\left[\left\|\nabla_{\boldsymbol{\xi}} f(\boldsymbol{\xi}^*, \boldsymbol{\theta}) - \nabla_{\boldsymbol{\xi}} F(\boldsymbol{\xi}^*)\right\|^2 \,\middle|\, \boldsymbol{\xi}_\ell\right] \tag{103}$$

$$\le 2\sigma^2 \left\|\nabla_{\boldsymbol{\xi}} F(\boldsymbol{\xi}_\ell)\right\|^2 + 2V^*. \tag{104}$$

$$\mathcal{C}_{k^*-1} \leq \sum_{k=0}^{k^*-1} \left(\frac{V_{k,k}}{\epsilon^2 \left\| \nabla_{\boldsymbol{\xi}} F(\boldsymbol{\xi}_k) \right\|^2} + M_{min} \right) \tag{105}$$

$$\leq \sum_{k=0}^{k^*-1} \left(\frac{2\sigma^2}{\epsilon^2} + \frac{2V^*}{\epsilon^2 \left\| \nabla_{\boldsymbol{\xi}} F(\boldsymbol{\xi}_k) \right\|^2} + M_{min} \right) \tag{106}$$

$$\leq \sum_{k=0}^{k^*-1} \left(\frac{2\sigma^2}{\epsilon^2} + \frac{2V^*}{\epsilon^2 tol} + M_{min} \right) \tag{107}$$

$$\leq \left(\frac{2\sigma^2}{\epsilon^2} + \frac{2V^*}{\epsilon^2 tol} + M_{min} \right) k^*. \tag{108}$$

Taking expectation and substituting $\mathbb{E}\left[k^*\right]$ from Corollary 4 finishes the proof. □

Table 1 presents, for different methods, the cost to reach a desired tolerance in expectation minimization.

Remark 7 (*Stopping criterion*) In practice, applying the stopping criterion (68) requires an approximation of the mean gradient norm at each iteration. A natural approach is to use the `MICE` estimator as such an approximation, yielding

$$\left\| \nabla_{\boldsymbol{\xi}} \mathcal{F}_{k^*} \right\|^2 < tol, \tag{109}$$

provided that the error in the mean gradient is controlled in a relative sense. This quality assurance requires a certain number of gradient samples. For example, let us consider the ideal case of stopping when we start inside the stopping region, near the optimal point $\boldsymbol{\xi}^*$. To this end, suppose that the initial iteration point, $\boldsymbol{\xi}_0$, is such that $\left\| \nabla_{\boldsymbol{\xi}} F(\boldsymbol{\xi}_0) \right\|^2 \leq tol$. What is the cost needed to stop by sampling gradients at $\boldsymbol{\xi}_0$ without iterating at all? Observing that we need a tolerance tol, we thus need a number of samples M that satisfies

$$\frac{\mathbb{E}\left[\left\| \nabla_{\boldsymbol{\xi}} f(\boldsymbol{\xi}_0, \theta) \right\|^2 \right]}{tol} \leq M. \tag{110}$$

Compare the last estimate with (86) and (99).

3.2.2 Finite Sum Minimization Problems

Corollary 7 (Cost analysis of `SGD-MICE` on the finite sum case) *If Assumptions of Corollary 2 hold,* `SGD-MICE` *achieves a stopping criterion with expected gradient*

Table 1 Summary of the main gradient sampling cost bounds in expectation minimization

Method	Gradient estimator / mechanism	Assumptions (expectation case)	Expected gradient sampling cost to reach $\left\|\nabla_{\boldsymbol{\xi}} F(\boldsymbol{\xi}_{k^*})\right\|^2 \leq tol$
vanilla `SGD`	Fixed-batch Monte Carlo gradients	Standard (see [18])	Standard sublinear baseline
`SGD-A`	Adaptive batch sizes to enforce relative error	Cor. 1 + Ass. 7	$\mathcal{O}\left(\kappa\, tol^{-1} \log(tol^{-1})\right)$
vanilla `SGD-MICE`	`MICE` with `Add` only (no operators)	Cor. 1 + Ass. 7	$\mathcal{O}\left(\kappa^2\, tol^{-1}\right)$
`SGD-MICE` + operators	`MICE` with `Add/Drop/Restart/Clip`	Empirical improvement (see Sect. 5)	$\mathcal{O}\left(\kappa\, tol^{-1}\right)^a$

The bounds depend on the relative error parameter ϵ; we report the leading-order scaling in κ and tol (see Corollaries 5 and 6 for full expressions)

[a] Empirically observed.

sampling cost

$$\mathbb{E}\left[\mathcal{C}_{k^*-1} \mid \boldsymbol{\xi}_0\right] \leq \frac{(N-1)\left(8\frac{\kappa}{1-\epsilon^2}\sigma\sqrt{L(F(\boldsymbol{\xi}_0)-F(\boldsymbol{\xi}^*))}+\sqrt{V_*}\right)^2}{V_{0,k^*-1}} \log\left(\frac{V_{0,k^*-1}}{tol(N-1)\epsilon^2}+1\right) + M_{min}\left(\max\left\{0, \frac{\kappa}{1-\epsilon^2}\log(tol^{-1}2L(F(\boldsymbol{\xi}_0)-F(\boldsymbol{\xi}^*)))\right\}+\frac{\kappa}{1-\epsilon^2}\right) \tag{111}$$

Proof

$$\mathcal{C}_{k^*-1} = \sum_{\ell=0}^{k^*-1}(1+\mathbb{1}_{\underline{\mathcal{L}_{k^*-1}}}(\ell))M_{\ell,k^*-1} \tag{112}$$

$$\leq \frac{N}{N-1}\frac{\left(\sum_{\ell=0}^{k}\sqrt{(1+\mathbb{1}_{\underline{\mathcal{L}_{k^*-1}}}(\ell))V_{\ell,k^*-1}}\right)^2}{\epsilon^2\left\|\nabla_{\boldsymbol{\xi}}F\left(\boldsymbol{\xi}_{k^*-1}\right)\right\|^2+(N-1)^{-1}\sum_{\ell'=0}^{k}V_{\ell',k^*-1}}+(k^*-1)M_{min} \tag{113}$$

$$\leq 2\frac{\left(\sum_{\ell=0}^{k}\sqrt{(1+\mathbb{1}_{\underline{\mathcal{L}_{k^*-1}}}(\ell))V_{\ell,k^*-1}}\right)^2}{\epsilon^2 tol+(N-1)^{-1}V_{0,k^*-1}}+k^*M_{min} \tag{114}$$

Taking expectation conditioned on the initial iterate,

$$\mathbb{E}\left[\mathcal{C}_{k^*-1} \mid \boldsymbol{\xi}_0\right] \leq \frac{\left(4\sigma\sqrt{L(F(\boldsymbol{\xi}_0)-F(\boldsymbol{\xi}^*))}\left(\frac{1}{1-\sqrt{r}}\right)+\sqrt{V_*}\right)^2}{\epsilon^2 tol+(N-1)^{-1}V_{0,k^*-1}}+\mathbb{E}\left[k^* \mid \boldsymbol{\xi}_0\right]M_{min}. \tag{115}$$

Using the following logarithm inequality with $c/b+1>0$,

$$\frac{a}{b+c} \leq \frac{a}{c}\log\left(\frac{c}{b}+1\right), \tag{116}$$

gives

$$\mathbb{E}\left[\mathcal{C}_{k^*-1} \mid \boldsymbol{\xi}_0\right] \leq \frac{(N-1)\left(4\sigma\sqrt{L(F(\boldsymbol{\xi}_0)-F(\boldsymbol{\xi}^*))}\left(\frac{1}{1-\sqrt{r}}\right)+\sqrt{V_*}\right)^2}{V_{0,k^*-1}} \log\left(\frac{V_{0,k^*-1}}{tol(N-1)\epsilon^2}+1\right) + \mathbb{E}\left[k^* \mid \boldsymbol{\xi}_0\right]M_{min}. \tag{117}$$

Using Corollary 4 and (67) concludes the proof. □

Corollary 8 (Cost analysis of SGD-A on the finite sum case) *If the assumptions of Proposition 2 are satisfied, SGD-A finds an iterate* $\boldsymbol{\xi}_{k^*}$ *such that* $\left\|\nabla_{\boldsymbol{\xi}} F(\boldsymbol{\xi}_{k^*})\right\|^2 \leq tol$ *with expected gradient sampling cost*

$$\mathbb{E}\left[\mathcal{C}_{k^*-1}\right] \leq N \min\left\{1, \log\left(\frac{\frac{2V^*}{tol} + 2\sigma^2}{\epsilon^2(N-1)} + 1\right)\right\} \times \left(\max\left\{0, \frac{\kappa}{1-\epsilon^2}\log\left(tol^{-1} 2L\mathbb{E}\left[F(\boldsymbol{\xi}_0) - F(\boldsymbol{\xi}^*)\right]\right)\right\} + \frac{\kappa}{1-\epsilon^2}\right). \tag{118}$$

Proof When using SGD-A to solve the finite sum minimization problem while taking into consideration that variance goes to zero as $M \to N$, the sample size at iteration k is

$$M_k = \left\lceil \frac{N}{N-1} \frac{V_k}{\epsilon^2 \left\|\nabla_{\boldsymbol{\xi}} F(\boldsymbol{\xi}_k)\right\|^2 + \frac{V_k}{N-1}} \right\rceil, \tag{119}$$

where $V_k = \mathbb{E}\left[\left\|\nabla_{\boldsymbol{\xi}} f(\boldsymbol{\xi}_k, \boldsymbol{\theta}) - \nabla_{\boldsymbol{\xi}} F(\boldsymbol{\xi}_k)\right\|^2 \,\middle|\, \boldsymbol{\xi}_k \right]$. Thus, the total gradient sampling cost to reach iteration k^* is

$$\mathcal{C}_{k^*-1} \leq \sum_{\ell=0}^{k^*-1} \frac{N}{N-1} \frac{V_\ell}{\epsilon^2 \left\|\nabla_{\boldsymbol{\xi}} F(\boldsymbol{\xi}_\ell)\right\|^2 + \frac{V_\ell}{N-1}}. \tag{120}$$

Using (104),

$$\mathcal{C}_{k^*-1} \leq \sum_{\ell=0}^{k^*-1} N \frac{2\sigma^2 \left\|\nabla_{\boldsymbol{\xi}} F(\boldsymbol{\xi}_\ell)\right\|^2 + 2V^*}{\epsilon^2 \left\|\nabla_{\boldsymbol{\xi}} F(\boldsymbol{\xi}_\ell)\right\|^2 (N-1) + 2\sigma^2 \left\|\nabla_{\boldsymbol{\xi}} F(\boldsymbol{\xi}_\ell)\right\|^2 + 2V^*} \tag{121}$$

$$\leq N \frac{2\sigma^2 tol + 2V^*}{tol(\epsilon^2(N-1) + 2\sigma^2) + 2V^*} k^* \tag{122}$$

$$\leq N k^*. \tag{123}$$

Another bound can be obtained from (122) as

$$\mathcal{C}_{k^*-1} \leq N \frac{1}{\frac{\epsilon^2 tol(N-1)}{2\sigma^2 tol + 2V^*} + 1} k^* \tag{124}$$

$$\leq N \log\left(\frac{2\sigma^2 + 2\frac{V^*}{tol}}{\epsilon^2(N-1)} + 1\right) k^*. \tag{125}$$

Taking expectation and using (71) concludes the proof. □

Remark 8 (*More general $\boldsymbol{\theta}$ probability distributions*) Although in Assumption 1 we restricted our attention to the case where the probability distribution of $\boldsymbol{\theta}$, π, does not depend on $\boldsymbol{\xi}$, it is possible to use mappings to address more general cases. Indeed, let us consider the case where

$$\boldsymbol{\theta} = h(\tilde{\boldsymbol{\theta}}, \boldsymbol{\xi}), \tag{126}$$

for some given smooth function h and such that the distribution of $\tilde{\boldsymbol{\theta}}$, $\tilde{\pi}$, does not depend on $\boldsymbol{\xi}$. Then we can simply write, letting $\tilde{f}(\boldsymbol{\xi}, \tilde{\boldsymbol{\theta}}) = f(\boldsymbol{\xi}, h(\tilde{\boldsymbol{\theta}}, \boldsymbol{\xi}))$,

$$F(\boldsymbol{\xi}) = \mathbb{E}\left[f(\boldsymbol{\xi}, \boldsymbol{\theta})|\boldsymbol{\xi}\right] = \mathbb{E}\left[\tilde{f}(\boldsymbol{\xi}, \tilde{\boldsymbol{\theta}})|\boldsymbol{\xi}\right] \tag{127}$$

and, by sampling $\tilde{\boldsymbol{\theta}}$ instead of $\boldsymbol{\theta}$, we are back in the setup of Assumption 1.

Having established theoretical convergence guarantees and gradient sampling cost bounds for `SGD-MICE`, we now turn to practical implementation aspects of the algorithm.

4 `MICE` Algorithm

In this section, we describe the `MICE` algorithm and some of its practical implementation aspects. Before we start, let us discuss the resampling technique used to build an approximated probability distribution for the norm of the gradient.

Remark 9 (*Gradient resampling for calculating sample sizes*) To approximate the empirical distribution of $\|\nabla \mathcal{F}_k\|$, we perform a jackknife [45] resampling of the approximate mean gradient using sample subsets for each iteration $\ell \in \mathcal{L}_k$.

First, for each element $\ell \in \mathcal{L}_k$, we partition the index set $\mathcal{I}_{\ell,k}$ in n_{part} disjoint sets $\mathcal{I}_{\ell,k}^{(1)}, \mathcal{I}_{\ell,k}^{(2)}, .., \mathcal{I}_{\ell,k}^{(n_{\text{part}})}$ with the same cardinality. Then, we create, for each of these sets, their complement with respect to $\mathcal{I}_{\ell,k}$, i.e., $\overline{\mathcal{I}}_{\ell,k}^{(i)} = \mathcal{I}_{\ell,k} \setminus \mathcal{I}_{\ell,k}^{(i)}$ for all $i = 1, 2, .., n_{\text{part}}$. We use these complements to compute the average of these gradient differences excluding a portion of the data,

$$\overline{\mu}_{\ell,k}^{(i)} = \left|\overline{\mathcal{I}}_{\ell,k}^{(i)}\right|^{-1} \sum_{\alpha \in \overline{\mathcal{I}}_{\ell,k}^{(i)}} \Delta_{\ell,k,\alpha}, \tag{128}$$

which we then sample for each $\ell \in \mathcal{L}_k$ to get a single sample of the mean gradient,

$$\nabla_{\boldsymbol{\xi}} \mathcal{F}_{k,\nu} \coloneqq \sum_{\ell \in \mathcal{L}_k} \overline{\mu}_{\ell,k}^{(i_{\ell,\nu})} \tag{129}$$

by independently sampling $i_{\ell,\nu}$ from a categorical distribution with n_{part} categories. Sampling $\nabla_{\boldsymbol{\xi}} \mathcal{F}_{k,\nu}$ n_{samp} times, we construct a set of gradient mean estimates $\{\nabla_{\boldsymbol{\xi}} \mathcal{F}_{k,\nu}\}_{\nu=1}^{n_{\text{samp}}}$. Then, letting $p_{\text{re}} \leq 0.5$ be a quantile of the gradient norms where

$\left\|\nabla_{\boldsymbol{\xi}}\mathcal{F}_k^{\mathrm{re}}\right\|$ is the norm of gradient smaller than the p_{re} quantile, we approximate

$$\mathbb{E}\left[\,\|\nabla\mathcal{F}_k\|\mid\left\{\boldsymbol{\xi}_{\ell'}\right\}_{\ell'\in\mathcal{L}_k}\right]\approx\left\|\nabla_{\boldsymbol{\xi}}\mathcal{F}_k^{\mathrm{re}}\right\|. \tag{130}$$

Similarly, we set a right tail quantile $1-p_{\mathrm{stop}}$ with $p_{\mathrm{stop}}\le 0.5$ to define a gradient norm to be used as a stopping criterion. We stop at k if

$$\left\|\nabla_{\boldsymbol{\xi}}\mathcal{F}_k^{\mathrm{stop}}\right\|^2\le tol, \tag{131}$$

where $\left\|\nabla_{\boldsymbol{\xi}}\mathcal{F}_k^{\mathrm{stop}}\right\|$ is the norm of the gradient respective to the $1-p_{\mathrm{stop}}$ quantile.

The resampling procedure requires storing n_{part} partition-complement means per level, but it does not require additional gradient evaluations.

Algorithm 2 Gradient resampling (partition jackknife) for sizing and stopping.

Require: Index set $\mathcal{L}_k$, level samples $\{\Delta_{\ell,k,\alpha}\}_{\ell\in\mathcal{L}_k,\alpha\in\mathcal{I}_{\ell,k}}$, integers n_{part} and n_{samp}, quantiles $p_{\mathrm{re}}\le 0.5$ and $p_{\mathrm{stop}}\le 0.5$.

1: **for** $\ell\in\mathcal{L}_k$ **do**
2: Partition $\mathcal{I}_{\ell,k}$ into $\mathcal{I}_{\ell,k}^{(1)},\ldots,\mathcal{I}_{\ell,k}^{(n_{\mathrm{part}})}$ and define complements $\overline{\mathcal{I}}_{\ell,k}^{(i)}=\mathcal{I}_{\ell,k}\setminus\mathcal{I}_{\ell,k}^{(i)}$.
3: Compute $\overline{\mu}_{\ell,k}^{(i)}=\left|\overline{\mathcal{I}}_{\ell,k}^{(i)}\right|^{-1}\sum_{\alpha\in\overline{\mathcal{I}}_{\ell,k}^{(i)}}\Delta_{\ell,k,\alpha}$ for all $i=1,\ldots,n_{\mathrm{part}}$.
4: **end for**
5: **for** $\nu=1,\ldots,n_{\mathrm{samp}}$ **do**
6: Sample independently $i_{\ell,\nu}\in\{1,\ldots,n_{\mathrm{part}}\}$ for each $\ell\in\mathcal{L}_k$.
7: Form $\nabla_{\boldsymbol{\xi}}\mathcal{F}_{k,\nu}=\sum_{\ell\in\mathcal{L}_k}\overline{\mu}_{\ell,k}^{(i_{\ell,\nu})}$ and store $\left\|\nabla_{\boldsymbol{\xi}}\mathcal{F}_{k,\nu}\right\|$.
8: **end for**
9: Set $\left\|\nabla_{\boldsymbol{\xi}}\mathcal{F}_k^{\mathrm{re}}\right\|$ as the p_{re}-quantile of $\{\left\|\nabla_{\boldsymbol{\xi}}\mathcal{F}_{k,\nu}\right\|\}_{\nu=1}^{n_{\mathrm{samp}}}$.
10: Set $\left\|\nabla_{\boldsymbol{\xi}}\mathcal{F}_k^{\mathrm{stop}}\right\|$ as the $(1-p_{\mathrm{stop}})$-quantile of $\{\left\|\nabla_{\boldsymbol{\xi}}\mathcal{F}_{k,\nu}\right\|\}_{\nu=1}^{n_{\mathrm{samp}}}$.
11: **Stop** if $\left\|\nabla_{\boldsymbol{\xi}}\mathcal{F}_k^{\mathrm{stop}}\right\|^2\le tol$.

The resampling step performs n_{samp} aggregations of $|\mathcal{L}_k|$ vectors (plus norm evaluations), i.e., $\mathcal{O}\left(n_{\mathrm{samp}}|\mathcal{L}_k|d_{\boldsymbol{\xi}}\right)$ arithmetic operations, with memory overhead proportional to $n_{\mathrm{part}}|\mathcal{L}_k|d_{\boldsymbol{\xi}}$. On the other hand, the confidence on the empirical distribution built by the resampling technique depends on the resampling sample size n_{samp}. Conditional on the partition-complement means $\{\overline{\mu}_{\ell,k}^{(i)}\}$, the resampled norms $\{\left\|\nabla_{\boldsymbol{\xi}}\mathcal{F}_{k,\nu}\right\|\}_{\nu=1}^{n_{\mathrm{samp}}}$ in Algorithm 2 are i.i.d. draws (since the indices $i_{\ell,\nu}$ are sampled independently across ν). Therefore, for the empirical CDF $\widehat{F}_{n_{\mathrm{samp}}}$ of $\left\|\nabla_{\boldsymbol{\xi}}\mathcal{F}_{k,\nu}\right\|$, the

Dvoretzky–Kiefer–Wolfowitz inequality [46, 47] yields

$$\mathbb{P}\left[\sup_t \left|\widehat{F}_{n_{\text{samp}}}(t) - F(t)\right| \le \varepsilon \;\middle|\; \{\overline{\mu}_{\ell,k}^{(i)}\}\right] \ge 1 - 2\exp\left(-2n_{\text{samp}}\varepsilon^2\right). \tag{132}$$

In particular, with probability at least $1-\delta$ (conditionally), choosing $n_{\text{samp}} \ge \frac{1}{2\varepsilon^2}\log\frac{2}{\delta}$ guarantees that the empirical $(1-p_{\text{stop}})$ quantile used for stopping has tail miscoverage at most $p_{\text{stop}} + \varepsilon$ under the resampling distribution.

To control the work of the resampling technique, we measure the runtime needed to get a sample of $\nabla_{\boldsymbol{\xi}}\mathcal{F}_{k,\nu}$ and then set n_{samp} so that the overall time does not exceed a fraction δ_{re} of the remaining runtime of `MICE`. From our numerical tests, we recommend n_{part} to be set between 3 and 10, δ_{re} between 0.1 (for expensive gradients) and 1, $n_{\text{samp}} \ge 10$, and $p_{\text{re}} = 5\%$.

In Algorithm 4, we present the pseudocode for the `MICE` estimator, and in Algorithm 3 we present the algorithm to update the index set $\mathcal{L}_k$ from $\mathcal{L}_{k-1}$ according to Sect. 2.4. Two coupling algorithms for the multi-iteration stochastic optimizers are presented in Appendix A: `SGD-MICE` and `Adam-MICE`.

In general, keeping all gradient realizations for all iterations in memory may be computationally inefficient, especially for large-dimensional problems. To avoid this unnecessary memory overhead, we use Welford's online algorithm to estimate the variances $V_{\ell,k}$ online. We keep in memory only the samples mean and second-centered moments and update them in an online fashion [48]. This procedure makes the memory overhead much smaller than naively storing all gradients and evaluating variances when needed. Therefore, for each $\ell \in \mathcal{L}_k$ at iteration k, we need to store the mean gradient differences estimate, a vector of size $d_{\boldsymbol{\xi}}$; $V_{\ell,k}$, a scalar; and $M_{\ell,k}$, an integer. Also, we store the gradient mean estimate in case we might clip the index set at ℓ in the future, and the respective sum of the variances component-wise, also using Welford's algorithm. Thus, for first-order methods such as `Adam-MICE` and `SGD-MICE`, the memory overhead of `MICE` is of $2|\mathcal{L}_k|(d_{\boldsymbol{\xi}}+2)$ floating-point numbers and $|\mathcal{L}_k|$ integers. Thus, for large-scale problems, dropping iterations and restarting the index set are very important to reduce memory allocation. Regarding the computational overhead, updating each $V_{\ell,k}$ using Welford's algorithm at iteration k has complexity $\mathcal{O}\left((M_{\ell,k} - M_{\ell,k-1})\, d_{\boldsymbol{\xi}}\right)$. Computing the sample sizes using (33) or Algorithm 1 requires a number of operations that is $\mathcal{O}\left(|\mathcal{L}_k|\, d_{\boldsymbol{\xi}}\right)$. While sample sizes might be computed several times per iteration due to the progressive sample size increase, this cost does not increase with the dimensionality of the problem. The resampling technique presented in (129) increases the memory overhead by a factor n_{part} and the computational work by a factor δ_{re}.

5 Numerical Examples

We validate `MICE` through three experiments of increasing complexity: (i) synthetic quadratic functions to study scaling with condition number and ablate operator contributions, (ii) a stochastic Rosenbrock problem to demonstrate coupling with `Adam`,

Algorithm 3 Index-set update operator selection.

procedure `Index Set`$(\mathcal{L}_{k-1}, V_{k,k}, V_{k,k}^{\text{drop}})$
  $\mathcal{L}_k^{\text{add}} \leftarrow \mathcal{L}_{k-1} \cup \{k\}$
  $\mathcal{L}_k^{\text{drop}} \leftarrow \mathcal{L}_{k-1} \cup \{k\} \setminus \{k-1\}$
  $\mathcal{L}_k^{\text{rest}} \leftarrow \{k\}$
  Set $\mathcal{L}_k^{\text{clip},\ell^*}$ as in Sect. 2.4.3 with `Clip` "A" or "B"
  Set $\mathcal{L}_k \leftarrow \mathcal{L}_k^{\text{add}}$
  Compute $\Delta\mathcal{W}_k(\mathcal{L}_k)$ and $\Delta\mathcal{W}_k(\mathcal{L}_k^{\text{drop}})$ using $V_{k,k}$, $V_{k,k}^{\text{drop}}$ and (39)
  if $\Delta\mathcal{W}_k(\mathcal{L}_k^{\text{drop}}) \leq (1+\delta_{\text{drop}})\Delta\mathcal{W}_k(\mathcal{L}_k)$ **then**
    $\mathcal{L}_k \leftarrow \mathcal{L}_k^{\text{drop}}$
  end if
  if $\Delta\mathcal{W}_k(\mathcal{L}_k^{\text{rest}}) < (1+\delta_{\text{rest}})\Delta\mathcal{W}_k(\mathcal{L}_k)$ **then**
    $\mathcal{L}_k \leftarrow \mathcal{L}_k^{\text{rest}}$
  end if
  if $\min_{\ell\in\mathcal{L}_{k-1}} \Delta\mathcal{W}_k(\mathcal{L}_k^{\text{clip},\ell}) < \Delta\mathcal{W}_k(\mathcal{L}_k)$ **then**
    Clip $\mathcal{L}_k$ at the minimizing ℓ
  end if
  return $\mathcal{L}_k$
end procedure

and (iii) large-scale logistic regression comparing against state-of-the-art variance reduction baselines.

When using `SGD`, with or without `MICE`, we assume the constant L to be known and use it to compute the step-size $\eta = 1/L$. As a measure of the performance of the algorithms, we use the optimality gap, which is the difference between the approximate optimal value at iteration k and the exact optimal value,

$$F(\boldsymbol{\xi}_k) - F(\boldsymbol{\xi}^*). \tag{133}$$

In some examples, we know the optimal value and optimal point analytically; otherwise, we estimate numerically by letting optimization algorithms run for many iterations.

As for `MICE` parameters, when coupled with `SGD`, we use $\epsilon = \sqrt{1/3}$, and when coupled with `Adam` we use $\epsilon = 1$. The other parameters are fixed for all problems, showing the robustness of `MICE` with respect to the tuning: $\delta_{\text{drop}} = 0.5$, $\delta_{\text{rest}} = 1.0$, M_{min} is set to 5 for general iterations and 500 for restarts, and the maximum index

Algorithm 4 The `MICE` estimator.

procedure `MICE`

$\mathcal{I}_k \leftarrow \{\alpha\}_{\alpha=1}^{M_{min}}$

Sample $\boldsymbol{\theta}_\alpha \sim \pi \qquad \forall \alpha \in \mathcal{I}_k$

Compute $\nabla_{\boldsymbol{\xi}} f(\boldsymbol{\xi}_k, \boldsymbol{\theta}_\alpha)$

if $k = 0$ **then**

Set $\mathcal{L}_k \leftarrow \{0\}$ ▷ No differences at $k = 0$

Compute $V_{k,k} = \mathbb{E}\left[\left\| \nabla_{\boldsymbol{\xi}} f(\boldsymbol{\xi}_k, \boldsymbol{\theta}_\alpha) - \nabla_{\boldsymbol{\xi}} F(\boldsymbol{\xi}_k) \right\|^2 \,\middle|\, \boldsymbol{\xi}_k \right]$

else

Compute $\nabla_{\boldsymbol{\xi}} f(\boldsymbol{\xi}_{k-1}, \boldsymbol{\theta}_\alpha)$ and $\nabla_{\boldsymbol{\xi}} f(\boldsymbol{\xi}_{p_k(k-1)}, \boldsymbol{\theta}_\alpha)$

Compute $\begin{cases} V_{k,k} = \mathbb{E}\left[\left\| \nabla_{\boldsymbol{\xi}} f(\boldsymbol{\xi}_k, \boldsymbol{\theta}_\alpha) - \nabla_{\boldsymbol{\xi}} f(\boldsymbol{\xi}_{k-1}, \boldsymbol{\theta}_\alpha) \right\|^2 \,\middle|\, \boldsymbol{\xi}_k, \boldsymbol{\xi}_{k-1} \right] \\ V_{k,k}^{\text{drop}} = \mathbb{E}\left[\left\| \nabla_{\boldsymbol{\xi}} f(\boldsymbol{\xi}_k, \boldsymbol{\theta}_\alpha) - \nabla_{\boldsymbol{\xi}} f(\boldsymbol{\xi}_{p_k(k-1)}, \boldsymbol{\theta}_\alpha) \right\|^2 \,\middle|\, \boldsymbol{\xi}_k, \boldsymbol{\xi}_{p_k(k-1)} \right] \end{cases}$

Use Algorithm 3 to set $\mathcal{L}_k$

end if

while $\sum_{\ell \in \mathcal{L}_k} \frac{V_{\ell,k}}{M_{\ell,k}} - \frac{V_{\ell,k}}{N} \geq \epsilon^2 \left\| \nabla_{\boldsymbol{\xi}} F(\boldsymbol{\xi}_k) \right\|^2$ **do** ▷ For expectation minimization, $\frac{V_{\ell,k}}{N} = 0$.

Calculate $\left\{ M_{\ell,k}^* \right\}_{\ell \in \mathcal{L}_k}$ from (33) or Algorithm 1 using $\left\| \nabla_{\boldsymbol{\xi}} F(\boldsymbol{\xi}_k) \right\| \approx \left\| \nabla_{\boldsymbol{\xi}} \mathcal{F}_k^{\text{re}} \right\|$

for $\ell \in \mathcal{L}_k$ **do**

$\Delta M_{\ell,k} = \min\{(M_{\ell,k}^* - M_{\ell,k})_+, 2M_{\ell,k}, N - M_{\ell,k}\}$ ▷ $(a)_+ := \max\{a, 0\}$

$\mathcal{I}'_\ell \leftarrow \{\alpha\}_{\alpha=M_{\ell,k}+1}^{M_{\ell,k}+\Delta M_{\ell,k}}$

Sample $\boldsymbol{\theta}_\alpha \sim \pi \qquad \forall \alpha \in \mathcal{I}'_\ell$

Obtain $\Delta_{\ell,k,\alpha}$ from (14) for each $\alpha \in \mathcal{I}'_\ell$

Calculate $V_{\ell,k}$ from (22)

Get $\nabla_{\boldsymbol{\xi}} \mathcal{F}_k$ using (13)

end for

$M_{\ell,k} \leftarrow M_{\ell,k}^*$

end while

return $\nabla_{\boldsymbol{\xi}} \mathcal{F}_k = \sum_{\ell \in \mathcal{L}_k} \frac{1}{M_{\ell,k}^*} \sum_{\alpha \in \mathcal{I}_{\ell,k}} \Delta_{\ell,k,\alpha}$ from (13)

end procedure

set cardinality is set to 100 except when noted (in the sensitivity studies below and in the logistic regression over the dataset *HIGGS*.). For the continuous cases, we use the clipping "A", whereas, for the finite case, we use clipping "B". As for the resampling parameters, we use $n_{\text{part}} = 5$, $\delta_{\text{re}} = 1.0$, and $p_{\text{re}} = 0.05$ with a minimum resampling size of 10. In our numerical examples, we report the runtime taken by the different algorithms we test in addition to the usual number of gradient evaluations. Note, however, that the current `MICE` implementation is not optimized for performance, and could be much improved in this sense. Regarding the stopping criterion, except in the first example, we do not define a tol. Instead, we define a fixed gradient sampling cost that, when reached, halts execution. This choice allows us to better compare `SGD-MICE` with other methods.

The Python implementation of `MICE` and baseline methods used to generate the data and figures presented in this section is available at GitHub.[3] Moreover, `MICE` can be installed using PyPI; see the package documentation[4] for more information.

5.1 Random Quadratic Function

This problem is a simple numerical example devised to test the performance of `SGD-MICE` on the minimization of a strongly convex function. The function whose expected value we want to minimize is

$$f(\boldsymbol{\xi}, \theta) = \frac{1}{2}\boldsymbol{\xi} \cdot \boldsymbol{H}(\theta)\,\boldsymbol{\xi} - \boldsymbol{b} \cdot \boldsymbol{\xi}, \tag{134}$$

where

$$\boldsymbol{H}(\theta) \coloneqq \boldsymbol{I}_2(1-\theta) + \begin{bmatrix} 2\kappa & 0.5 \\ 0.5 & 1 \end{bmatrix}\theta, \tag{135}$$

$\boldsymbol{I}_2$ is the identity matrix of size 2, $\boldsymbol{b}$ is a vector of ones, and $\theta \sim \mathcal{U}(0,1)$. We use $\kappa = 100$ and initial guess $\boldsymbol{\xi}_0 = (20, 50)$. The objective function to be minimized is

$$F(\boldsymbol{\xi}) = \frac{1}{2}\boldsymbol{\xi} \cdot \mathbb{E}\left[\boldsymbol{H}(\theta)\right]\,\boldsymbol{\xi} - \boldsymbol{b} \cdot \boldsymbol{\xi}, \tag{136}$$

where

$$\mathbb{E}\left[\boldsymbol{H}(\theta)\right] = \begin{bmatrix} \kappa + 0.5 & 0.25 \\ 0.25 & 1 \end{bmatrix}. \tag{137}$$

The optimal point of this problem is $\boldsymbol{\xi}^* = \mathbb{E}\left[\boldsymbol{H}(\theta)\right]^{-1}\boldsymbol{b}$. To perform optimization using `SGD-MICE` and `SGD`, we use the unbiased gradient estimator

$$\nabla_{\boldsymbol{\xi}} f(\boldsymbol{\xi}, \theta) = \boldsymbol{H}(\theta)\,\boldsymbol{\xi} - \boldsymbol{b}. \tag{138}$$

We use the eigenvalues of the Hessian of the objective function, $\mathbb{E}\left[\boldsymbol{H}(\theta)\right]$, to calculate L and thus define the step-size as $1/L$. We set a stopping criterion of $tol = 10^{-8}$.

[3] https://github.com/agcarlon/mice.

[4] https://mice.readthedocs.io.

In Figs. 1 and 2, we present the optimality gap (133), the squared distance to the optimal point and the squared norm of the gradient estimate versus iteration and number of gradient sampling cost, respectively. In Fig. 2 we also plot the iteration reached versus gradient sampling cost. We mark the starting points, restarts, and ending points with blue, red, and purple squares, respectively; the dropped points with black $\times$, and the remaining iterations in the MICE index set with cyan dots. In Fig. 1, one can observe that SGD-MICE attains linear convergence with a constant step-size, as predicted in Proposition 2. In Fig. 2, we present the convergence plots versus gradient sampling cost, exhibiting numerical rates of $\mathcal{O}(\mathcal{C}_k^{-1})$. These rates are expected as the distance to the optimal point converges linearly (see Corollary 2) and the cost of sampling new gradients per iteration grows as $\mathcal{C}_k = \mathcal{O}(\left\|\nabla F(\boldsymbol{\xi}_k)\right\|^{-2})$, as shown in (33). Note that the convergence is exponential as in the deterministic case until around 4×10^4 gradient evaluations. After this point, there is a change of regime in which we achieve the asymptotic rates; after 4×10^4, the cost of performing each iteration grows exponentially.

Figure 3 presents the cardinality of the index set $\mathcal{L}_k$, the true squared relative L^2 error, an approximation of the error that we control to be less than ϵ, and $V_{k,k}$ versus iteration. Moreover, on the relative error plots, horizontal lines with the upper bounds we impose are presented. It can be seen that imposing the condition $\sum_{\ell \in \mathcal{L}_k} V_{\ell,k}/M_{\ell,k} \leq \epsilon^2 \left\|\nabla \mathcal{F}_k\right\|^2$ results in the actual relative squared error being below ϵ. Also, we split the empirical relative error between bias and statistical error, as discussed in Appendix B, C, D. Note that the bias is reset when MICE's index set is restarted. The $V_{k,k}$ plot illustrates how the variance of the last gradient difference decreases with the optimization, notably when a new element is added to the set $\mathcal{L}_k$.

To validate the robustness and performance increase of SGD-MICE, we compare it with SGD using Monte Carlo sampling, at every iteration, to estimate the mean gradient, which we call SGD-A. Likewise, SGD-MICE, SGD-A controls the relative error in the mean gradient approximation. In practice, SGD-A is SGD-MICE but only equipped with the Restart index set operator. We run both methods until the tolerance $tol = 10^{-8}$ is reached. Then, we run *vanilla* SGD (Robbins–Monro algorithm [32]) with sample size 1000 until it reaches the same work as SGD-A.

In Fig. 4, we present the optimality gap per iteration and number of gradient evaluations for SGD-A, SGD-MICE, and *vanilla* SGD. Our SGD-MICE achieved the desired tolerance with 3% of the cost needed by SGD-A, illustrating the performance improvement of using the data from previous iterations efficiently.

Although this example is very simple, it illustrates the performance of SGD-MICE in an ideal situation where both L and μ are known. Finally, SGD-MICE was able to automatically decide whether to drop iterations, restart, or clip the index set to minimize the overall work required to attain the linear convergence per iteration.

In Sect. 3.2, we prove that, for expectation minimization, the gradient sampling cost necessary to reach a certain $\left\|\nabla_{\boldsymbol{\xi}} F(\xi_{k^*})\right\|^2 < tol$ is $\mathcal{O}\left(\kappa^2 tol^{-1}\right)$ for SGD-MICE and $\mathcal{O}\left(\kappa tol^{-1} \log(tol^{-1})\right)$ for SGD-A. To validate numerically the dependency of the cost with respect to the conditioning number, we evaluated both SGD-MICE and SGD-A with different condition numbers until the stopping criterion. Moreover, we also tested SGD-MICE with and without the index set operators Restart, Drop,

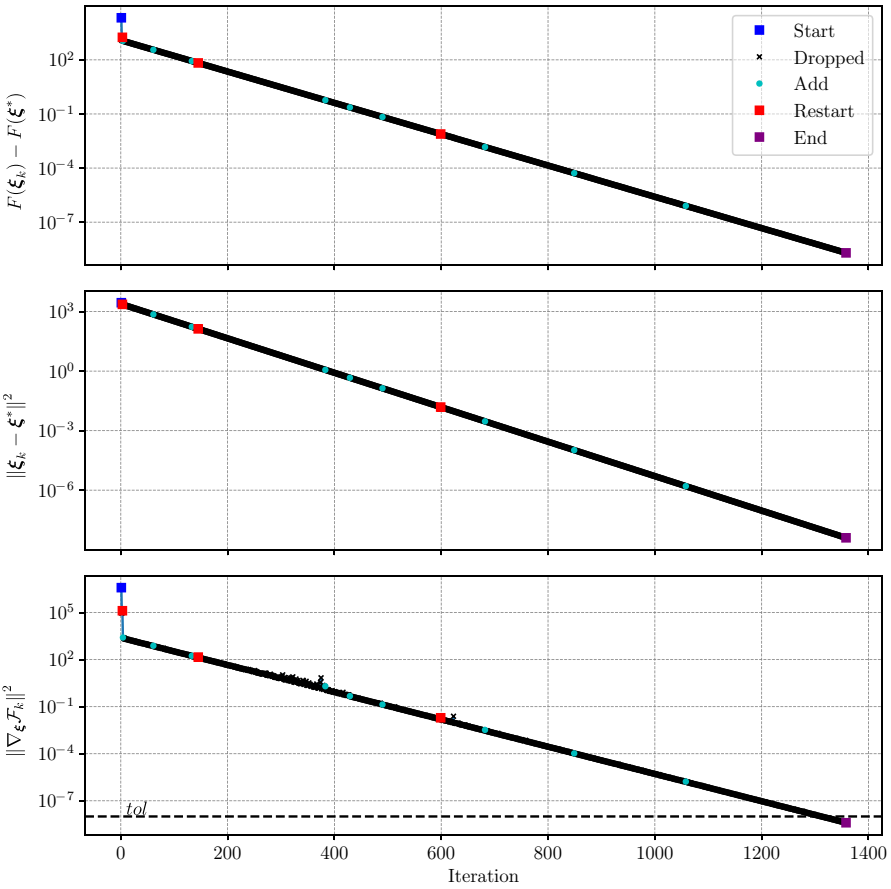


Fig. 1 Single run, random quadratic example, Eq. (136) with $\kappa = 100$. Optimality gap (top), squared distance to the optimal point (center), and squared norm of gradient estimate (bottom) per iteration for `SGD-MICE`. The starting point, the restarts, and the end are marked respectively as blue, red, and purple squares, iterations dropped with black $\times$, and the remaining `MICE` points with cyan circles. `SGD-MICE` is able to achieve linear L^2 convergence as predicted in Proposition 2

and `Clip`. The reasoning for doing this test is that, in the analysis of Corollary 5, we consider the case where all iterates are kept in the index set. However, in practice, one would expect `SGD-MICE` with the index set operators to perform better than both *vanilla* `SGD-MICE` (without the operators) and `SGD-A`; in one extreme case where all iterates are kept, we recover *vanilla* `SGD-MICE`, and in another extreme case we restart every iteration, resulting in `SGD-A`. The gradient sampling cost versus κ for these tests is presented in Fig. 5.

Figure 5 illustrates that enabling the index-set operators (notably `Restart`) changes the observed scaling with the condition number from $\mathcal{O}\left(\kappa^2\right)$ (add-only, no operators) toward $\mathcal{O}\left(\kappa\right)$ (up to the fixed $\log(tol^{-1})$ factor for `SGD-A`). This behavior is consistent with the fact that `SGD-A` is a special case of `SGD-MICE` obtained by restarting at every iteration. Our implementation selects at each iteration the opera-

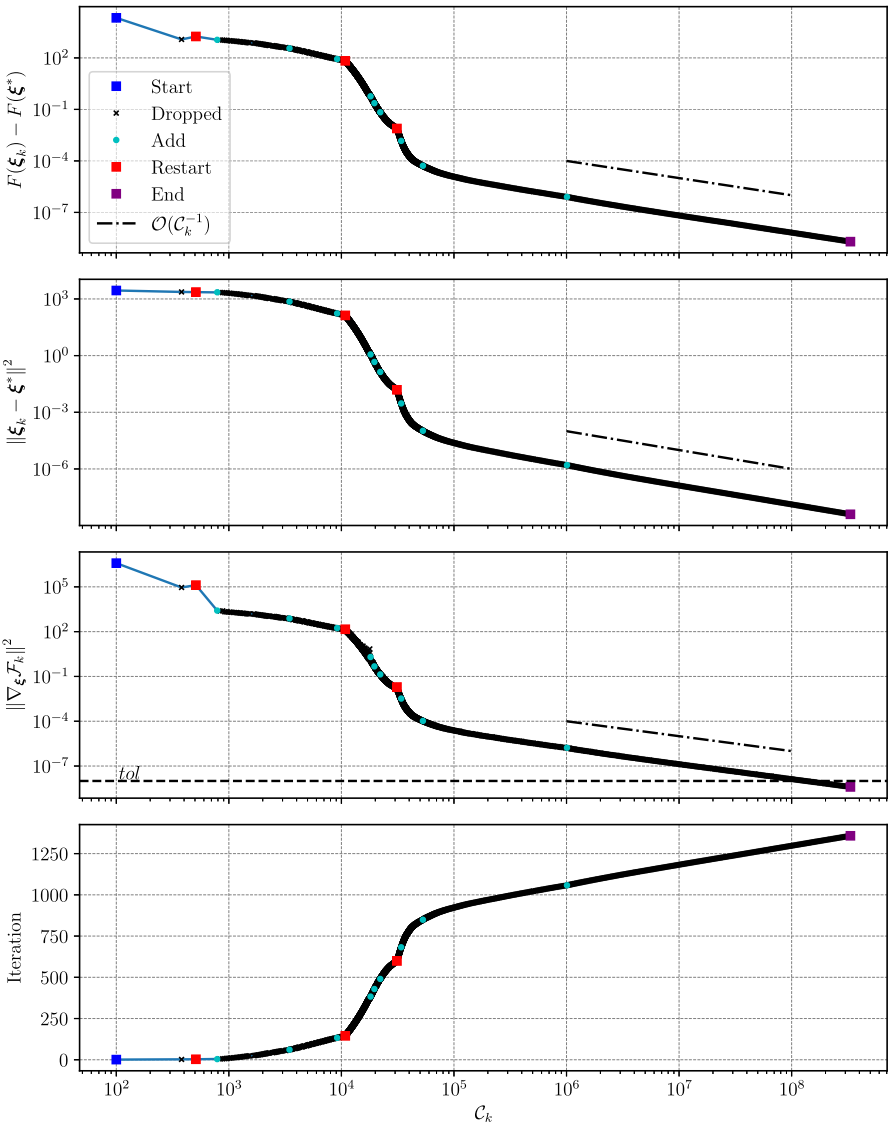


Fig. 2 Single run, random quadratic example, Eq. (136) with $\kappa = 100$. Optimality gap (top), squared distance to the optimal point (center top), squared norm of gradient estimate (center bottom), and number of iterations (bottom) per number of gradient evaluations for `SGD-MICE`. The starting point, the restarts, and the end are marked respectively as blue, red, and purple squares, iterations dropped with black $\times$, and the remaining `MICE` points with cyan circles. The asymptotic convergence rate of $\mathcal{O}(\mathcal{C}_k^{-1})$ is presented when expected

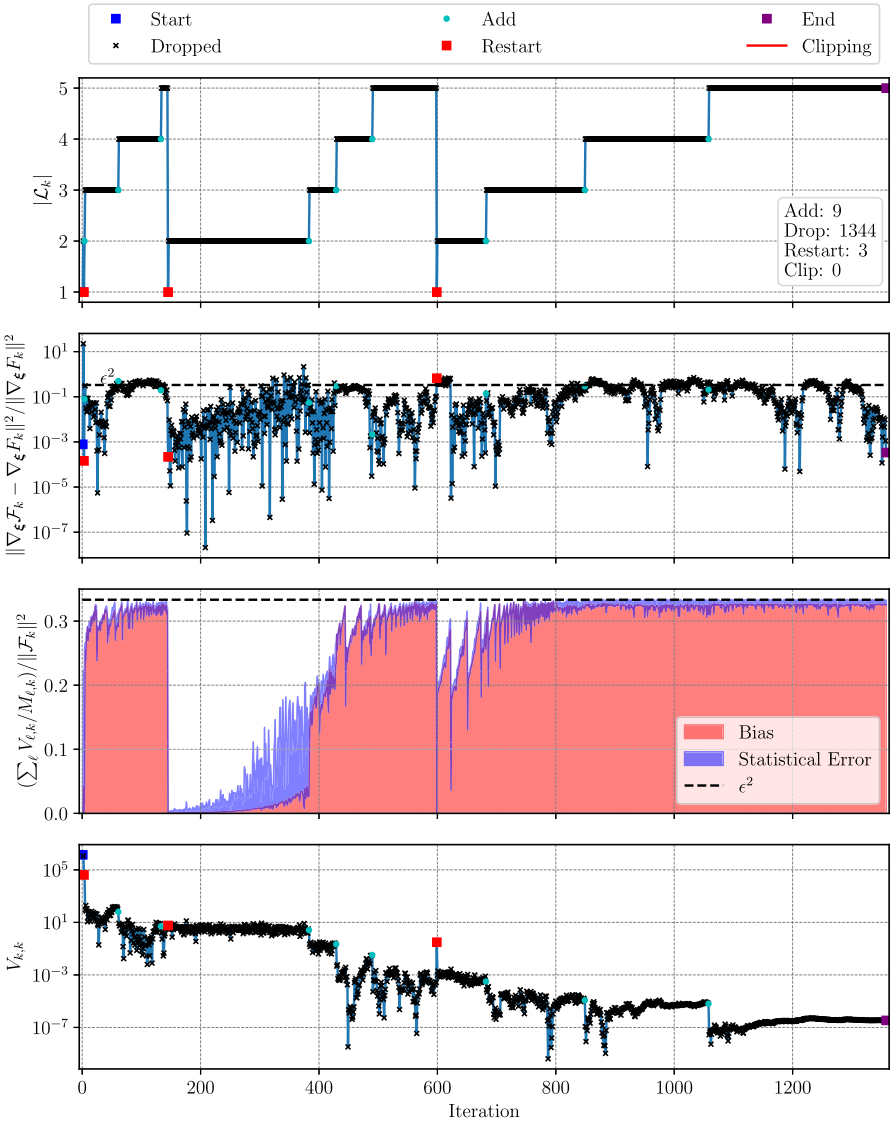


Fig. 3 Single run, random quadratic example, Eq. (136) with $\kappa = 100$. From top to bottom, cardinality of the index set, true squared relative error, empirical relative error, and $V_{k,k}$ versus iteration. The starting point, the restarts, and the end are marked respectively as blue, red, and purple squares, iterations dropped with black $\times$, and the remaining MICE points with cyan circles. Dashed lines represent bounds used to control relative errors when applied. In the empirical relative error plot, we split the relative error between bias and statistical error

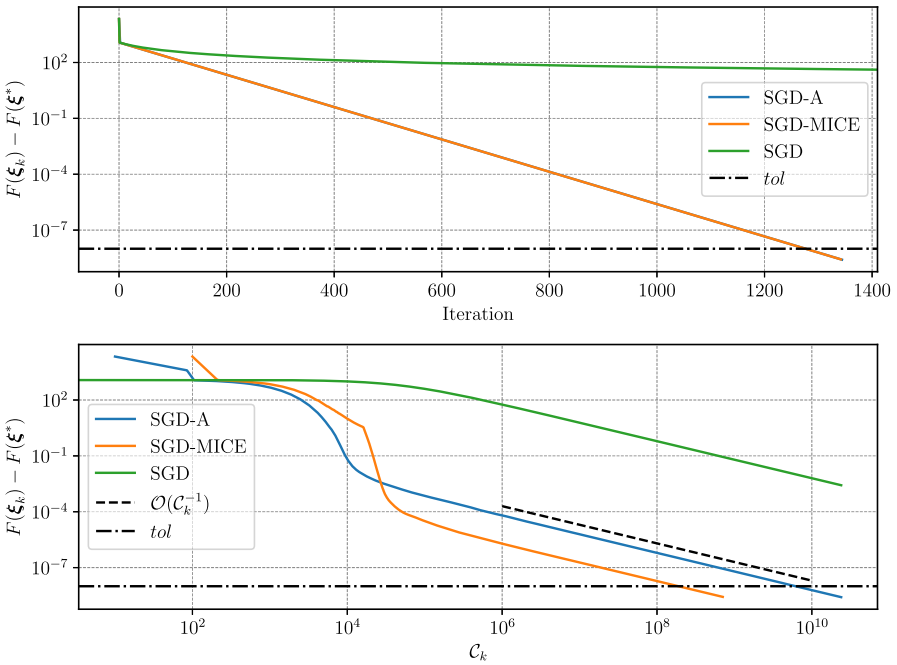


Fig. 4 Single run, random quadratic example, Eq. (136) with $\kappa = 100$. Optimality gap versus iteration (top) and gradient sampling cost (bottom) for `SGD-A`, `SGD-MICE`, and *vanilla* `SGD`. Dash-dotted lines represent tol, and the dashed line in the bottom plot illustrates the expected convergence rate of the optimality gap per cost, $\mathcal{O}\left(\mathcal{C}_k^{-1}\right)$. The top plot is limited to 1400 iterations to illustrate `SGD-A` and `SGD-MICE` even though `SGD` required close to 2.4×10^6 iterations. `SGD-MICE` achieves tol with less than 3% of the sampling cost of `SGD-A` and both achieve a much lower optimality gap than `SGD` for the same cost

tor that minimizes a per-iteration work proxy, so it can be interpreted as competing locally with this restart baseline. While this local comparison does not by itself yield a full global complexity guarantee for the adaptive operator selection (since future variance structure depends on past decisions), it provides intuition for why restarting can remove the additional κ factor in the cost and why the operator-enhanced method empirically matches the $\mathcal{O}(\kappa)$ trend in this example.

In Remark 9, we present a resampling technique to take more informed decisions on stopping criterion and error control. To validate our stopping criterion, we performed a thousand independent runs of `SGD-MICE` for different values of tol, using the resampling to decide both sample sizes and the stopping criterion. Figure 6 presents violin plots with approximations of empirical distributions of the squared gradient norms where optimization stopped and the percentage of times this quantity exceeded tol. Moreover, we show both the case where we use the resampling technique and when we do not use it. For lower tolerances, the resampling technique indeed reduced the percentage of premature stops, however, in both cases, a general trend of decrease following tol is observed.

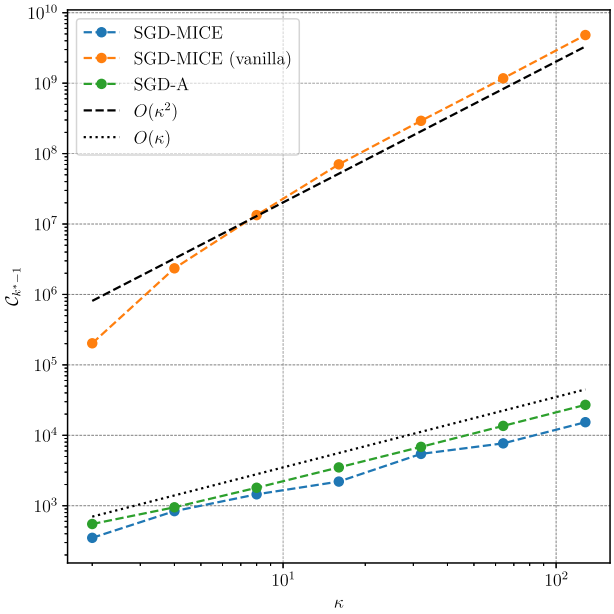


Fig. 5 Gradient sampling cost versus condition number for *vanilla* `SGD-MICE` (without `Restart`, `Drop`, or `Clip`), `SGD-MICE` (with `Restart`, `Drop`, and `Clip`), and `SGD-A`. The algorithms are run until they reach the stopping criterion defined as $\left\|\nabla_{\xi} F(\xi_{k^*})\right\|^2 < tol$. We also plot reference lines for $\mathcal{O}\left(\kappa^2\right)$ and $\mathcal{O}(\kappa)$. Note that *vanilla* `SGD-MICE` cost increases as $\mathcal{O}\left(\kappa^2\right)$ as predicted in Corollary 5 whereas `SGD-A` cost increases as $\mathcal{O}(\kappa)$, as predicted in Corollary 6. Surprisingly, once the index set operators `Restart`, `Drop`, and `Clip` are considered, `SGD-MICE` cost dramatically decreases, not only by a constant factor but effectively matching the rate of `SGD-A` of $\mathcal{O}(\kappa)$

5.1.1 Benchmarks and Ablations

To evaluate the performance of `SGD-MICE` in higher-dimensional settings, we use a d-dimensional strongly-convex quadratic problem, which generalizes the two-dimensional setup of Sect. 5.1 (Eq. (136)). First, we benchmark the runtime of `SGD-MICE` and break it down into the contributions of gradient evaluations, variance estimation (using the Welford algorithm), resampling (as in Remark 9), and the index set operators defined in Sect. 2.4. Moreover, we break the index set benchmark further down into the contributions of `Add`, `Drop`, `Restart`, and `Clip`. The average runtime shares over $n = 50$ runs (with a cap of 100,000 gradient evaluations per run) are presented in Fig. 7.

To quantify the contribution of each index-set operator under a fixed gradient budget, we run an ablation on the same quadratic setup (dimension $d = 100$, condition number $\kappa = 100$) with a cap of 100,000 gradient evaluations per run. We repeat each configuration $n = 50$ times and report the mean and standard deviation of the relative optimality (final optimality gap divided by initial gap) and the mean number of gradient evaluations used. The configurations are: `Add` only (no `Drop` , `Restart` , or `Clip`); `Add` +`Drop` ; `Add` +`Drop` +`Restart` ; and All+Clip (`Add` , `Drop` ,

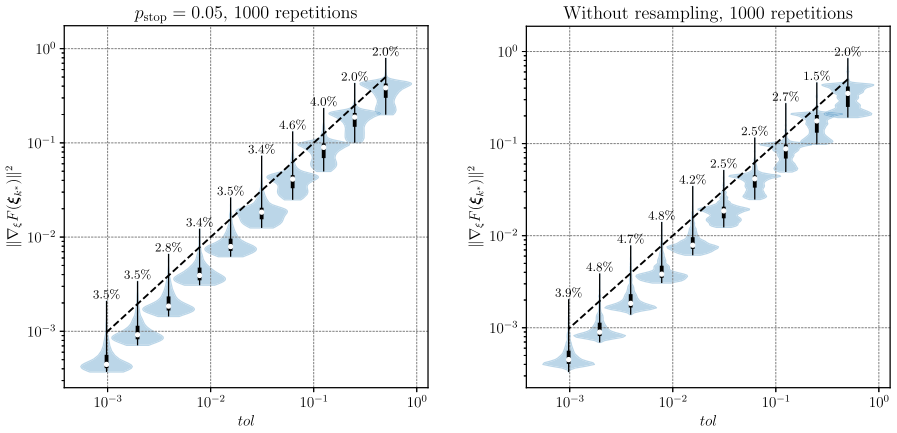


Fig. 6 Random quadratic problem: consistency plot to validate the stopping criterion with the resampling technique (left) and without it (right). We present violin plots for the squared norm of the final gradient for different values of tol. The light blue shade represents an empirical pdf approximated using a Gaussian kernel density estimate, the thin hair represents the interval between maximum and minimum, the thick hair illustrates the quantiles between 0.25 and 0.75, and the white dot marks the median. A thousand independent runs were used to obtain the data presented. The dashed lines represent tol and the percentage of runs with $\left\|\nabla_{\boldsymbol{\xi}} F(\xi_{k^*})\right\|^2 > tol$ are presented for each tol

`Restart` , and `Clip`). Table 2 summarizes the results; the mean number of times each operator was triggered (drop, restart, clip) is also reported.

The operator ablation results in Table 2 reveal that the `Restart` operator is critical for achieving high-quality solutions under a fixed gradient budget: enabling `Restart` reduces the mean relative optimality from approximately 10^{-2} (`Add` +`Drop`) to 10^{-9} (`Add` +`Drop` +`Restart`), a dramatic improvement of seven orders of magnitude. In contrast, `Drop` alone yields only a modest improvement over the add-only baseline (from 0.0120 to 0.00974), and `Clip` has minimal additional impact in this quadratic setting. The frequency counts show that `Drop` is invoked frequently (approximately 940 times per run) once `Restart` is enabled, whereas `Restart` itself triggers only 1–3 times and `Clip` is rarely needed ($\sim$ 6 times). These findings confirm that our greedy index set policy effectively balances the trade-off between reusing accumulated gradient information (via `Add` and selective `Drop`) and periodically resetting the estimator (via `Restart`).

Table 3 shows sensitivity to the relative-error tolerance ϵ (All+Clip, same quadratic setup, budget 100,000, $n = 50$ runs per value). Under a fixed gradient budget, larger values of ϵ yield better final optimality; increasing ϵ from 0.3 to 1.0 improves the mean relative optimality from 3.64×10^{-9} to 7.94×10^{-10}. This behavior is expected, since a looser error tolerance allows the algorithm to spend fewer gradients per iteration (smaller batch sizes at each level), thereby enabling more iterations and further progress toward the optimum within the same total gradient budget. While the theoretical cost analysis in Sect. 3.2 identifies $\epsilon = 1/\sqrt{3}$ as minimizing the leading-order sampling cost under the variance-sum sizing rule, the present fixed-budget experiments suggest that the optimal choice of ϵ depends on the problem at hand.

Table 2 Operator ablation on the random quadratic problem ($d = 100, \kappa = 100$): fixed budget of 100,000 gradient evaluations, $n = 50$ runs

Configuration	Mean (rel. opt.)	Std	Mean grad. evals	Drop	Restart	Clip
`Add` only	0.0120	0.00230	99,708	0.0	0.0	0.0
`Add` +`Drop`	0.00974	0.00077	99,719	184.7	0.0	0.0
`Add` +`Drop` +`Restart`	1.18×10^{-9}	9.48×10^{-10}	99,645	947.6	2.9	0.0
All+`Clip`	1.54×10^{-9}	1.33×10^{-9}	99,730	940.9	1.3	5.8

Relative optimality is final gap/initial gap

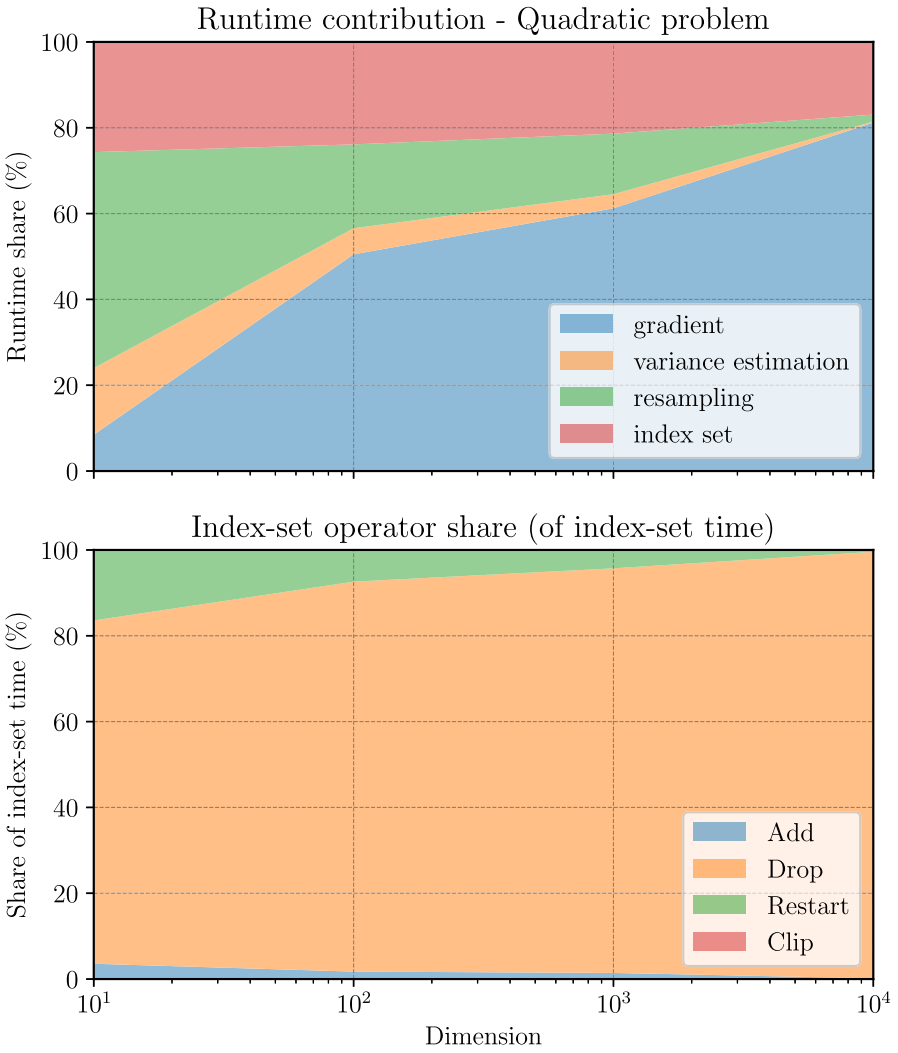


Fig. 7 Quadratic problem benchmark: on top, the runtime breakdown for `SGD-MICE` versus dimensionality, and on the bottom, a breakdown of the runtime spent by each index set operator. The gradient evaluation cost dominates for dimensions $d \geq 100$, with operator overhead remaining under 10% across all tested dimensions. Within the index-set operators, `Drop` dominates the runtime because it requires gradient evaluations to determine whether to discard the most recent iterate

Table 4 shows sensitivity to δ_{drop} (left) and δ_{rest} (right): relative optimality and mean number of drop/restart events (All+Clip; for the drop sweep $\delta_{\text{rest}} = 0$, for the restart sweep $\delta_{\text{drop}} = 0.5$). Same quadratic setup, budget $100,000$, $n = 50$ runs per value. The results indicate that performance is fairly robust to the choice of δ_{drop}: all tested values (0.0 to 1.0) achieve similar relative optimality and trigger comparable numbers of drop events (871 to 947 per run). In contrast, the restart threshold δ_{rest} exhibits a more pronounced effect: increasing δ_{rest} from 0 to 1.0 improves mean relative optimality by

Table 3 Sensitivity to ϵ ($d = 100$, $\kappa = 100$, budget 100,000, $n = 50$ runs per value)

ϵ	Mean (rel. opt.)	Std
0.3	3.64×10^{-9}	2.16×10^{-9}
0.5	1.76×10^{-9}	1.21×10^{-9}
0.577	1.26×10^{-9}	7.18×10^{-10}
0.7	1.17×10^{-9}	9.02×10^{-10}
1.0	7.94×10^{-10}	6.16×10^{-10}

Table 4 Sensitivity to δ_{drop} and δ_{rest} ($d = 100$, $\kappa = 100$, budget 100,000, $n = 50$ runs per value)

Sensitivity to δ_{drop}			Sensitivity to δ_{rest}		
δ_{drop}	Rel. opt	Mean drops	δ_{rest}	Rel. opt	Mean restarts
0.0	1.30×10^{-9}	871.4	0.0	1.86×10^{-9}	1.4
0.25	1.25×10^{-9}	936.9	0.25	1.11×10^{-9}	1.6
0.5	1.45×10^{-9}	940.8	0.5	1.14×10^{-9}	1.6
0.75	1.63×10^{-9}	940.8	0.75	1.03×10^{-9}	1.7
1.0	1.50×10^{-9}	946.5	1.0	8.59×10^{-10}	1.9

Table 5 Sensitivity to maximum index set cardinality ($d = 100$, $\kappa = 100$, budget 100,000, $n = 50$ runs per value)

Max $\lvert\mathcal{L}_k\rvert$	Mean (rel. opt.)	Std
100	1.53×10^{-9}	1.20×10^{-9}
500	1.48×10^{-9}	1.07×10^{-9}
1000	1.44×10^{-9}	1.05×10^{-9}

roughly a factor of two (from 1.86×10^{-9} to 8.59×10^{-10}) while modestly increasing the restart count from 1.4 to 1.9 events per run. This suggests that a more aggressive restart policy (larger δ_{rest}) is beneficial in this quadratic setting, though the overall frequency of restarts remains low.

Table 5 shows sensitivity to the maximum index set cardinality $\max_k |\mathcal{L}_k|$ (All+Clip, same quadratic §setup, budget 100,000, $n = 50$ runs per value). The results demonstrate robustness to the choice of maximum index-set size: increasing the cap from 100 to 1000 yields only a slight improvement in mean relative optimality (from 1.53×10^{-9} to 1.44×10^{-9}). This indicates that, for this problem, the greedy operator policy naturally maintains a relatively compact index set, so that even a modest cap of 100 does not severely constrain performance. In practice, imposing a cap remains useful to bound memory usage and per-iteration overhead, and the present experiments confirm that moderate caps (e.g., 100–500) suffice for effective optimization on this quadratic benchmark.

5.2 Stochastic Rosenbrock Function

The goal of this example is to test the performance of `Adam-MICE`, that is, `Adam` coupled with our gradient estimator `MICE`, in minimizing the expected value of the stochastic Rosenbrock function in (140), showing that `MICE` can be coupled with different first-order optimization methods in a non-intrusive manner. Here we adapt the deterministic Rosenbrock function to the stochastic setting, specializing our optimization problem (1) with

$$f(\boldsymbol{\xi}, \boldsymbol{\theta}) = (a - \xi_0 + \theta_0)^2 + b\left(-\xi_0^2 + \xi_1 + \theta_0^2 - \theta_1^2\right)^2, \tag{139}$$

where $a = 1, b = 100, \theta_0, \theta_1 \sim \mathcal{N}(0, \sigma_\theta^2)$. The objective function to be minimized is thus

$$F(\boldsymbol{\xi}) = (a - \xi_0)^2 + \sigma_\theta^2 + b\left(4\sigma_\theta^4 + (\xi_1 - \xi_0^2)^2\right), \tag{140}$$

and its gradient is given by

$$\nabla_{\boldsymbol{\xi}} F(\boldsymbol{\xi}) = \begin{bmatrix} -2a + 4b\xi_0^3 - 4b\xi_0\xi_1 + 2\xi_0 \\ -2b\xi_0^2 + 2b\xi_1 \end{bmatrix}, \tag{141}$$

which coincides with the gradient of the deterministic Rosenbrock function. Therefore, the optimal point of the stochastic Rosenbrock is the same as the one of the deterministic: $\boldsymbol{\xi}^* = (a, a^2)$. To perform the optimization, we sample the stochastic gradient

$$\nabla_{\boldsymbol{\xi}} f(\boldsymbol{\xi}, \boldsymbol{\theta}) = \begin{bmatrix} -2a + 4b\xi_0\left(\xi_0^2 - \xi_1 - \theta_0^2 + \theta_1^2\right) + 2\xi_0 - 2\theta_0 \\ 2b\left(-\xi_0^2 + \xi_1 + \theta_0^2 - \theta_1^2\right) \end{bmatrix}. \tag{142}$$

Although this is still a low dimensional example, minimizing the Rosenbrock function poses a difficult optimization problem for first-order methods; these tend to advance slowly in the region where the gradient has near-zero norm. Moreover, when noise is introduced in gradient estimates, their relative error can become large, affecting the optimization convergence.

We compare the convergence of the classical `Adam` algorithm against `Adam-MICE`. To illustrate the effect of the dispersion of the random variable $\boldsymbol{\theta}$, two distinct noise levels are considered, namely $\sigma_\theta = 10^{-4}$ and $\sigma_\theta = 10^{-1}$. As for the optimization setup, we set `Adam-MICE` with fixed step-size 0.3 and `Adam` with a decreasing step-size $\eta_k = 0.01/\sqrt{k}$, which we observed to be the best step-sizes for each method. The stopping criterion for both algorithms is set as 10^7 gradient evaluations. For `Adam-MICE`, we use $\epsilon = 1$, whereas for `Adam` we use a fixed batch size of 100. In all cases, we start the optimization from $\boldsymbol{\xi}_0 = (-1.5, 2.5)$

In Fig. 8, we present, for σ_θ of 10^{-4} (left) and 10^{-1} (right), the optimality gap for both `Adam` and `Adam-MICE` versus the number of gradients, iterations, and runtime in seconds. It is clear that `Adam-MICE` is more stable than `Adam` as the latter oscillates as it approximates the optimal point in both cases. The efficient control of the error

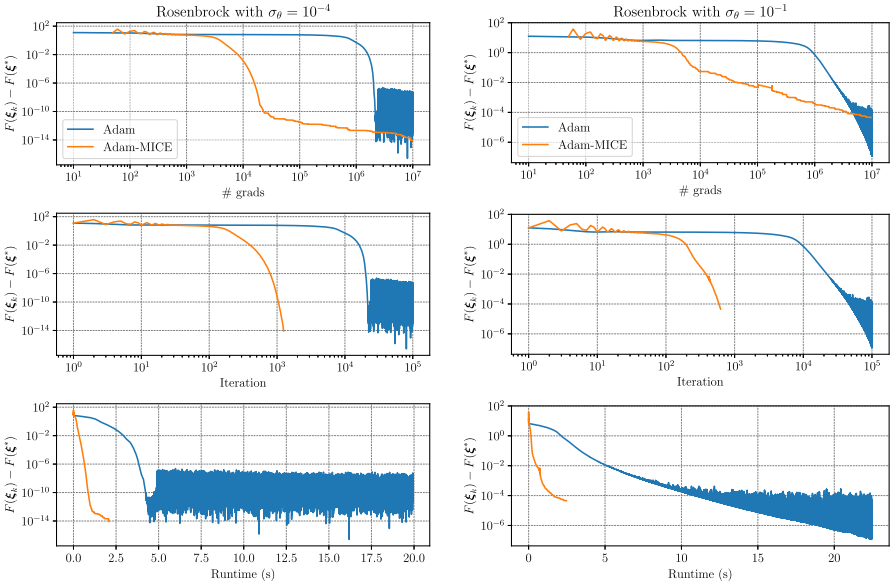


Fig. 8 Single run, stochastic Rosenbrock function example, (140) with $\sigma_\theta = 10^{-4}$. Optimality gap for `Adam` and `Adam-MICE` versus the number of gradient evaluations (top), iterations (center), and runtime in seconds (bottom)

in gradient estimates allows `Adam-MICE` to converge monotonically in the asymptotic phase. Moreover, the number of iterations and the runtime are much smaller for `Adam-MICE` than for `Adam`.

As a conclusion, even though `Adam` has its own mechanisms to control the statistical error of gradients, coupling it with `MICE`, for this example, has proven to be advantageous as it allows more evaluations to be performed simultaneously. Moreover, as the gradient error is controlled, we can use `Adam` with a fixed step-size. Also, `MICE` allows for a stopping criterion based on the gradient norm, which would not be possible for *vanilla* `Adam`.

5.3 Logistic Regression

In this example, we train logistic regression models using `SGD-MICE`, `SAG` [49], `SAGA` [27], `SARAH` [24], and `SVRG` [23] to compare their performances. Here, we present a more practical application of `MICE`, where we can test its performance on high-dimensional settings with finite populations. Therefore, we calculate the error as in (2) and use Algorithm 1 to obtain the optimal sample sizes. To train the logistic regression model for binary classification, we use the ℓ_2-regularized log-loss function

$$F(\boldsymbol{\xi}) = \frac{1}{N}\sum_{i=1}^{N} f\left(\boldsymbol{\xi}, \boldsymbol{\theta}_i = (\boldsymbol{x}_i, y_i)\right) = \frac{1}{N}\sum_{i=1}^{N} \log(1 + \exp(-y_i \boldsymbol{\xi} \cdot \boldsymbol{x}_i)) + \frac{\lambda}{2}\|\boldsymbol{\xi}\|^2, \tag{143}$$

Table 6 Size, number of features, and regularization parameters for the datasets used in the logistic regression example

Dataset	Size	Features	λ	κ	epochs
mushrooms	8124	112	10^{-5}	12316.30	100
gisette	6000	5000	10^{-4}	1811.21	50
HIGGS	11,000,000	28	10^{-4}	765.76	10

Table 7 Step-sizes chosen for each method for the logistic regression example

Method	SAG	SAGA	SARAH	SVRG
Step-size	$\frac{1}{16(\hat{L}+\mu N)}$	$\frac{1}{2(\hat{L}+\mu N)}$	$\frac{1}{2\hat{L}}$	$\frac{1}{2\hat{L}}$

where each data point $(\boldsymbol{x}_i, y_i)$ is such that $\boldsymbol{x}_i \in \mathbb{R}^{d_{\boldsymbol{\xi}}}$ and $y_i \in \{-1, 1\}$. We use the datasets *mushrooms*, *gisette*, and *Higgs*, obtained from LibSVM.[5] The size of the datasets N, number of features $d_{\boldsymbol{\xi}}$, and regularization parameters λ are presented in Table 6.

When using SGD-MICE for training the logistic regression model, we use $\epsilon = 1/\sqrt{3}$. For the other methods, we use batch sizes of size 10. Since we have finite populations, we use Algorithm 1 to calculate the sample-sizes. SGD-MICE step is based on the Lipschitz smoothness of the true objective function as presented in Proposition 2. Conversely, the other methods rely on a Lipschitz constant that must hold for all data points, which we refer to as $\hat{L}$. A maximum index set cardinality of 100 is imposed on SGD-MICE; if $|\mathcal{L}_k| = 100$, we restart the index set. For the *HIGGS* dataset, due to the smaller dimensionality (22 variables), we used a maximum index set cardinality of 1000. The step-sizes for SAG, SAGA, SARAH, and SVRG are presented in Table 7. These steps were chosen as the best performing for each case based on the recommendations of their original papers.

To evaluate the consistency of SGD-MICE versus the other baseline methods, we perform 100 independent runs of each method for each dataset. Figure 9 presents confidence intervals and medians of the relative optimality gap (the optimality gap normalized by its starting value) for the *mushrooms*, *gisette*, and *HIGGS* datasets versus the number of gradient evaluations, iterations, and runtimes in seconds.

In the three cases studied, SGD-MICE performed better than the baseline methods in terms of optimality gap decrease per gradient sampling cost and number of iterations. However, the runtime of SGD-MICE exceeded that of the other methods due to the MICE estimator overhead. We note, however, that the gradients in this logistic regression model are fairly cheap; in the case where gradient evaluations are more expensive, the MICE overhead becomes negligible, cf. Fig. 7.

Figure 10 presents the index set cardinalities versus iterations of SGD-MICE for the three datasets. Moreover, we present the iterations that were kept in the index set, the ones that were dropped, as well as restarts and clippings.

[5] https://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/binary.html.

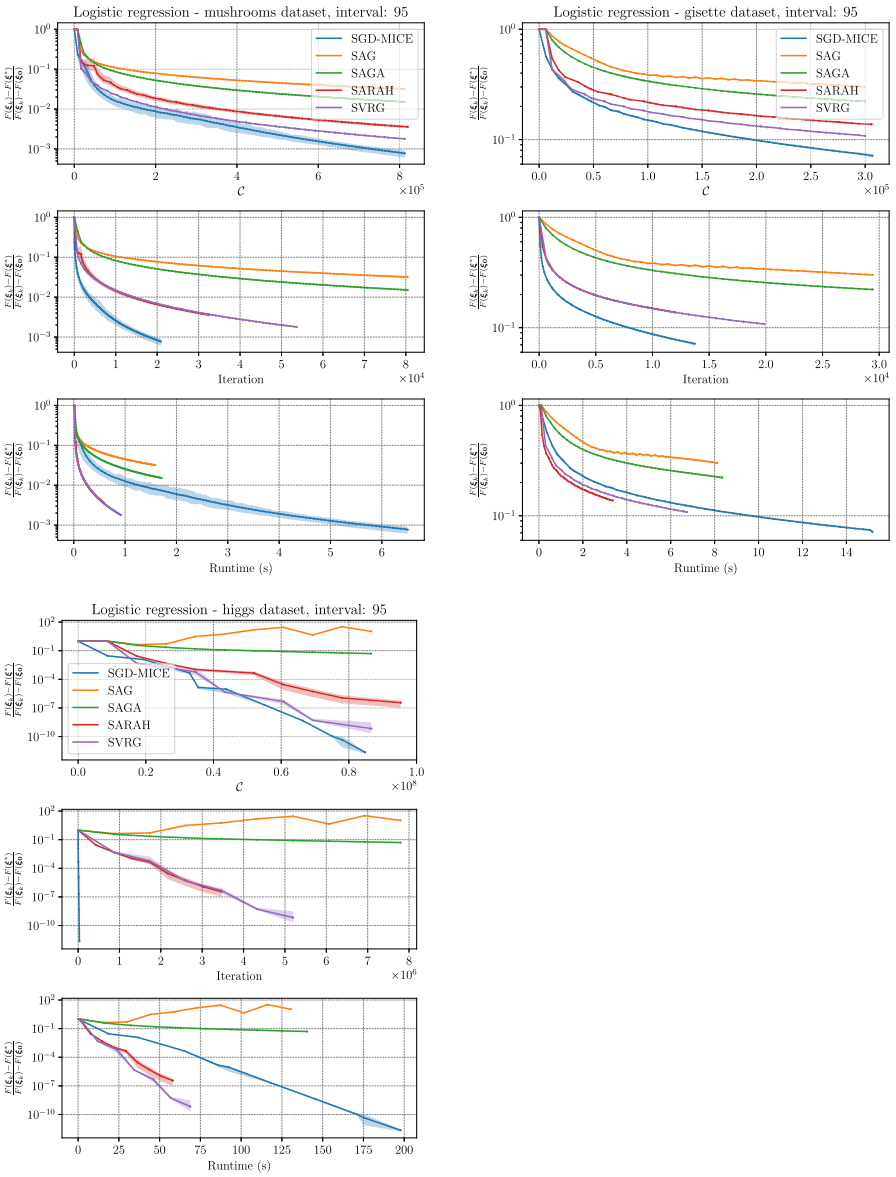


Fig. 9 A hundred runs, logistic regression examples. Relative optimality gap versus gradient sampling cost (top), iterations (center), and runtime in seconds (bottom) for `SGD-MICE`, `SAG`, `SAGA`, `SARAH`, and `SVRG`. The shaded regions represent confidence intervals between percentiles encompassing 95% of values

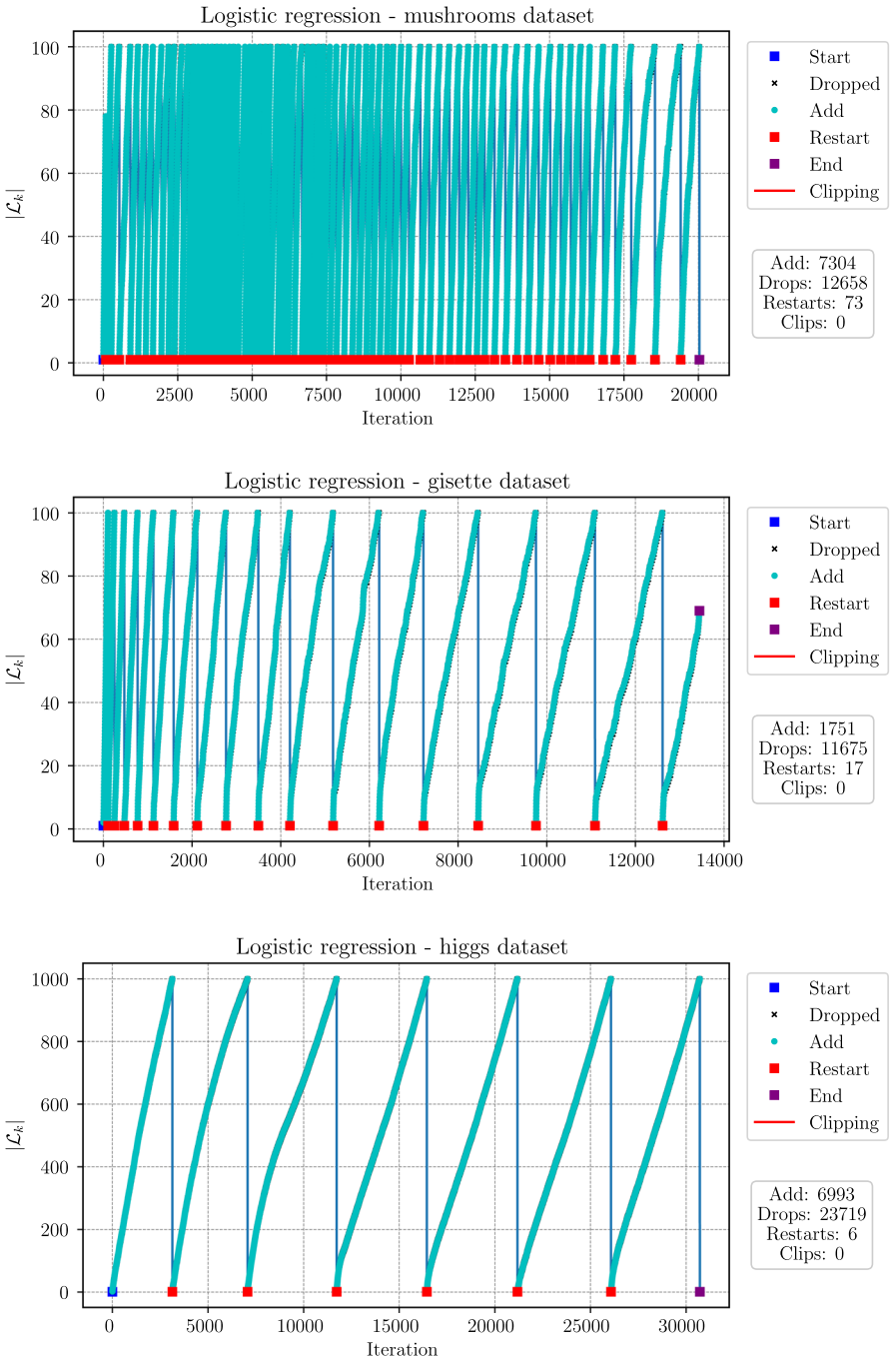


Fig. 10 Index set cardinality versus iteration for the logistic regression of the *mushrooms* dataset (top), *gisette* dataset (center), and *HIGGS* dataset (bottom). We mark the dropped iteration as black $\times$'s, the iterations kept at the index set as cyan circles, restarts as red squares, and clippings as red lines

From the results of this example, we observe that MICE performs well in problems with a reasonably large number of parameters, for instance, 5000 in the *gisette*, and finite dataset populations ranging from the thousands to the millions. One can conclude from the results obtained that SGD-MICE performs consistently better in terms of gradient sampling cost compared to the other methods. Note that both SAG and SAGA need to decrease their step-sizes as the sample-size increases, and that SARAH and SVRG need to reevaluate the full-gradient after a few epochs to keep their convergence.

6 Conclusion

We introduced the *Multi-Iteration Stochastic Optimizers*, a novel class of first-order stochastic optimization methods that use the *Multi-Iteration stochastiC Estimator* (MICE). The MICE estimator utilizes successive control variates along optimization paths, efficiently reusing previously computed gradients to achieve accurate mean gradient approximations. At each iteration, it adaptively samples gradients to satisfy a user-specified relative error tolerance. Moreover, it employs a greedy strategy to determine which iterates to retain in memory and which to discard.

Thanks to its ability to control relative gradient error, MICE facilitates robust stopping criteria based on the gradient norm. Moreover, its nonintrusive design makes it readily integrable with existing first-order optimization methods, significantly expanding its applicability.

We provided a rigorous theoretical analysis for SGD-MICE, highlighting its strong performance across different classes of optimization problems. In particular, we proved exponential convergence in the L^2 sense for gradient-dominated functions under constant step-size conditions. For strongly convex problems, our results demonstrate that SGD-MICE achieves accuracy tol with an average complexity of $\mathcal{O}(tol^{-1})$ gradient evaluations, outperforming conventional adaptive batch-size SGD methods, which require $\mathcal{O}(tol^{-1}\log(tol^{-1}))$ evaluations.

Numerically, we validated our theory through three illustrative examples, employing consistent MICE parameters across diverse scenarios. The tests ranged from a quadratic function with stochastic Hessian to a stochastic Rosenbrock problem solved via Adam-MICE, and finally logistic regression training on large-scale datasets. In the latter, SGD-MICE demonstrated competitive performance against established variance-reduction methods (SAG, SAGA, SVRG, and SARAH), confirming its practical efficiency and scalability.

Future research directions include extending MICE to quasi-Newton methods, exploring constrained optimization settings through standard techniques like projected gradients and active set methods, and investigating additional gradient estimation error sources, such as biases from numerical discretizations. Exploring MICE in more realistic optimization scenarios, such as training neural networks [50], is another natural direction.

Appendix A: Multi-iteration Stochastic Optimizers

In this section, we present the detailed algorithms for the multi-iteration stochastic optimizers using the `MICE` estimator for the mean gradient. In Algorithms 5, 6, we respectively describe the pseudocodes for `SGD-MICE` and `Adam-MICE`.

Algorithm 5 Pseudocode for `SGD-MICE` with fixed step-size. `SGD-MICE` requires an unbiased estimator of the true gradient, ∇f; a distribution from which $\boldsymbol{\theta}$ can be sampled, π; a starting point, $\boldsymbol{\xi}_0$; and a tolerance on the squared gradient norm, tol.

procedure SGD- MICE($\nabla_{\boldsymbol{\xi}} f, \pi, \boldsymbol{\xi}_0, tol$)
  $k \leftarrow 0$
  while $\left\| \nabla_{\boldsymbol{\xi}} \mathcal{F}_k^{\text{stop}} \right\|^2 > tol$ **do** ▷ $\left\| \nabla_{\boldsymbol{\xi}} \mathcal{F}_k^{\text{stop}} \right\|$ computed as in Remark 9
    Evaluate $\nabla_{\boldsymbol{\xi}} \mathcal{F}_k$ using Algorithm 4
    $\boldsymbol{\xi}_{k+1} \leftarrow \boldsymbol{\xi}_k - \eta \nabla_{\boldsymbol{\xi}} \mathcal{F}_k$
    $k \leftarrow k + 1$
  end while
  return optimum approximation $\boldsymbol{\xi}_{k^*}$
end procedure

Adapting stochastic optimization algorithms to use `MICE` is as straight-forward as substituting the gradient estimator and the stopping criterion, as can be seen in Algorithms 5 and 6.

Algorithm 6 Pseudocode for `Adam-MICE` with fixed step-size. `Adam-MICE` requires an unbiased estimator of the true gradient, ∇f; a distribution from which $\boldsymbol{\theta}$ can be sampled, π; a starting point, $\boldsymbol{\xi}_0$; and a tolerance on the squared gradient norm, tol. Moreover, `Adam-MICE` requires the constants β_1, β_2, and ϵ_{Adam}. We use the values recommended by [33], $\beta_1 = 0.9$, $\beta_2 = 0.999$, and $\epsilon_{\text{Adam}} = 10^{-8}$.

procedure ADAM- MICE($\nabla_{\boldsymbol{\xi}} f, \pi, \boldsymbol{\xi}_0, tol$)
  Initialize $\boldsymbol{m}_0$ and $\boldsymbol{v}_0$ as zero-vectors
  $k \leftarrow 0$
  while $\left\| \nabla_{\boldsymbol{\xi}} \mathcal{F}_k^{\text{stop}} \right\|^2 > tol$ **do** ▷ $\left\| \nabla_{\boldsymbol{\xi}} \mathcal{F}_k^{\text{stop}} \right\|$ computed as in Remark 9
    Evaluate $\nabla_{\boldsymbol{\xi}} \mathcal{F}_k$ using Algorithm 4
    $\boldsymbol{m}_{k+1} \leftarrow \beta_1 \boldsymbol{m}_k + (1-\beta_1) \nabla_{\boldsymbol{\xi}} \mathcal{F}_k$
    $\boldsymbol{v}_{k+1} \leftarrow \beta_2 \boldsymbol{v}_k + (1-\beta_2) \nabla_{\boldsymbol{\xi}} \mathcal{F}_k^2$ ▷ The gradient estimates are squared element-wise
    $\hat{\boldsymbol{m}}_{k+1} \leftarrow \boldsymbol{m}_{k+1} / (1-\beta_1^{k+1})$
    $\hat{\boldsymbol{v}}_{k+1} \leftarrow \boldsymbol{v}_{k+1} / (1-\beta_2^{k+1})$
    $\boldsymbol{\xi}_{k+1} \leftarrow \boldsymbol{\xi}_k - \eta \hat{\boldsymbol{m}}_{k+1} / (\sqrt{\hat{\boldsymbol{v}}_{k+1}} + \epsilon_{\text{Adam}})$
    $k \leftarrow k+1$
  end while
  return optimum approximation $\boldsymbol{\xi}_{k^*}$
end procedure

Appendix B: Variance Reduction Gradient Estimators in the Finite-Sum Case

This short appendix subsection recalls the standard gradient estimators used by `SVRG`, `SARAH`, `SAG`, and `SAGA` in the finite-sum setting and explains how `SVRG` and `SARAH` correspond to representative choices of the `MICE` index set.

Finite-Sum Objective

Consider

$$\min_{\boldsymbol{\xi} \in \mathbb{R}^{d_{\boldsymbol{\xi}}}} F(\boldsymbol{\xi}) \coloneqq \frac{1}{n} \sum_{i=1}^{n} f_i(\boldsymbol{\xi}), \tag{144}$$

with full gradient $\nabla F(\boldsymbol{\xi}) = \frac{1}{n}\sum_{i=1}^{n} \nabla f_i(\boldsymbol{\xi})$. At iteration k, let $\mathcal{B}_k \subset \{1, \ldots, n\}$ denote a mini-batch of indices.

SVRG

SVRG [23] uses a control variate anchored at a snapshot point $\boldsymbol{\xi}_{\tilde{k}}$ (updated periodically every m iterations in the classical implementation). Given a snapshot iterate $\boldsymbol{\xi}_{\tilde{k}}$ and its full gradient $\nabla F(\boldsymbol{\xi}_{\tilde{k}})$, the SVRG estimator at $\boldsymbol{\xi}_k$ is

$$v_k \coloneqq \frac{1}{|\mathcal{B}_k|}\sum_{i\in\mathcal{B}_k}\big(\nabla f_i(\boldsymbol{\xi}_k) - \nabla f_i(\boldsymbol{\xi}_{\tilde{k}})\big) + \nabla F(\boldsymbol{\xi}_{\tilde{k}}). \tag{145}$$

In MICE terms, this corresponds to the two-level index set $\mathcal{L}_k = \{\tilde{k}, k\}$: the snapshot provides an anchor (a "level" computed with a full batch), and the current iterate contributes a correction via gradient differences. Updating the snapshot every m iterations corresponds to a periodic restart of the anchor.

SARAH

SARAH [24] uses successive control variates between consecutive iterates. With an initial full-gradient computation $v_{\tilde{k}} \coloneqq \nabla F(\boldsymbol{\xi}_{\tilde{k}})$, it updates recursively

$$v_k \coloneqq \frac{1}{|\mathcal{B}_k|}\sum_{i\in\mathcal{B}_k}\big(\nabla f_i(\boldsymbol{\xi}_k) - \nabla f_i(\boldsymbol{\xi}_{k-1})\big) + v_{k-1}. \tag{146}$$

This corresponds to the add-only / consecutive-differences regime of MICE: taking $\mathcal{L}_k = \{0, 1, \ldots, k\}$ yields a telescoping sum of level differences (see Remark 1 in the main text), while MICE further allows (i) cumulative sample growth at each level and (ii) adaptive sparsification of $\mathcal{L}_k$ via Drop/Restart/Clip operators.

SAG

SAG [49] maintains a memory table of per-sample gradients. Let $\{g_i\}_{i=1}^{n}$ store the most recently computed gradients and define their running average

$$\bar{g}_k \coloneqq \frac{1}{n}\sum_{i=1}^{n} g_i. \tag{147}$$

Sampling $i_k \sim \mathrm{Unif}\{1, \ldots, n\}$, SAG updates $g_{i_k} \leftarrow \nabla f_{i_k}(\boldsymbol{\xi}_k)$ and uses $\bar{g}_k$ as a direction. Unlike SVRG and SARAH, this estimator is generally biased at finite k because the table entries correspond to gradients evaluated at different past iterates; nevertheless, the method enjoys linear convergence under standard assumptions. SAG is not

naturally represented by an iteration-indexed set $\mathcal{L}_k \subset \{0, \dots, k\}$, since its control variates are indexed by *data points*.

SAGA

SAGA [27] stores a memory table $\{g_i\}_{i=1}^n$ of the most recently computed per-sample gradients. Sampling $i_k \sim \mathrm{Unif}\{1, \dots, n\}$, its estimator is

$$v_k \coloneqq \nabla f_{i_k}(\boldsymbol{\xi}_k) - g_{i_k} + \frac{1}{n}\sum_{i=1}^{n} g_i, \qquad g_{i_k} \leftarrow \nabla f_{i_k}(\boldsymbol{\xi}_k). \tag{148}$$

As with SAG, SAGA is not directly captured by an iteration-indexed set $\mathcal{L}_k \subset \{0, \dots, k\}$ because its control variates are indexed by *data points* rather than past iterates. Nevertheless, both SAGA and MICE reduce variance by reusing previously computed gradient information: SAGA stores per-sample gradients, while MICE stores (and resamples) per-iteration gradient differences along the optimization path.

Appendix C: Error Decomposition of the MICE Estimator

The MICE estimator has a conditional bias due to the reuse of previous information. Here we prove that, if the statistical error of the estimator is controlled every iteration, then the bias is implicitly controlled as well. Recall the MICE estimator is defined as

$$\nabla \mathcal{F}_k = \sum_{\ell \in \mathcal{L}_k} \frac{1}{M_{\ell,k}} \sum_{\alpha \in \mathcal{I}_{\ell,k}} \Delta_{\ell,k,\alpha}, \tag{149}$$

where

$$\Delta_{\ell,k,\alpha} = \begin{cases} \nabla f(\boldsymbol{\xi}_\ell, \boldsymbol{\theta}_\alpha) - \nabla f(\boldsymbol{\xi}_{p_k(\ell)}, \boldsymbol{\theta}_\alpha) & \text{if } \ell > \min\{\mathcal{L}_k\} \\ \nabla f(\boldsymbol{\xi}_0, \boldsymbol{\theta}_\alpha) & \text{if } \ell = \min\{\mathcal{L}_k\}. \end{cases} \tag{150}$$

The squared L^2 error of the MICE estimator can be decomposed as

$$\mathbb{E}\left[\left\|\nabla \mathcal{F}_k - \nabla_{\boldsymbol{\xi}} F(\boldsymbol{\xi}_k)\right\|^2\right] = \underbrace{\mathbb{E}\left[\left\|\nabla \mathcal{F}_k - \mathbb{E}\left[\nabla \mathcal{F}_k \mid \left\{\boldsymbol{\xi}_{\ell'}\right\}_{\ell' \in \mathcal{L}_k}\right]\right\|^2\right]}_{\text{statistical error}} \tag{151}$$

$$+ \underbrace{\mathbb{E}\left[\left\|\mathbb{E}\left[\nabla \mathcal{F}_k \mid \left\{\boldsymbol{\xi}_{\ell'}\right\}_{\ell' \in \mathcal{L}_k}\right] - \nabla_{\boldsymbol{\xi}} F(\boldsymbol{\xi}_k)\right\|^2\right]}_{\text{bias term}}. \tag{152}$$

due to

$$\begin{aligned}&\mathbb{E}\Big[\mathbb{E}\Big[\Big\langle\nabla\mathcal{F}_k-\mathbb{E}\Big[\nabla\mathcal{F}_k\mid\{\boldsymbol{\xi}_{\ell'}\}_{\ell'\in\mathcal{L}_k}\Big],\mathbb{E}\Big[\nabla\mathcal{F}_k\mid\{\boldsymbol{\xi}_{\ell'}\}_{\ell'\in\mathcal{L}_k}\Big]-\nabla_{\boldsymbol{\xi}}F(\boldsymbol{\xi}_k)\Big\rangle\Big|\{\boldsymbol{\xi}_{\ell'}\}_{\ell'\in\mathcal{L}_k}\Big]\Big]\\&=\mathbb{E}\Bigg[\Big\langle\underbrace{\mathbb{E}\Big[\nabla\mathcal{F}_k-\mathbb{E}\Big[\nabla\mathcal{F}_k\mid\{\boldsymbol{\xi}_{\ell'}\}_{\ell'\in\mathcal{L}_k}\Big]\Big|\{\boldsymbol{\xi}_{\ell'}\}_{\ell'\in\mathcal{L}_k}\Big]}_{=0},\mathbb{E}\Big[\nabla\mathcal{F}_k\mid\{\boldsymbol{\xi}_{\ell'}\}_{\ell'\in\mathcal{L}_k}\Big]-\nabla_{\boldsymbol{\xi}}F(\boldsymbol{\xi}_k)\Big\rangle\Bigg]\end{aligned}\tag{153}$$

Before we analyze the bias and statistical errors, let us analyze the conditional expectation of the MICE estimator,

$$\mathbb{E}\Big[\nabla\mathcal{F}_k\mid\{\boldsymbol{\xi}_{\ell'}\}_{\ell'\in\mathcal{L}_k}\Big]=\sum_{\ell\in\mathcal{L}_k}\frac{1}{M_{\ell,k}}\sum_{\alpha\in\mathcal{I}_{\ell,k}}\mathbb{E}\Big[\Delta_{\ell,k,\alpha}\mid\{\boldsymbol{\xi}_{\ell'}\}_{\ell'\in\mathcal{L}_k}\Big],\tag{154}$$

and noting that, for $\tilde{k}<k$, $\Delta_{\ell,\tilde{k},\alpha}\mid\{\boldsymbol{\xi}_{\ell'}\}_{\ell'\in\mathcal{L}_k}$ is deterministic,

$$\mathbb{E}\Big[\Delta_{\ell,\tilde{k},\alpha}\Big|\{\boldsymbol{\xi}_{\ell'}\}_{\ell'\in\mathcal{L}_k}\Big]=\begin{cases}\Delta_{\ell,\tilde{k},\alpha}&\text{if }\tilde{k}<k\\\nabla F(\boldsymbol{\xi}_\ell)-\nabla F(\boldsymbol{\xi}_{p_k(\ell)})&\text{if }\tilde{k}=k.\end{cases}\tag{155}$$

Let $\mathcal{L}_k^{\cap}=\mathcal{L}_k\cap\mathcal{L}_{k-1}$. Splitting the summands in MICE between the terms computed at k and the previous ones,

$$\nabla\mathcal{F}_k=\underbrace{\Bigg\{\sum_{\ell\in\mathcal{L}_k^{\cap}}\frac{1}{M_{\ell,k}}\sum_{\alpha\in\mathcal{I}_{\ell,k-1}}\Delta_{\ell,k,\alpha}\Bigg\}}_{\text{previously computed}}+\underbrace{\Bigg\{\frac{1}{M_{k,k}}\sum_{\alpha\in\mathcal{I}_{k,k}}\Delta_{k,k,\alpha}+\sum_{\ell\in\mathcal{L}_k^{\cap}}\frac{1}{M_{\ell,k}}\sum_{\alpha\in\mathcal{I}_{\ell,k}\setminus\mathcal{I}_{\ell,k-1}}\Delta_{\ell,k,\alpha}\Bigg\}}_{\text{computed at }k},\tag{156}$$

taking the expectation conditioned on $\{\boldsymbol{\xi}_{\ell'}\}_{\ell'\in\mathcal{L}_k}$, and using $\nabla F(\boldsymbol{\xi}_{p_k(\min\{\mathcal{L}_k\})})=0$,

$$\begin{aligned}\mathbb{E}\Big[\nabla\mathcal{F}_k\mid\{\boldsymbol{\xi}_{\ell'}\}_{\ell'\in\mathcal{L}_k}\Big]&=\sum_{\ell\in\mathcal{L}_k^{\cap}}\frac{1}{M_{\ell,k}}\sum_{\alpha\in\mathcal{I}_{\ell,k-1}}\Delta_{\ell,k,\alpha}\\&\quad+\nabla F(\boldsymbol{\xi}_k)-\nabla F(\boldsymbol{\xi}_{k-1})+\sum_{\ell\in\mathcal{L}_k^{\cap}}\frac{M_{\ell,k}-M_{\ell,k-1}}{M_{\ell,k}}(\nabla F(\boldsymbol{\xi}_\ell)-\nabla F(\boldsymbol{\xi}_{p_k(\ell)}))\end{aligned}\tag{157}$$

$$\begin{aligned}&=\sum_{\ell\in\mathcal{L}_k^{\cap}}\frac{1}{M_{\ell,k}}\sum_{\alpha\in\mathcal{I}_{\ell,k-1}}\Delta_{\ell,k,\alpha}+\nabla F(\boldsymbol{\xi}_k)\\&\quad-\sum_{\ell\in\mathcal{L}_k^{\cap}}\frac{M_{\ell,k-1}}{M_{\ell,k}}(\nabla F(\boldsymbol{\xi}_\ell)-\nabla F(\boldsymbol{\xi}_{p_k(\ell)}))\end{aligned}\tag{158}$$

$$=\nabla_{\boldsymbol{\xi}}F(\boldsymbol{\xi}_k)+\sum_{\ell\in\mathcal{L}_k^{\cap}}\frac{M_{\ell,k-1}}{M_{\ell,k}}\hat{\boldsymbol{\mu}}_{\ell,k-1}-\sum_{\ell\in\mathcal{L}_k^{\cap}}\frac{M_{\ell,k-1}}{M_{\ell,k}}\boldsymbol{\mu}_{\ell,k-1},\tag{159}$$

where in (158) we used $\sum_{\ell\in\mathcal{L}_k^{\cap}}(\nabla F(\boldsymbol{\xi}_\ell) - \nabla F(\boldsymbol{\xi}_{p_k(\ell)})) = \nabla_{\boldsymbol{\xi}} F(\boldsymbol{\xi}_{k-1})$.

Next, we investigate the bias of the MICE estimator conditioned on the current iterate $\boldsymbol{\xi}_k$ and its contribution to the squared L^2 error.

Proposition 3 (Bias of the MICE estimator in expectation minimization) *Let the bias of the MICE estimator be defined as*

$$\boldsymbol{b}_k \coloneqq \mathbb{E}\left[\nabla \mathcal{F}_k \mid \left\{\boldsymbol{\xi}_{\ell'}\right\}_{\ell'\in\mathcal{L}_k}\right] - \nabla F(\boldsymbol{\xi}_k). \tag{160}$$

Then, the bias is

$$\boldsymbol{b}_k = \sum_{\ell\in\mathcal{L}_k^{\cap}} \frac{M_{\ell,k-1}}{M_{\ell,k}} \left(\hat{\boldsymbol{\mu}}_{\ell,k-1} - \boldsymbol{\mu}_{\ell,k-1}\right), \tag{161}$$

and its contribution to the squared L^2 error is

$$\mathbb{E}\left[\|\boldsymbol{b}_k\|^2\right] = \mathbb{E}\left[\sum_{\ell\in\mathcal{L}_k^{\cap}} \frac{M_{\ell,k-1}}{M_{\ell,k}^2} V_{\ell,k-1}\right]. \tag{162}$$

Proof Equation (161) follows directly in (159). Now, let's investigate $\mathbb{E}\left[\|\boldsymbol{b}_k\|^2\right]$.

$$\mathbb{E}\left[\|\boldsymbol{b}_k\|^2\right] = \mathbb{E}\left[\left\|\sum_{\ell\in\mathcal{L}_k^{\cap}} \frac{M_{\ell,k-1}}{M_{\ell,k}} \left(\hat{\boldsymbol{\mu}}_{\ell,k-1} - \boldsymbol{\mu}_{\ell,k-1}\right)\right\|^2\right] \tag{163}$$

$$= \mathbb{E}\Bigg[\sum_{\ell\in\mathcal{L}_k^{\cap}} \Bigg(\frac{M_{\ell,k-1}^2}{M_{\ell,k}^2} \mathbb{E}\left[\left\|\hat{\boldsymbol{\mu}}_{\ell,k-1} - \boldsymbol{\mu}_{\ell,k-1}\right\|^2 \,\middle|\, \left\{\boldsymbol{\xi}_{\ell'}\right\}_{\ell'\in\mathcal{L}_\ell}\right] + 2 \sum_{j\in\mathcal{L}_k^{\cap}: j>\ell} \frac{M_{\ell,k-1}}{M_{\ell,k}} \frac{M_{j,k-1}}{M_{j,k}} \mathbb{E}\left[\left\langle \hat{\boldsymbol{\mu}}_{\ell,k-1} - \boldsymbol{\mu}_{\ell,k-1}, \hat{\boldsymbol{\mu}}_{j,k-1} - \boldsymbol{\mu}_{j,k-1}\right\rangle \mid \left\{\boldsymbol{\xi}_{\ell'}\right\}_{\ell'\in\mathcal{L}_j}\right]\Bigg)\Bigg]. \tag{164}$$

Using Lemma 1 and $\mathbb{E}\left[\left\|\hat{\boldsymbol{\mu}}_{\ell,k-1} - \boldsymbol{\mu}_{\ell,k-1}\right\|^2 \,\middle|\, \left\{\boldsymbol{\xi}_{\ell'}\right\}_{\ell'\in\mathcal{L}_\ell}\right] = V_{\ell,k-1} M_{\ell,k-1}^{-1}$ concludes the proof. □

Note from (161) that $\mathbb{E}[\boldsymbol{b}_k] = \mathbf{0}$.

Corollary 9 (Bias of the MICE estimator in finite sum minimization) *The bias $\boldsymbol{b}_k$ of the MICE estimator in finite sum minimization is similar to the expectation minimization one, with the consideration of the finite population correction factor,*

$$\mathbb{E}\left[\|\boldsymbol{b}_k\|^2\right] = \mathbb{E}\left[\sum_{\ell\in\mathcal{L}_k^{\cap}} \left(\frac{N - M_{\ell,k-1}}{N}\right) \frac{M_{\ell,k-1}}{M_{\ell,k}^2} V_{\ell,k-1}\right]. \tag{165}$$

Proof The proof follows exactly as in Proposition 3, except the finite population correction factor is used in the centered second moment of $\hat{\boldsymbol{\mu}}_{\ell,k-1}$, $\mathbb{E}\left[\left\|\hat{\boldsymbol{\mu}}_{\ell,k-1}-\boldsymbol{\mu}_{\ell,k-1}\right\|^2 \,\middle|\, \{\boldsymbol{\xi}_{\ell'}\}_{\ell'\in\mathcal{L}_\ell}\right] = (N-M_{\ell,k-1})N^{-1}V_{\ell,k-1}M_{\ell,k-1}^{-1}$. □

Proposition 4 (Statistical error of the MICE estimator in expectation minimization) *The statistical error of the MICE estimator in the case of expectation minimization is*

$$\mathbb{E}\left[\left\|\nabla\mathcal{F}_k-\mathbb{E}\left[\nabla\mathcal{F}_k \mid \{\boldsymbol{\xi}_{\ell'}\}_{\ell'\in\mathcal{L}_k}\right]\right\|^2\right]=\mathbb{E}\left[\sum_{\ell\in\mathcal{L}_k^{\cap}}\frac{(M_{\ell,k}-M_{\ell,k-1})V_{\ell,k}}{M_{\ell,k}^2}\right]+\mathbb{E}\left[\frac{V_{k,k}}{M_{k,k}}\right] \tag{166}$$

Proof From (151), we can use Lemma 2 and Proposition 3 to get

$$\begin{aligned}
&\mathbb{E}\left[\left\|\nabla\mathcal{F}_k-\mathbb{E}\left[\nabla\mathcal{F}_k \mid \{\boldsymbol{\xi}_{\ell'}\}_{\ell'\in\mathcal{L}_k}\right]\right\|^2\right]\\
&\quad=\mathbb{E}\left[\left\|\nabla\mathcal{F}_k-\nabla_{\boldsymbol{\xi}}F(\boldsymbol{\xi}_k)\right\|^2\right]-\mathbb{E}\left[\left\|\mathbb{E}\left[\nabla\mathcal{F}_k \mid \{\boldsymbol{\xi}_{\ell'}\}_{\ell'\in\mathcal{L}_k}\right]-\nabla_{\boldsymbol{\xi}}F(\boldsymbol{\xi}_k)\right\|^2\right] &(167)\\
&\quad=\mathbb{E}\left[\sum_{\ell\in\mathcal{L}_k}\frac{V_{\ell,k}}{M_{\ell,k}}\right]-\mathbb{E}\left[\sum_{\ell\in\mathcal{L}_k^{\cap}}\frac{M_{\ell,k-1}}{M_{\ell,k}^2}V_{\ell,k-1}\right] &(168)\\
&\quad=\mathbb{E}\left[\sum_{\ell\in\mathcal{L}_k^{\cap}}\frac{V_{\ell,k}}{M_{\ell,k}}-\sum_{\ell\in\mathcal{L}_k^{\cap}}\frac{M_{\ell,k-1}}{M_{\ell,k}^2}V_{\ell,k-1}\right]+\mathbb{E}\left[\frac{V_{k,k}}{M_{k,k}}\right]. &(169)
\end{aligned}$$

Using $V_{\ell,k}=V_{\ell,k-1}$ for $\ell\in\mathcal{L}_k^{\cap}$ concludes the proof. □

Corollary 10 (Statistical error of the MICE estimator in finite sum minimization) *The statistical error in the finite sum minimization case is*

$$\begin{aligned}
&\mathbb{E}\left[\left\|\nabla\mathcal{F}_k-\mathbb{E}\left[\nabla\mathcal{F}_k \mid \{\boldsymbol{\xi}_{\ell'}\}_{\ell'\in\mathcal{L}_k}\right]\right\|^2\right]=\\
&\mathbb{E}\left[\sum_{\ell\in\mathcal{L}_k^{\cap}}\left(\frac{N-M_{\ell,k}}{N}\right)\frac{(M_{\ell,k}-M_{\ell,k-1})V_{\ell,k}}{M_{\ell,k}^2}\right]+\left(\frac{N-M_{k,k}}{N}\right)\mathbb{E}\left[\frac{V_{k,k}}{M_{k,k}}\right] \qquad (170)
\end{aligned}$$

Proof The proof follows exactly as in Proposition 4, except Remark 5 and Corollary 9 are used instead of Lemma 2 and Proposition 3. □

Appendix D: A High-Probability Bound for Add-Only MICE

In this appendix we give a simple high-probability control of the realized MICE gradient estimation error in an add-only regime, which is sufficient to prove Corollary 3.

Setup (add-only / fixed-anchor regime)

Recall $\nabla\mathcal{F}_k = \nabla_{\boldsymbol{\xi}}\mathcal{F}_k$ and define the realized gradient estimation error

$$e_k := \nabla\mathcal{F}_k - \nabla_{\boldsymbol{\xi}} F(\boldsymbol{\xi}_k). \tag{171}$$

We assume add-only index sets $\mathcal{L}_{k-1} \subseteq \mathcal{L}_k$ and thus the previous rule $p_k(\ell) = \ell - 1$, so that once a level ℓ is introduced it remains anchored to the fixed pair $(\boldsymbol{\xi}_\ell, \boldsymbol{\xi}_{\ell-1})$ until a restart occurs. In this regime, each level contribution is a sample mean of i.i.d. centered increments at that anchored pair. Moreover, conditional on the iterate history, the sample sets used at distinct levels are independent by construction, so the level contributions are conditionally independent. We work under the tail assumption in Assumption 6.

Lemma 3 (A single good event for MICE error control) *Assume Assumption 6 holds for every level. Assume that, conditional on the iterate history, the sample sets used at distinct levels are independent. Fix a summable schedule $(\delta_k)_{k\geq 0}$ with $\sum_{k\geq 0}\delta_k \leq \delta$. Assume that at each iteration k, the sample sizes $\{M_{\ell,k}\}_{\ell\in\mathcal{L}_k}$ are chosen based only on the past (in particular, before drawing any new samples at iteration k). Define the event $\Omega_\delta^{\mathrm{MICE}}$ by requiring that, for all $k \geq 0$,*

$$\|e_k\| \leq \sqrt{2d_\xi \,\log\Big(\frac{2d_\xi}{\delta_k}\Big)}\left(\sum_{\ell\in\mathcal{L}_k}\frac{\sigma_\ell^2}{M_{\ell,k}}\right)^{1/2}. \tag{172}$$

Then $\mathbb{P}[\Omega_\delta^{\mathrm{MICE}}] \geq 1-\delta$.

Proof Fix $k \geq 0$ and a coordinate $j \in \{1,\dots,d_\xi\}$. Condition on the iterate history and on the chosen sample sizes $\{M_{\ell,k}\}_{\ell\in\mathcal{L}_k}$, so that the sample sizes can be treated as fixed when applying tail bounds to the samples. In the add-only regime,

$$(e_k)_j = \sum_{\ell\in\mathcal{L}_k}\frac{1}{M_{\ell,k}}\sum_{i=1}^{M_{\ell,k}}(Z_{\ell,i})_j. \tag{173}$$

By Assumption 6, each coordinatewise sample mean is sub-Gaussian with proxy variance $\sigma_\ell^2/M_{\ell,k}$, and by independence across levels the sum $(e_k)_j$ is sub-Gaussian with proxy variance $v_k := \sum_{\ell\in\mathcal{L}_k}\sigma_\ell^2/M_{\ell,k}$. Therefore, for all $t > 0$,

$$\mathbb{P}\big[|(e_k)_j| \geq t\big] \leq 2\exp\left(-\frac{t^2}{2v_k}\right). \tag{174}$$

Taking $t = \sqrt{2v_k \log(2d_\xi/\delta_k)}$ gives $\mathbb{P}[|(e_k)_j| \geq t] \leq \delta_k/d_\xi$. A union bound over $j = 1, \ldots, d_\xi$ yields that with probability at least $1-\delta_k$, $\|e_k\|_\infty \leq \sqrt{2v_k \log(2d_\xi/\delta_k)}$. Since $\|e_k\| \leq \sqrt{d_\xi}\ \|e_k\|_\infty$, the bound (172) holds with probability at least $1-\delta_k$. Finally, a union bound over $k \geq 0$ and $\sum_{k\geq 0} \delta_k \leq \delta$ yields $\mathbb{P}[\Omega_\delta^{\mathrm{MICE}}] \geq 1-\delta$. □

Corollary 11 (Uniform relative error control under the variance-sum constraint) *Under the conditions of Lemma 3, if the sample sizes satisfy (63) for all $k \geq 0$, then with probability at least $1-\delta$,*

$$\|e_k\| \leq \epsilon \left\|\nabla_\xi F(\xi_k)\right\|, \qquad \forall\, k \geq 0. \tag{175}$$

Consequently, Theorem 1 applies and (61) holds for all $k \geq 0$ with probability at least $1-\delta$.

Lemma 4 (High-probability iteration bound to reach tol) *Assume the conditions of Corollary 3. In addition, assume F satisfies Assumption 3. Let $r_{\mathrm{hp}} \in (0,1)$ be the contraction factor in (61) and define*

$$K(tol) := \left\lceil \frac{\log\big(2L(F(\xi_0) - F(\xi^*))/tol\big)}{\log(1/r_{\mathrm{hp}})} \right\rceil. \tag{176}$$

Then, with probability at least $1-\delta$, we have $k^*(tol) \leq K(tol)$, where $k^*(tol)$ is defined in (68).

Proof By Corollary 3, with probability at least $1-\delta$ we have for all $k \geq 0$,

$$F(\xi_k) - F(\xi^*) \leq r_{\mathrm{hp}}^k \big(F(\xi_0) - F(\xi^*)\big). \tag{177}$$

Since F is convex and L-smooth, (55) holds (as used earlier in Corollary 1). Therefore, on the same event,

$$\|\nabla F(\xi_k)\|^2 \leq 2L\, r_{\mathrm{hp}}^k \big(F(\xi_0) - F(\xi^*)\big). \tag{178}$$

By the definition of $K(tol)$, for all $k \geq K(tol)$ we have $2L\, r_{\mathrm{hp}}^k (F(\xi_0) - F(\xi^*)) \leq tol$, hence $\|\nabla F(\xi_k)\|^2 \leq tol$ and therefore $k^*(tol) \leq K(tol)$. □

Corollary 12 (High-probability sampling-cost bound under variance-sum sizing) *Assume the conditions of Corollary 3 and Lemma 4. In particular, work in the add-only regime. Fix a summable schedule $(\delta_k)_{k\geq 0}$ with $\sum_{k\geq 0} \delta_k \leq \delta$ and assume it is nonincreasing. For each iteration k, define*

$$R_k := \frac{\epsilon^2}{2d_\xi\, \log(2d_\xi/\delta_k)}\ \left\|\nabla F(\xi_k)\right\|^2. \tag{179}$$

Assume the sample sizes are chosen (predictably) to satisfy the variance-sum constraint

$$\sum_{\ell \in \mathcal{L}_k} \frac{\sigma_\ell^2}{M_{\ell,k}} \leq R_k, \qquad \forall\, k \geq 0, \tag{180}$$

and that, at each iteration k, they are (approximately) work-optimal for the continuous relaxation

$$\min_{\{M_{\ell,k}>0\}} \sum_{\ell\in\mathcal{L}_k} (1+\mathbb{1}_{\underline{\mathcal{L}_k}}(\ell))\, M_{\ell,k} \quad s.t. \quad \sum_{\ell\in\mathcal{L}_k} \frac{\sigma_\ell^2}{M_{\ell,k}} \le R_k, \tag{181}$$

whose KKT optimizer is

$$M^\star_{\ell,k} = \frac{\sigma_\ell}{\sqrt{(1+\mathbb{1}_{\underline{\mathcal{L}_k}}(\ell))}} \frac{\sum_{j\in\mathcal{L}_k} \sigma_j \sqrt{(1+\mathbb{1}_{\underline{\mathcal{L}_k}}(j))}}{R_k}, \qquad \ell\in\mathcal{L}_k. \tag{182}$$

In the add-only regime with cumulative sampling, the cumulative gradient-evaluation cost $\mathcal{C}_k$ (defined as the telescoping sum of increments in (37)) coincides with the cost of the final cumulative sample sizes at iteration k, i.e., $\mathcal{C}_k = \sum_{\ell\in\mathcal{L}_k}(1+\mathbb{1}_{\underline{\mathcal{L}_k}}(\ell))\, M_{\ell,k}$. Then, with probability at least $1-\delta$,

$$\mathcal{C}_{k^*(tol)-1} \le \frac{2d_\xi\ \log(2d_\xi/\delta_{K(tol)-1})}{\epsilon^2\, tol} \left(\sum_{\ell\in\mathcal{L}_{K(tol)-1}} \sigma_\ell \sqrt{(1+\mathbb{1}_{\underline{\mathcal{L}_k}}(\ell))} \right)^2 + \sum_{\ell\in\mathcal{L}_{K(tol)-1}} (1+\mathbb{1}_{\underline{\mathcal{L}_k}}(\ell)), \tag{183}$$

where $K(tol)$ is defined in (176).

Proof Under the KKT sizing (182), the corresponding (continuous) sampling cost satisfies

$$\sum_{\ell\in\mathcal{L}_k} (1+\mathbb{1}_{\underline{\mathcal{L}_k}}(\ell))\, M^\star_{\ell,k} = \frac{\left(\sum_{\ell\in\mathcal{L}_k} \sigma_\ell \sqrt{(1+\mathbb{1}_{\underline{\mathcal{L}_k}}(\ell))}\right)^2}{R_k}. \tag{184}$$

Rounding to integers by $M_{\ell,k} = \lceil M^\star_{\ell,k} \rceil$ increases the left-hand side by at most $\sum_{\ell\in\mathcal{L}_k}(1+\mathbb{1}_{\underline{\mathcal{L}_k}}(\ell))$. On the event of Lemma 4 we have $k^*(tol) \le K(tol)$. At the last pre-stopping iterate $k = k^*(tol) - 1$ we have $\|\nabla F(\boldsymbol{\xi}_k)\|^2 > tol$, hence from (179),

$$\frac{1}{R_k} \le \frac{2d_\xi\ \log(2d_\xi/\delta_k)}{\epsilon^2\, tol} \le \frac{2d_\xi\ \log(2d_\xi/\delta_{K(tol)-1})}{\epsilon^2\, tol}, \tag{185}$$

where the last inequality uses that (δ_k) is nonincreasing and $k \le K(tol) - 1$. Finally, using add-only monotonicity $\mathcal{L}_{k^*(tol)-1} \subseteq \mathcal{L}_{K(tol)-1}$ yields (183). □

Acknowledgements Last but not least, we want to thank Dr. Sören Wolfers and Prof. Jesper Oppelstrup. They both provided us with valuable ideas and constructive comments.

Funding Open Access funding enabled and organized by Projekt DEAL. This publication is based upon work supported by the King Abdullah University of Science and Technology (KAUST) Office of Sponsored Research (OSR) under Award No. OSR-2019-CRG8-4033, the Alexander von Humboldt Foundation, and Coordination for the Improvement of Higher Education Personnel (CAPES).

Data Availability The data that support the findings of this study are available from the corresponding author upon reasonable request. The python implementation of `MICE` and the baseline methods used to generate the data and figures in this work are available at GitHub https://github.com/agcarlon/mice. Moreover, `MICE` can be installed using PyPi https://pypi.org/project/mice/. We use the datasets *mushrooms*, *gisette*, and *Higgs*, obtained from LibSVM https://www.csie.ntu.edu.tw/\protect\unhbox\voidb@x\penalty\@M\cjlin/libsvmtools/datasets/binary.html. The exact configurations and random seeds used to generate all figures are provided in the repository.

Declarations

Conflict of interest The authors have no conflicts to disclose.

Open Access This article is licensed under a Creative Commons Attribution 4.0 International License, which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons licence, and indicate if changes were made. The images or other third party material in this article are included in the article's Creative Commons licence, unless indicated otherwise in a credit line to the material. If material is not included in the article's Creative Commons licence and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder. To view a copy of this licence, visit http://creativecommons.org/licenses/by/4.0/.

References

1. Marti, K.: Stochastic optimization methods. In: Stochastic Optimization Methods: Applications in Engineering and Operations Research, pp. 1–35. Springer, Berlin (2015)
2. Uryasev, S., Pardalos, P.M.: Stochastic Optimization: Algorithms And Applications, vol. 54. Springer, Berlin (2013)
3. Birge, J.R., Louveaux, F.: Introduction To Stochastic Programming, 2nd edn. Springer, Berlin (2011)
4. Lan, G.: First-order and Stochastic Optimization Methods for Machine Learning. Springer, Berlin (2020)
5. Wallace, S.W., Ziemba, W.T.: Applications of Stochastic Programming. SIAM, Philadelphia (2005)
6. Fleming, W.H., R, R.W.: Deterministic and Stochastic Optimal Control, vol. 1. Springer, Berlin (2012)
7. Ziemba, W.T., Vickson, R.G.: Stochastic Optimization Models in Finance. Academic Press, New York (2014)
8. Conejo, A.J., Carrión, M., Morales, J.M.: Decision Making Under Uncertainty in Electricity Markets, vol. 1. Springer, Berlin (2010)
9. Bayer, C., Tempone, R., Wolfers, S.: Pricing American options by exercise rate optimization. Quant. Finance **20**(11), 1749–1760 (2020). https://doi.org/10.1080/14697688.2020.1750678
10. Chang, F.-R.: Stochastic Optimization in Continuous Time. Cambridge University Press, Berlin (2004)
11. Azcue, P., Muler, N.: Stochastic Optimization in Insurance. SpringerBriefs in Quantitative Finance, p. 146. Springer, New York (2014). https://doi.org/10.1007/978-1-4939-0995-7 . (**A dynamic programming approach**)
12. Ding, Z.: Stochastic Optimization and Its Application to Communication Networks and the Smart Grid. University of Florida Digital Collections, University of Florida, Gainesville (2012)
13. Yao, D.D., Zhang, H., Zhou, X.Y. (eds.): Stochastic Modeling and Optimization, p. 468. Springer, New York (2003). https://doi.org/10.1007/978-0-387-21757-4. With applications in queues, finance, and supply chains
14. Ryan, E.G., Drovandi, C.C., McGree, J.M., Pettitt, A.N.: A review of modern computational algorithms for Bayesian optimal design. Int. Stat. Rev. **84**(1), 128–154 (2016)
15. Carlon, A.G., Dia, B.M., Espath, L., Lopez, R.H., Tempone, R.: Nesterov-aided stochastic gradient methods using Laplace approximation for Bayesian design optimization. Comput. Methods Appl. Mech. Eng. (2020). https://doi.org/10.1016/j.cma.2020.112909
16. Heinrich, S.: The multilevel method of dependent tests. In: Advances in Stochastic Simulation Methods, pp. 47–61. Springer, Berlin (2000)

17. Giles, M.B.: Multilevel Monte Carlo path simulation. Oper. Res. **56**(3), 607–617 (2008)
18. Ruppert, D.: Efficient estimations from a slowly convergent Robbins–Monro process. Technical report. Cornell University Operations Research and Industrial Engineering (1988)
19. Byrd, R.H., Chin, G.M., Nocedal, J., Wu, Y.: Sample size selection in optimization methods for machine learning. Math. Program. **134**(1), 127–155 (2012)
20. Balles, L., Romero, J., Hennig, P.: Coupling adaptive batch sizes with learning rates. arXiv preprint (2016)
21. Bollapragada, R., Byrd, R., Nocedal, J.: Adaptive sampling strategies for stochastic optimization. SIAM J. Optim. **28**(4), 3312–3343 (2018)
22. Vaswani, S., Bach, F., Schmidt, M.: Fast and faster convergence of SGD for over-parameterized models and an accelerated perceptron. In: Proceedings of the 22nd International Conference on Artificial Intelligence and Statistics. Proceedings of Machine Learning Research, vol. 89, pp. 1195–1204. PMLR, Naha (2019)
23. Johnson, R., Zhang, T.: Accelerating stochastic gradient descent using predictive variance reduction. In: Advances In Neural Information Processing Systems, pp. 315–323 (2013)
24. Nguyen, L.M., Liu, J., Scheinberg, K., Takáč, M.: Sarah: a novel method for machine learning problems using stochastic recursive gradient. In: Proceedings of the 34th International Conference on Machine Learning, vol. 70, Sydney, pp. 2613–2621 (2017). JMLR.org
25. Nguyen, L., Scheinberg, K., Takáč, M.: Inexact Sarah algorithm for stochastic optimization. Optim. Methods Softw. (2020). https://doi.org/10.1080/10556788.2020.1818081
26. Fang, C., Li, C.J., Lin, Z., Zhang, T.: Spider: Near-optimal non-convex optimization via stochastic path-integrated differential estimator. In: Advances in Neural Information Processing Systems, Montreal, pp. 689–699 (2018)
27. Defazio, A., Bach, F., Lacoste-Julien, S.: Saga: a fast incremental gradient method with support for non-strongly convex composite objectives. In: Advances In Neural Information Processing Systems, pp. 1646–1654 (2014)
28. Friedlander, M.P., Schmidt, M.: Hybrid deterministic-stochastic methods for data fitting. SIAM J. Sci. Comput. **34**(3), 1380–1405 (2012)
29. De, S., Yadav, A., Jacobs, D., Goldstein, T.: Big Batch SGD: automated inference using adaptive batch sizes. arXiv preprint (2016)
30. Ji, K., Wang, Z., Zhou, Y., Liang, Y.: Faster stochastic algorithms via history-gradient aided batch size adaptation. arXiv preprint (2019)
31. Polyak, B.T.: Introduction to Optimization. Optimization Software, p. 1. Publications Division, New York (1987)
32. Robbins, H., Monro, S.: A stochastic approximation method. Ann. Math. Stat. **22**, 400–407 (1951)
33. Kingma, D.P., Ba, J.: Adam: a method for stochastic optimization. arXiv preprint (2014)
34. Spall, J.C.: Introduction to Stochastic Search and Optimization: Estimation, Simulation, and Control, vol. 65. Wiley, Hoboken (2005)
35. Shapiro, A., Dentcheva, D., Ruszczyński, A.: Lectures on Stochastic Programming: Modeling and Theory. SIAM, Philadelphia (2014)
36. Nitanda, A.: Stochastic proximal gradient descent with acceleration techniques. In: Advances in Neural Information Processing Systems, pp. 1574–1582 (2014)
37. Konečnỳ, J., Liu, J., Richtárik, P., Takáč, M.: Mini-batch semi-stochastic gradient descent in the proximal setting. IEEE J. Sel. Top. Signal Process **10**(2), 242–255 (2016)
38. Heinrich, S.: Multilevel Monte Carlo methods. In: Large-scale Scientific Computing. Lecture Notes in Computer Science, vol. 2179, pp. 58–67. Springer, Berlin (2001)
39. Giles, M.B.: Multilevel monte carlo methods. Acta Numer **24**, 259–328 (2015)
40. Dereich, S., Müller-Gronbach, T.: General multilevel adaptations for stochastic approximation algorithms of Robbins–Monro and Polyak–Ruppert type. Numer. Math. **142**(2), 279–328 (2019). https://doi.org/10.1007/s00211-019-01024-y
41. Yang, S., Wang, M., Fang, E.X.: Multilevel stochastic gradient methods for nested composition optimization. SIAM J. Optim. **29**(1), 616–659 (2019)
42. Davison, A.C., Hinkley, D.V.: Bootstrap Methods and Their Application, vol. 1. Cambridge University Press, Cambridge (1997)
43. Karimi, H., Nutini, J., Schmidt, M.: Linear convergence of gradient and proximal-gradient methods under the polyak-łojasiewicz condition. In: Joint European Conference On Machine Learning And Knowledge Discovery In Databases, pp. 795–811. Springer (2016)

44. Nesterov, Y.: Lectures on Convex Optimization, vol. 137. Springer, Berlin (2018)
45. Efron, B.: The Jackknife, the Bootstrap, and Other Resampling Plans, vol. 38. Siam, Philadelphia (1982)
46. Dvoretzky, A., Kiefer, J., Wolfowitz, J.: Asymptotic minimax character of the sample distribution function and of the classical multinomial estimator. Ann. Math. Stat. **27**, 642–669 (1956)
47. Massart, P.: The Tight Constant in the Dvoretzky–Kiefer–Wolfowitz inequality. Ann. Probab. **18**, 1269–1283 (1990)
48. Welford, B.P.: Note on a method for calculating corrected sums of squares and products. Technometrics **4**(3), 419–420 (1962)
49. Schmidt, M., Le Roux, N., Bach, F.: Minimizing finite sums with the stochastic average gradient. Math. Program. **162**(1–2), 83–112 (2017)
50. Xu, P., Roosta-Khorasani, F., Mahoney, M.W.: Second-order optimization for non-convex machine learning: an empirical study. In: Proceedings of the 2020 Siam International Conference on Data Mining, pp. 199–207. SIAM, Cincinnati (2020)

Publisher's Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.