Karlsruhe Institute of Technology

wave phenomena
analysis and numerics

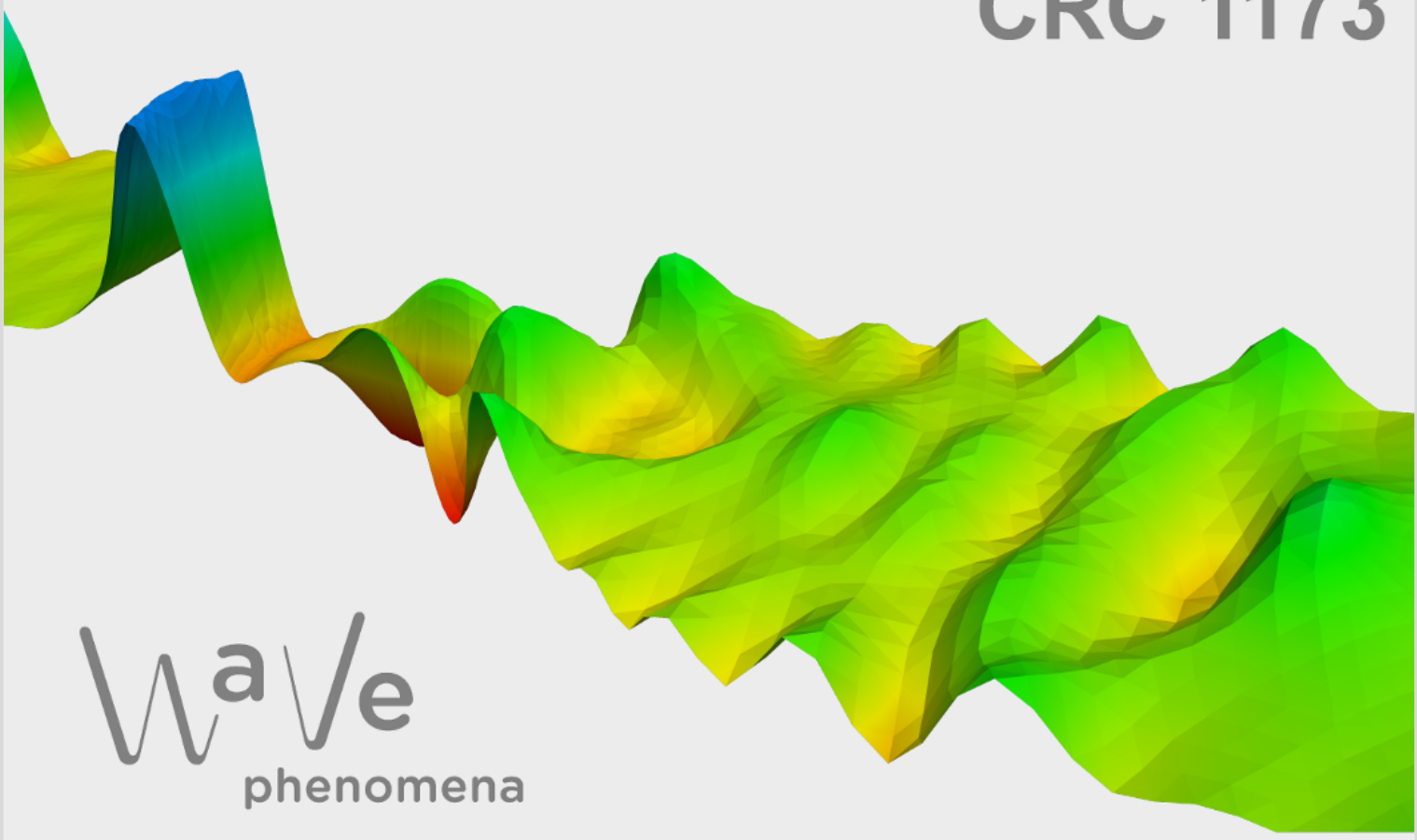
Multilevel stochastic gradient descent for optimal control under uncertainty

Niklas Baumgarten, David Schneiderhan

CRC Preprint 2025/28, June 2025

KARLSRUHE INSTITUTE OF TECHNOLOGY

CRC 1173



Participating universities

Universität Stuttgart

EBERHARD KARLS
UNIVERSITÄT
TÜBINGEN

Funded by

ISSN 2365-662X

MULTILEVEL STOCHASTIC GRADIENT DESCENT FOR OPTIMAL CONTROL UNDER UNCERTAINTY

NIKLAS BAUMGARTEN AND DAVID SCHNEIDERHAN

Abstract. We present a multilevel stochastic gradient descent method for the optimal control of systems governed by partial differential equations under uncertain input data. The gradient descent method used to find the optimal control leverages a parallel multilevel Monte Carlo method as stochastic gradient estimator. As a result, we achieve precise control over the stochastic gradient's bias, introduced by numerical approximation, and its sampling error, arising from the use of incomplete gradients, while optimally managing computational resources. We show that the method exhibits linear convergence in the number of optimization steps while avoiding the cost of computing the full gradient at the highest fidelity. Numerical experiments demonstrate that the method significantly outperforms the standard (mini-) batched stochastic gradient descent method in terms of convergence speed and accuracy. The method is particularly well-suited for high-dimensional control problems, taking advantage of parallel computing resources and a distributed multilevel data structure. Additionally, we evaluate and implement different step size strategies, optimizer schemes, and budgeting techniques. The method's performance is studied using a two-dimensional elliptic subsurface diffusion problem with log-normal coefficients and Matérn covariance.

1 Introduction

The state of a physical, technological, or economical process, as a function of space and time, is often described by partial differential equations (PDEs). Controlling the state of such processes, for example, by imposing boundary conditions or external forces, is of significant interest in all the aforementioned applications. However, determining the optimal control of a PDE-governed system is a computationally demanding task, particularly when the PDE involves high-dimensional, uncertain input data and the control has to be found with high precision.

Known methods to uncertain optimal control problems (OCPs) focus on minimizing the expected distance between the state and a desired target. Finding solutions to such problems often requires three key functionalities: solving the minimization problem, e.g. by using stochastic gradient descent (SGD) methods; addressing the uncertainty through sufficient sampling of the input data; and discretizing the PDEs with finite element (FE) methods. Approaches which fall under this description can be found in [1, 2, 3, 4, 5].

Motivated by this work, we take an integrated approach, combining all three functionalities in a single algorithm which leverages multilevel variance reduction, as in multilevel Monte Carlo (MLMC) methods [6, 7], and parallel computing resources. Even though other sampling methods to discretize the input space involve sparse grids [8, 9] and quasi-Monte Carlo methods [10, 11], we based our approach on the budgeted multilevel Monte Carlo (BMLMC) method [12, 13, 14], which provides high-performance and broad applicability. As baseline, we consider a step size controlled parallel (mini-) batched stochastic gradient descent (BSGD) method, e.g. used in [15, 16], and show that our method improves it significantly in terms of convergence speed, achievable accuracy, scalability and robustness (cf. Figure 5 for a direct comparison).

Date: June 23, 2025.

niklas.baumgarten@uni-heidelberg.de
david.schneiderhan@kit.edu

As a model problem, we consider a two-dimensional elliptic subsurface diffusion problem with log-normal coefficients (illustrations in Figure 1), generated using the memory efficient stochastic PDE sampling technique of [17]. We intentionally choose a high-dimensional elliptic problem to not lay the focus on the PDE, but rather on the algorithm and its properties. To get the gradients for the SGD, BSGD or the here introduced multilevel stochastic gradient descent (MLSGD) method, we solve the adjoint system corresponding to the PDE constraint as outlined in [1], also related to the adjoint Monte Carlo method described in [18].

Previous multilevel ideas for OCPs can be found in [19] for pathwise control, or in [20] for sample average approximation (SAA) assembled into a large linear system — similar to [12] — then solved using a multigrid algorithm. The results in [21, 22] provide a foundation for multilevel gradient estimation of OCPs, e.g., then used in a nonlinear conjugate gradient method. The approaches in [3, 4] address the optimization problem using stochastic approximation (SA), tracing back to [23], in the form of a SGD method. Our method embeds the mentioned multilevel gradient estimation in this SGD approach. In [24], the SA is also extended to a multilevel setting, considering [25], from which we draw further motivation to develop the MLSGD proposed with this paper. Related ideas can also be found in the context of Bayesian inverse problems, where the Stein variation gradient descent has also been extended by multilevel ideas [26, 27].

Machine learning has been the main driver for the development of novel SGD methods to enable large-scale training of neural networks, often striking the balance between per-iteration cost and expected improvements [28]. Features such as variance (noise) reduction and parallelism through BSGD [29], adaptive moment estimation (ADAM) [30], adaptive step sizes [31], or averaging and aggregation schemes [32, 33], have been successfully applied to various problems. We take inspiration in these approaches, incorporating them into the proposed MLSGD method for OCPs and conjecture that the algorithm is applicable in machine learning as well.

Concluded, recent developments, theory and existing algorithms motivate the combination of multilevel variance reduction and SGD methods, however, we have not found a highly-performant, adaptive, and parallel method for high-dimensional control problems yet. With this paper, we propose new ways to realize such a method which in particular features:

Algorithmic description and convergence analysis. Building on the assumptions and notations in Section 2, we present a detailed description of the MLSGD method for OCPs in Algorithm 1 and Algorithm 2, incorporating the standard BSGD method recalled in Section 3 and the MLMC estimation from Section 4. In Section 5, we establish the linear convergence of the proposed method in terms of optimization steps through Theorem 5.2 and Corollary 5.4. This result follows from bounding the error of the gradient estimation through an appropriate choice of the multilevel batch size (see Lemma 5.1). The method achieves this convergence rate (like an SAA approach) with a complexity similar to the standard MLMC method, as it circumvents full gradient evaluations (as done for SA) at the highest level. As a result, MLSGD improves convergence rates, speed, and accuracy by adaptive the multilevel batches.

Adaptivity, budgeting and error control by resources. In extension to the new MLSGD method presented in Section 5, we incorporate the adaptive step size rule from [31], adaptive multilevel batches similar to [22], the optimal distribution of the computational load as in [14], and a posteriori error control with dynamic programming [13] into Algorithm 3. We impose the given computational resources, such as the total available memory and the reserved CPU-time budget, as additional constraints to the optimization problem resulting in 6.1. The final Budgeted Multilevel Stochastic Gradient Descent (BMLSGD) method allows for total error control by the given computational resources through Corollary 6.2 and is presented in Algorithm 3.

Numerical experiments with HPC resources. Lastly, we note that the method is designed for HPC resources implemented on a domain- and sample-distributed multi-index data structure [14], enabling efficient use of resources as in [13]. While our study focuses on an elliptic model problem, the algorithm and implementation are designed for broader applicability, demonstrating parallel scalability through the usage of the FE software M++ [34], here used in version [35]. In our numerical experiments, presented across several sections (see Sections 3.2, 5.2 and 6.2), we compare all introduced algorithms (see Sections 3.1, 5.1, 6.1) to support the theoretical findings mentioned in the previous paragraphs. We particularly highlight again Figure 5 illustrating the superior performance of BMLSGD over BSGD, and Figure 7 which demonstrates that this method also scales well with increased computational resources.

2 Assumptions and Notations

Goal of the proposed methodology is to find the optimal control to a PDE governed system with uncertain input data. Similar problems are considered for example in [3, 21, 4], presenting theoretical foundation for our approach; however, the algorithm and the notation are closely related to [12, 13, 14]. The PDE system of interest is defined on a bounded polygonal spatial domain $\mathcal{D} \subset \mathbb{R}^d$ with $d \in \{1, 2, 3\}$, while the uncertainty of the input data is captured by a complete probability space $(\Omega, \mathcal{F}, \mathbb{P})$. Let further $(V, \langle\cdot,\cdot\rangle_V), (W, \langle\cdot,\cdot\rangle_W)$ denote Hilbert spaces with $V \subseteq W \subseteq \mathrm{L}^2(\mathcal{D})$, where V is an appropriate space for an PDE sample solution. Lastly, let $\mathrm{L}^2(\Omega, V)$ and $\mathrm{L}^2(\Omega, W)$ denote Bochner spaces containing all L^2-integrable maps from the probability space $(\Omega, \mathcal{F}, \mathbb{P})$ to V and W, respectively.

2.1 Optimization Problem

We search for an admissible control $\mathbf{z} \in Z$ to an optimal control problem (OCP), where Z is a non-empty, closed and convex set

$$Z \coloneqq \left\{\mathbf{z} \in W \colon \mathbf{z}_{\mathrm{ad}}^{\mathrm{low}}(\mathbf{x}) \leq \mathbf{z}(\mathbf{x}) \leq \mathbf{z}_{\mathrm{ad}}^{\mathrm{up}}(\mathbf{x}),\ \text{a.e. } \mathbf{x} \in \mathcal{D}\right\}$$

with $\mathbf{z}_{\mathrm{ad}}^{\mathrm{low}}, \mathbf{z}_{\mathrm{ad}}^{\mathrm{up}} \in W$. The control is found, if the distance of some prescribed target $\mathbf{d} \in W$ to the state solution $\mathbf{u} \in \mathrm{L}^2(\Omega, V)$ of the PDE is minimal in expectation.

Problem 2.1 (Optimal Control under Uncertainty)**.** *Given the desired target state $\mathbf{d} \in W$ and a cost factor $\lambda \geq 0$, find the optimal, admissible and deterministic control $\mathbf{z} \in Z$, such that*

$$\min_{\mathbf{z}\in Z} J(\mathbf{z}) \coloneqq \mathbb{E}\left[j(\cdot, \mathbf{z})\right] \quad \textit{with} \quad j(\omega, \mathbf{z}) \coloneqq \tfrac{1}{2}\left\|\mathbf{u}[\omega] - \mathbf{d}\right\|_W^2 + \tfrac{\lambda}{2}\left\|\mathbf{z}\right\|_W^2 \tag{2.1}$$

under the constraint that the state $\mathbf{u} \in \mathrm{L}^2(\Omega, V)$ is the solution of

$$\mathcal{G}[\omega]\,\mathbf{u}(\omega, \mathbf{x}) = \mathbf{z}(\mathbf{x}) \tag{2.2}$$

with $\mathcal{G}[\omega]$ representing the uncertain PDE system.

To ensure a unique solution to Problem 2.1 and to show convergence of the used SGD methods, we suppose the following conditions on the optimization problem.

Assumption 2.2.

(1) *The functional $j \colon \Omega \times W \to \mathbb{R}$ is L^2-Fréchet differentiable on W, i.e. for every open $\mathcal{Z} \subset W$ containing $\mathbf{z}$, there exists a linear operator $\mathcal{A} \colon \Omega \times \mathcal{Z} \to \mathcal{L}(W, \mathbb{R})$, such that*

$$\lim_{W \ni \mathbf{h} \to 0} \frac{\left\|j(\cdot, \mathbf{z}+\mathbf{h}) - j(\cdot, \mathbf{z}) + \mathcal{A}(\cdot, \mathbf{z})\mathbf{h}\right\|_{\mathrm{L}^2(\Omega)}}{\left\|\mathbf{h}\right\|_W} = 0.$$

(2) *The mapping* $\mathbf{z} \mapsto j(\omega, \mathbf{z})$ *is strongly* μ*-convex on the admissible set* Z *for almost every* $\omega \in \Omega$*, implying* $\mathbf{z} \mapsto J(\mathbf{z})$ *being strongly* μ*-convex, too, i.e., there exists a constant* $\mu > 0$*, such that for all* $\mathbf{z}^{(1)}, \mathbf{z}^{(2)} \in Z$*,*

$$\big\langle \nabla J[\mathbf{z}^{(1)}] - \nabla J[\mathbf{z}^{(2)}],\, \mathbf{z}^{(1)} - \mathbf{z}^{(2)} \big\rangle_W \geq \mu \big\| \mathbf{z}^{(1)} - \mathbf{z}^{(2)} \big\|_W^2 \quad \textit{and} \tag{2.3}$$

$$J(\mathbf{z}^{(2)}) \geq J(\mathbf{z}^{(1)}) + \big\langle \nabla J[\mathbf{z}^{(1)}],\, \mathbf{z}^{(2)} - \mathbf{z}^{(1)} \big\rangle_W + \tfrac{\mu}{2} \big\| \mathbf{z}^{(2)} - \mathbf{z}^{(1)} \big\|_W^2. \tag{2.4}$$

(3) ∇J *is Lipschitz continuous, i.e., there exists a constant* $c_{\mathrm{Lip}} \geq 0$*, such that*

$$\big\| \nabla J[\mathbf{z}^{(1)}] - \nabla J[\mathbf{z}^{(2)}] \big\|_W \leq c_{\mathrm{Lip}} \big\| \mathbf{z}^{(1)} - \mathbf{z}^{(2)} \big\|_W, \quad \forall\, \mathbf{z}^{(1)}, \mathbf{z}^{(2)} \in Z. \tag{2.5}$$

Under Assumption 2.2, the solution $\mathbf{z}^* \in Z$ to Problem 2.1 satisfies besides (2.2) also the optimality condition (cf. [36])

$$\nabla J[\mathbf{z}^*](\mathbf{x}) = 0 \quad \text{with} \quad \nabla J[\mathbf{z}](\mathbf{x}) = \lambda \mathbf{z}(\mathbf{x}) - \mathbb{E}[\mathbf{q}](\mathbf{x}) \tag{2.6}$$

everywhere in $\mathcal{D}$, where $\mathbf{q} \in \mathrm{L}^2(\Omega, V)$ is the solution of the adjoint PDE system

$$\mathcal{G}^*[\omega]\, \mathbf{q}(\omega, \mathbf{x}) = \mathbf{d}(\mathbf{x}) - \mathbf{u}(\omega, \mathbf{x}) \tag{2.7}$$

and $\mathcal{G}^*[\omega]$ is the uncertain adjoint system of $\mathcal{G}[\omega]$.

2.2 Approximation

We want to find approximate solutions to Problem 2.1, which involves three main components: (i) computing finite element (FE) solutions $\mathbf{u}_\ell \in V_\ell$ for system (2.2) and $\mathbf{q}_\ell \in V_\ell$ for system (2.7), V_ℓ denoting a suitable finite element space associated with V at discretization level ℓ; (ii) sampling finite-dimensional representations of stochastic events $\Omega \ni \omega \mapsto \mathbf{y}_\ell \in V_\ell$ to generate the input data for the PDE systems; (out of simplicity, we take V_ℓ for all functions with a discrete representation, but remark that different spaces can be chosen as well) and (iii) finding the root in (2.6) using an iterative stochastic approximation indexed by $k \in \mathbb{N}_0$. The iteration scheme of a standard SGD method is given with step sizes $t_k > 0$ by

$$\mathbf{z}_\ell^{(k+1)} \leftarrow \pi_Z\big(\mathbf{z}_\ell^{(k)} - t_k(\lambda \mathbf{z}_\ell^{(k)} - \mathbf{q}_\ell^{(k)})\big), \quad \text{where} \quad \pi_Z(\mathbf{z}_\ell) = \operatorname*{arg\,min}_{\mathbf{w}_\ell \in Z_\ell} \|\mathbf{z}_\ell - \mathbf{w}_\ell\|_W \tag{2.8}$$

ensures that the control $\mathbf{z}_\ell^{(k+1)}$ remains in the admissible discretized space $Z_\ell \subset V_\ell$. This approach, using $\mathbf{g}_\ell^{(k)} = \lambda \mathbf{z}_\ell^{(k)} - \mathbf{q}_\ell^{(k)}$ as stochastic gradient, is motivated by [1], with the methodology further developed in [3], where the step size t_k is assumed to satisfy

$$t_k > 0, \quad \sum_{k=1}^{\infty} t_k = \infty, \quad \sum_{k=1}^{\infty} t_k^2 < \infty, \tag{2.9}$$

to ensure convergence (cf. [28, 31] for further reading on step size control). This is combined with an adaptive mesh refinement and a step size decay to control the sampling error and the bias introduced by the FE method. Here, we approach this by introducing a MLMC estimator for the gradients, controlling both errors while drastically speeding up the computations and reducing the variance to make the choice of the step sizes less critical.

To outline all necessary assumptions for this approach, we consider an increasing sequence of sub-σ-algebras $\{\mathcal{F}_k\}_{k \in \mathbb{N}_0}$ of $\mathcal{F}$ (a Filtration), such that $\mathbf{z}_\ell^{(k)}$ and $\mathbf{z}^{(k)}$ are $\mathcal{F}_k$-measurable. Since all quantities updated in the optimization depend upon the previous steps, we introduce with the conditional expectation $\mathbb{E}_k[\,\cdot\,] := \mathbb{E}[\,\cdot\,|\mathcal{F}_k]$ (cf. [37, 38]) the norm $\|\mathbf{v}^{(k)}\|_{\mathrm{L}_k^2(\Omega,W)}^2 := \mathbb{E}_k\big[\|\mathbf{v}^{(k)}\|_W^2\big]$ and the space

$$\mathrm{L}_k^2(\Omega, W) := \Big\{ \mathbf{v}^{(k)} \in \mathrm{L}^2(\Omega, W) :\ \|\mathbf{v}^{(k)}\|_{\mathrm{L}_k^2(\Omega,W)}^2 < \infty \Big\}.$$

We remark that the computed control in (2.8) is a discretized random-field $\mathbf{z}_\ell^{(k)} \in \mathrm{L}_k^2(\Omega, Z_\ell)$ unlike the true solution $\mathbf{z}^* \in Z$ to Problem 2.1, which is deterministic.

As gradient estimator, the MLMC method gives its computational advantage through the variance reduction of the state and adjoint level differences

$$\mathbf{v}_\ell^{(k)} \coloneqq \mathbf{u}_\ell^{(k)} - \mathrm{P}_{\ell-1}^\ell \mathbf{u}_{\ell-1}^{(k)},\ \mathbf{v}_0^{(k)} \coloneqq \mathbf{u}_0^{(k)} \text{ and } \mathbf{p}_\ell^{(k)} \coloneqq \mathbf{q}_\ell^{(k)} - \mathrm{P}_{\ell-1}^\ell \mathbf{q}_{\ell-1}^{(k)},\ \mathbf{p}_0^{(k)} \coloneqq \mathbf{q}_0^{(k)}, \tag{2.10}$$

where we made use of the linear isometric projection operator $\mathrm{P}_{\ell-1}^\ell \colon V_{\ell-1} \to V_\ell$ as in [14]. The assumptions required for the MLMC estimation are summarized below.

Assumption 2.3. *The FE approximations of (2.2) and (2.7) with mesh diameter $h_\ell = h_0\, 2^{-\ell}$ satisfy in every optimization step k*

$$\big\|\mathbb{E}_k[\mathbf{u}_\ell^{(k)} - \mathbf{u}^{(k)}]\big\|_W \leq u_k h_\ell^{\alpha_\mathbf{u}} \tag{2.11a}$$

$$\big\|\mathbf{v}_\ell^{(k)} - \mathbb{E}_k[\mathbf{v}_\ell^{(k)}]\big\|_{\mathrm{L}_k^2(\Omega,W)}^2 \leq v_k h_\ell^{\beta_\mathbf{v}} \tag{2.11b}$$

$$\big\|\mathbb{E}_k\big[\mathbf{q}_\ell^{(k)} - \mathbf{q}^{(k)}\big]\big\|_W \leq q_k h_\ell^{\alpha_\mathbf{q}} \tag{2.12a}$$

$$\big\|\mathbf{p}_\ell^{(k)} - \mathbb{E}_k[\mathbf{p}_\ell^{(k)}]\big\|_{\mathrm{L}_k^2(\Omega,W)}^2 \leq p_k h_\ell^{\beta_\mathbf{p}} \tag{2.12b}$$

with exponents $\alpha_\mathbf{u}, \alpha_\mathbf{q}, \beta_\mathbf{v}, \beta_\mathbf{p} > 0$ and $u_k, q_k, v_k, p_k > 0$ being independent of the discretization level ℓ. The state $\mathbf{u}^{(k)}$ and the adjoint $\mathbf{q}^{(k)}$ represent the true solutions to the continuous problems (2.2) and (2.7) within the optimization algorithm. We further assume that the computation of the state-adjoint pair $(\mathbf{v}_\ell^{(k)}, \mathbf{p}_\ell^{(k)})$ is bounded with $\gamma_{\mathrm{CT}}, \gamma_{\mathrm{Mem}} > 0$ and $c_k, m_k > 0$ in its computing-time and memory footprint

$$\mathrm{C}^{\mathrm{CT}}\big((\mathbf{v}_\ell^{(k)}, \mathbf{p}_\ell^{(k)})\big) \leq c_k h_\ell^{-\gamma_{\mathrm{CT}}}. \tag{2.13}$$

$$\mathrm{C}^{\mathrm{Mem}}\big((\mathbf{v}_\ell^{(k)}, \mathbf{p}_\ell^{(k)})\big) \leq m_k h_\ell^{-\gamma_{\mathrm{Mem}}} \tag{2.14}$$

Lastly, with $c_\mathcal{G}, z_k > 0$ and $\alpha_\mathbf{z} > 0$ being also independent of ℓ, we suppose

$$\big\|\mathbf{z}_\ell^{(k)} - \mathbf{z}^{(k)}\big\|_{\mathrm{L}_k^2(\Omega,W)}^2 \leq c_\mathcal{G} \mathbb{E}_k\big[\big\|\mathbf{q}_\ell^{(k)} - \mathbf{q}^{(k)}\big\|_W^2\big] \leq z_k h_\ell^{2\alpha_\mathbf{z}}. \tag{2.15}$$

Remark 2.4.

(1) *Assumption 2.3 is based on [14] and adapted to fit to the MLSGD method by extending it to every optimization step.*

(2) *The constants $u_k, q_k, v_k, p_k, c_k, m_k$ and z_k possess k-dependence due to changing right-hand-sides of (2.2) and (2.7) as the optimization runs (cf. arguments of [39, 40, 41] for an elliptic PDE).*

(3) *The exponents $\alpha_\mathbf{u}, \alpha_\mathbf{q}, \beta_\mathbf{v}, \beta_\mathbf{p}, \gamma_{\mathrm{CT}}, \gamma_{\mathrm{Mem}}$ and $\alpha_\mathbf{z}$ are assumed to be independent of k as the regularity of the PDE solutions is not expected to change during the optimization.*

(4) *In the setting of this paper we denote $\alpha \coloneqq \alpha_\mathbf{q}, \beta \coloneqq \beta_\mathbf{p}$ and $\gamma \in \{\gamma_{\mathrm{CT}}, \gamma_{\mathrm{Mem}}\}$.*

(5) *Though assumptions (2.11a) and (2.11b) are not used in theory, we present numerical estimates on them in Section 6.2*

(6) *The cost bounds (2.13) and (2.14), while in [14] imposed for full field MLMC estimation, also might find justification in machine learning through a formulation with respect to the problem size. Commonly, computing time budgets and memory constraints are among the limiting factors in the training of neural networks.*

(7) *The inequality on the control error (2.15) is used in Lemma 5.1 to avoid specific assumptions on the operators $\mathcal{G}[\omega]$ and $\mathcal{G}^*[\omega]$ as well as the applied discretization schemes. This is motivated by inserting (2.2) and (2.7) into the left-hand side of (2.15) and thereof, $c_\mathcal{G}$ encodes information about the operators, whilst assumed of being independent of the optimization step k.*

(8) *By the work in [7, 42, 13, 14] the asymptotic behaviour of all inequalities in Assumption 2.3 can be estimated during runtime of the algorithm, enabling adaptivity and verification of the assumptions as done in Section 6.2.*

2.3 Example PDE

As an example of the system $\mathcal{G}[\omega]$, we consider the elliptic diffusion equation on $W = \mathrm{L}^2(\mathcal{D})$ and $V = \mathrm{H}^1_0(\mathcal{D})$ with log-normal coefficients and homogeneous Dirichlet boundary conditions

$$\left\{ \begin{array}{rcll} -\operatorname{div}\big(\exp(\mathbf{y}(\omega,\mathbf{x}))\nabla \mathbf{u}(\omega,\mathbf{x})\big) & = & \mathbf{z}(\mathbf{x}) & \text{on } \Omega\times\mathcal{D} \\ \mathbf{u}(\omega,\mathbf{x}) & = & 0 & \text{on } \Omega\times\partial\mathcal{D} \end{array} \right. \tag{2.16}$$

The adjoint operator $\mathcal{G}^*[\omega]$ of this system is the same as $\mathcal{G}[\omega]$ due to the problem's symmetry. This problem will serve as an example for the numerical experiments in this paper (cf. [3, 21] for more analysis); however, we emphasize that the methodology is not limited to this specific PDE as we do not rely on any properties other than already stated in the Sections 2.1 and 2.2. The majority of the used functionality within the presented algorithms has already been used for hyperbolic PDE systems [34, 13, 14]. Further analysis on other PDEs is also given, for example, in [5, 11, 16].

For our numerical experiments, we consider the unit square $\mathcal{D} = (0,1)^2$ as domain, impose the target state $\mathbf{d}(\mathbf{x}) = \sin(2\pi x_1)\sin(2\pi x_2)$, consider the cost factor $\lambda = 10^{-8}$ and set the admissible bounds to $\mathbf{z}_{\text{ad}}^{\text{low}} \equiv -1000, \mathbf{z}_{\text{ad}}^{\text{up}} \equiv 1000$. This leads to π_Z having no impact on the presented experiments, but can be adapted to the setting of [3]. To define the log-normal diffusion coefficient in (2.16), we consider a Gaussian random field (GRF), denoted $\mathbf{y}(\omega,\mathbf{x})$, with mean-zero and the Matérn covariance function

$$\operatorname{Cov}(\mathbf{x}_1,\mathbf{x}_2) = \frac{\sigma^2}{2^{\nu-1}\Gamma(\nu)}(\kappa\mathbf{r})^\nu \mathcal{K}_\nu(\kappa\mathbf{r}), \quad \mathbf{r} = \|\mathbf{x}_1-\mathbf{x}_2\|_2, \quad \kappa = \frac{\sqrt{2\nu}}{\lambda_\kappa},$$

where Γ is the gamma function and $\mathcal{K}_\nu$ the modified Bessel function of the second kind. Realizations of the GRFs are computed with SPDE sampling [43], particularly with the method introduced in [17]. Throughout the experiments, we set the parameters $\sigma^2 = 1.5$, $\nu = 1$ and $\lambda_\kappa = 0.1$. For two independent realizations $\mathbf{y}_\ell^{(m)}$ of the GRF with different mesh diameters, we refer to the four plots on the left of Figure 1. The plots on the right show the computed control $\mathbf{z}_\ell^{(k)}$ after $k = 10$ and $k = 100$ iterations of (3.1).

We solve all PDE systems (needed for the Dirichlet-Neumann averaging of [17] as well as the state and adjoint equations) with standard Lagrange linear FE, multigrid preconditioning and CG methods. We note that a large selection of other solvers and FE spaces, as explored in [12], are available and applicable in the used software [34] and within the proposed algorithms.

We remark that most numerical experiments are performed on a single node on the HoreKa supercomputer utilizing 64 CPUs. In Section 6.2, we present scaling experiments of moderate size up to sixteen nodes with a total of 1024 CPUs. Though the presented numerical results are for two-dimensional domains $\mathcal{D}$, the method is designed with three-dimensional domains in mind (cf. outlook of Figure 8). First numerical results for this were already achieved, however, omitted as this also requires an indepth discussion of the HPC techniques and challenges which is, such as the PDE system, not the main focus of this paper.

3 Batched Stochastic Gradient Descent

As a foundation and baseline for MLSGD, we first introduce the BSGD method. We begin with an algorithmic description in Section 3.1, followed by the first numerical insights and experiments in Section 3.2, and conclude with a discussion of its convergence rate and computational complexity in Section 3.3.

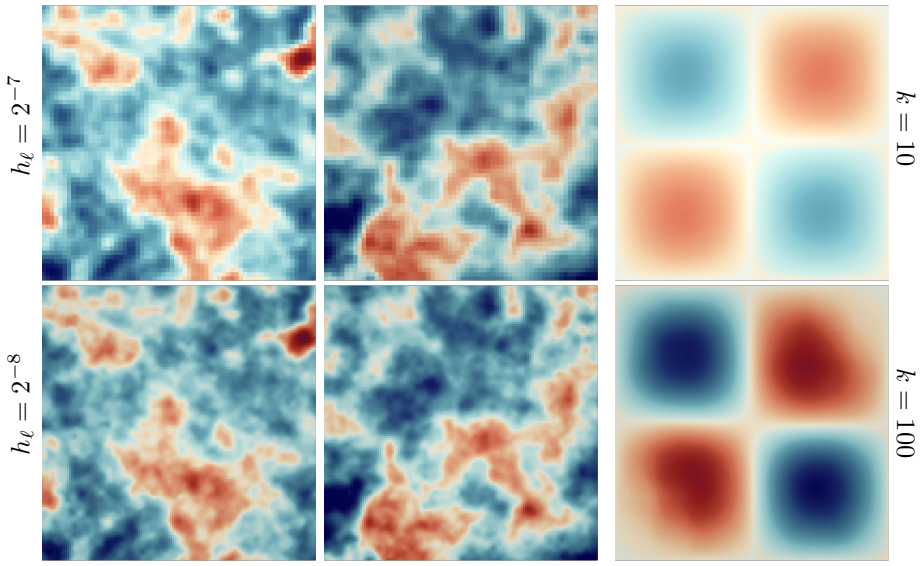


FIGURE 1. Left to right: Two GRF samples $\mathbf{y}_\ell^{(m)}$ on different mesh diameters; Computed control $\mathbf{z}_\ell^{(k)}$ after $k = 10$ and $k = 100$ iterations of Algorithm 1.

3.1 Algorithm

At its core, BSGD generates, such as (2.8), a minimizing sequence $\{\mathbf{z}_\ell^{(k)}\}_{k=0}^K \subset Z_\ell$ of approximations to find the optima of Problem 2.1. We refer to Algorithm 1 for a high-level functional pseudocode generating this sequence.

BSGD function. The algorithm starts in the `BSGD` function taking an initial guess $\mathbf{z}_\ell^{(0)} \in Z_\ell$, an appropriate step size rule $\{t_k\}_{k=0}^{K-1}$ (here directly given as a sequence of length K, which may be replaced by a function) and the batch size M as inputs. The sequence $\{\mathbf{z}_\ell^{(k)}\}_{k=0}^K \subset Z_\ell$ is generated by iteratively solving the root problem (2.6) with estimates to $\mathbb{E}_k[\mathbf{q}^{(k)}] \in \mathrm{L}_k^2(\Omega, V)$

$$\mathbf{z}_\ell^{(k+1)} \leftarrow \pi_Z\big(\mathbf{z}_\ell^{(k)} - t_k(\lambda \mathbf{z}_\ell^{(k)} - E_M^{\mathrm{MC}}[\mathbf{q}_\ell^{(k)}])\big) \text{ with } E_M^{\mathrm{MC}}[\mathbf{q}_\ell^{(k)}] \coloneqq \frac{1}{M}\sum_{m=1}^M \mathbf{q}_\ell^{(m,k)}. \tag{3.1}$$

Here, we represent the batch estimation as a MC method using M independent and identically distributed samples of the approximated adjoint solutions $\mathbf{q}_\ell^{(m,k)}$. These samples are generated within the `BatchEstimation` function called in each optimization step.

Batch estimation function. Every call to `BatchEstimation` takes the current control $\mathbf{z}_\ell^{(k)}$ and the batch size M as input. Then, M independent realizations of $\mathbf{y}_\ell^{(m)}$ are drawn and used as input, together with the fixed $\mathbf{z}_\ell^{(k)}$, to approximate the solution of the systems (2.2) and (2.7). This computation can be fully parallelized for all samples, provided sufficient memory and processing resources are available. The function returns the Monte Carlo estimate of the adjoint solution, $E_M^{\mathrm{MC}}[\mathbf{q}_\ell^{(k)}]$, to be used in (3.1), as well as an estimate of the objective (2.1)

$$J_M^{\mathrm{MC}}(\mathbf{z}_\ell^{(k)}) \coloneqq \frac{1}{M}\sum_{m=1}^M \tfrac{1}{2}\big\|\mathbf{u}_\ell^{(m,k)} - \mathbf{d}\big\|_W^2 + \tfrac{\lambda}{2}\big\|\mathbf{z}_\ell^{(k)}\big\|_W^2 \tag{3.2}$$

Algorithm 1 Batched Stochastic Gradient Descent (BSGD)

```
function BSGD(z_ℓ^(0), {t_k}_{k=0}^{K-1}, M):
    for k = 0, ..., K-1:
        E_M^MC[q_ℓ^(k)], J_M^MC(z_ℓ^(k)) ← BatchEstimation(z_ℓ^(k), M)
        z_ℓ^(k+1) ← π_Z(z_ℓ^(k) − t_k(λ z_ℓ^(k) − E_M^MC[q_ℓ^(k)]))
    return z_ℓ^(K)

function BatchEstimation(z_ℓ^(k), M):
    for m = 1, 2, ..., M:
        // Sampling method for ω ↦ y_ℓ^(m)
        y_ℓ^(m) ← [ ] { ⋮
        // Find state u_ℓ^(m,k) to control z_ℓ^(k) and realization y_ℓ^(m)
        u_ℓ^(m,k) ← [y_ℓ^(m), z_ℓ^(k)] { Find u_ℓ^(m,k) ∈ V_ℓ such that:
                                          G_ℓ[y_ℓ^(m)] u_ℓ^(m,k) = z_ℓ^(k)
        // Find adjoint q_ℓ^(m,k) to state u_ℓ^(m,k) and realization y_ℓ^(m)
        q_ℓ^(m,k) ← [y_ℓ^(m), d − u_ℓ^(m,k)] { Find q_ℓ^(m,k) ∈ V_ℓ such that:
                                                G_ℓ*[y_ℓ^(m)] q_ℓ^(m,k) = d − u_ℓ^(m,k)
    // Return result of estimators defined in (3.1) and (3.2)
    return E_M^MC[q_ℓ^(k)], J_M^MC(z_ℓ^(k))
```

using the state approximations $\mathbf{u}_\ell^{(m,k)}$. In machine learning, this is often referred to as empirical risk [28], which we use here to experimentally monitor convergence (see Subsections 3.2, 5.2, and 6.2). However, in the convergence analysis, we prefer using $J(\mathbf{z}_\ell^{(k)})$ to avoid introducing an additional estimation error. Note that we also use $J(\mathbf{z}^{(k)})$, for instance in Lemma 5.1, to express the objective (2.1) for the continuous control $\mathbf{z}^{(k)} \in \mathrm{L}^2_k(\Omega, W)$ of (2.2), which is not spatially discretized but still subject to stochastic approximation.

3.2 Experiments

To convey the functionality of the BSGD method and its implementation in M++ [34], we present the first numerical experiments to evaluate the method and to motivate potential improvements.

Investigation of the batch size. We examine the influence of a variable batch size M on a grid with mesh diameter $h_\ell = 2^{-7}$ combined with the constant step size $t_k \equiv 100$ and the initial control $\mathbf{z}^{(0)} \equiv 0$. Figure 2 illustrates two plots: the left one presents estimates of $J_M^{\mathrm{MC}}(\mathbf{z}_\ell^{(k)})$ according to (3.2) over the iteration k, while the right plot shows evaluations of the gradient used in (3.1) in the L^2-norm $\|E_M^{\mathrm{MC}}[\mathbf{g}_\ell^{(k)}]\|_W := \|\lambda \mathbf{z}_\ell^{(k)} - E_M^{\mathrm{MC}}[\mathbf{q}_\ell^{(k)}]\|_{\mathrm{L}^2(\mathcal{D})}$ plotted against the total computing time. A higher noise level in $J_M^{\mathrm{MC}}(\mathbf{z}_\ell^{(k)})$ is observed for smaller batch sizes M, yet all methods oscillate around the same value. The impact of the batch size on the gradient norm is even more pronounced: the optimality condition (2.6) is visibly better satisfied with larger batches, but this comes at the cost of an increased computational effort. Beyond this, a smoothed descent and arguably better convergence rates are observed for larger batches. Thus,

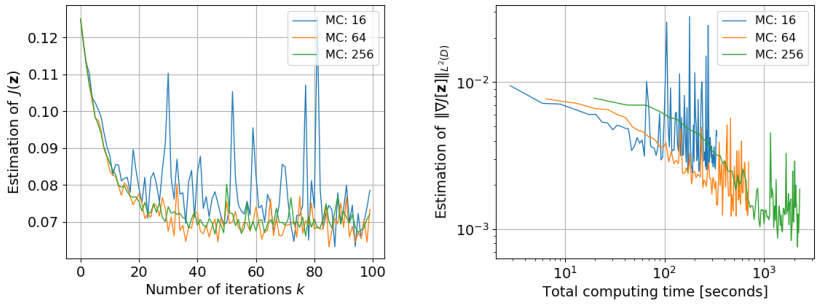


FIGURE 2. Comparison of different batch sizes M.

if high accuracy is desired, larger batches are more reliable in achieving the result. However, as soon as we impose a tight limit on the computational cost, the choice for the smaller batch might be preferable.

Further experiments. Running Algorithm 1 requires the choice of several hyperparameters and adaptions to the given computational environment. This includes besides the batch size M, the appropriate choice of the discretization level ℓ, the step size t_k, the number of optimization steps K, the parallelization strategy, and further configurations of the involved numerical solvers and discretization schemes. We omit a detailed presentation of our numerical experiments conducted on this, and refer to Section 6 where we develop an adaptive algorithm determining several of the hyperparameters automatically. Before we turn to this, we recall the convergence and complexity theory of BSGD methods.

3.3 Properties and Discussion

As just experimentally observed in Section 3.2, the batch size M is crucial, M being too small leads to a high variance in the gradient estimate, while M being too large increases the computational cost. This leads to a trade-off between the per-iteration cost and expected per-iteration improvements. To navigate this trade-off, we recall the convergence properties of SAA and SA methods, here distinguished through the batch size $M^{\mathrm{SAA}} \gg M^{\mathrm{SA}}$.

Convergence and complexity. Assuming strong μ-convexity (2.3) and an appropriate choice of step sizes t_k (e.g. satisfying (2.9)), an SGD method (including mini-batches of size M^{SA}) converges at a sub-linear rate, in particular (cf. [28, 1, 31, 4])

$$\mathbb{E}\big[J(\mathbf{z}_\ell^{(K)}) - J(\mathbf{z}^*)\big] \leq \mathcal{O}(K^{-1}) \quad \text{at cost} \quad \mathrm{C}_\epsilon^{\mathrm{SA}} \lesssim M^{\mathrm{SA}} h_\ell^{-\gamma} \epsilon^{-1}. \tag{3.3}$$

Here, h_ℓ is assumed to be sufficiently small, and $\mathrm{C}_\epsilon^{\mathrm{SA}}$ represents the cost required to reach an accuracy of at least $\epsilon > 0$, where $M^{\mathrm{SA}} h_\ell^{-\gamma}$ denotes the cost of one batch, and $h_\ell^{-\gamma}$ is proportional to the cost per sample, as assumed in (2.13) and (2.14).

On the other hand, if the batch size is large enough to fall within the SAA regime, classical optimization methods (e.g., conjugate gradient descent, BFGS, ...) achieve at least linear convergence rates. That is, for some $\rho \in (0,1)$, we have (cf. [28, 22])

$$\mathbb{E}\big[J(\mathbf{z}_\ell^{(K)}) - J(\mathbf{z}^*)\big] \leq \mathcal{O}(\rho^K) \quad \text{at cost} \quad \mathrm{C}_\epsilon^{\mathrm{SAA}} \lesssim -M^{\mathrm{SAA}} h_\ell^{-\gamma} \log(\epsilon) \tag{3.4}$$

to reach accuracy $0 < \epsilon < 1$. Thus, SAA methods offer superior convergence rates compared to SA methods. However, since $M^{\mathrm{SAA}} \gg M^{\mathrm{SA}}$, each optimization step in SAA is significantly more expensive. In cases where the input data exhibits high approximate similarity, SA methods are in practice more efficient [28]. But, once the optimization error falls within the same

order as the approximate similarity, SAA methods become preferable, as SA methods can no longer guarantee an improvement in expectation. We refer again to Figure 2 where aspects of this discussion can be observed, too.

Adaptivity and step sizes. To combat the increased variance of SA methods, but also to reduce the total computational cost of SAA methods, adaptive batch sizes (cf. [15, 16, 22] for optimal control), adaptive step sizes [31] and the ADAM optimizer [30] have been proposed, giving more robust paths of iterates $\mathbf{z}_\ell^{(k)}$, objectives $J_M^{\mathrm{MC}}(\mathbf{z}_\ell^{(k)})$ and gradient evaluations. This also enables a stopping criterion other than a maximum number of optimization steps K. We also note that convergence can still be shown for the convex case, by considering the averaging scheme of [32], for details see [3]. This also requires a decreasing or sufficiently small sequence of t_k, which also has to be chosen carefully to ensure convergence.

4 Multilevel Monte Carlo

As a remedy to the observed challenges in the previous section, we propose the use of an MLMC estimator to determine the expected discretized adjoint solution in each optimization step k. Expanding $\mathbb{E}_k[\mathbf{q}_L^{(k)}]$ in a telescoping sum over discretization levels $\ell = 0, \dots, L$ of the FE mesh, inserting equation (2.10) and using the isometric transfer operators $\mathrm{P}_\ell^L \colon V_\ell \to V_L$ gives

$$\mathbb{E}_k[\mathbf{q}_L^{(k)}] = \mathrm{P}_0^L \mathbb{E}_k[\mathbf{q}_0^{(k)}] + \sum_{\ell=1}^{L} \mathrm{P}_\ell^L \mathbb{E}_k[\mathbf{q}_\ell^{(k)} - \mathrm{P}_{\ell-1}^\ell \mathbf{q}_{\ell-1}^{(k)}] = \sum_{\ell=0}^{L} \mathbb{E}_k[\mathbf{p}_\ell^{(k)}] \approx \mathbb{E}_k[\mathbf{q}^{(k)}].$$

This motivates the MLMC estimator (cf. [14] for details in similar notation)

$$E^{\mathrm{ML}}[\mathbf{q}_L^{(k)}] \coloneqq \sum_{\ell=0}^{L} \mathrm{P}_\ell^L E_{M_\ell}^{\mathrm{MC}}[\mathbf{p}_\ell^{(k)}] \quad \text{with} \quad E_{M_\ell}^{\mathrm{MC}}[\mathbf{p}_\ell^{(k)}] \coloneqq \frac{1}{M_\ell} \sum_{m=1}^{M_\ell} \mathbf{p}_\ell^{(m,k)}, \tag{4.1}$$

where $\{M_\ell\}_{\ell=0}^L$ denotes the number of samples on each level. Note that for $\ell \neq 0$ the difference in (2.10) is the same sample computed on two different levels, i.e., $\mathbf{p}_\ell^{(m,k)} = \mathbf{q}_\ell^{(m,k)} - \mathrm{P}_{\ell-1}^\ell \mathbf{q}_{\ell-1}^{(m,k)}$ (cf. Figure 1 for an illustration of the same input data on two different levels). The mean squared error in the above approximation is expressed by (cf. e.g. [44, Theorem 3.1] for a similar setting)

$$\mathrm{err}_k^{\mathrm{MSE}}\big(E^{\mathrm{ML}}[\mathbf{q}_L^{(k)}]\big) = \underbrace{\sum_{\ell=0}^{L} M_\ell^{-1} \big\|\mathbf{p}_\ell^{(k)} - \mathbb{E}_k[\mathbf{p}_\ell^{(k)}]\big\|_{\mathrm{L}_k^2(\Omega,W)}^2}_{\coloneqq \mathrm{err}_k^{\mathrm{sam}}} + \underbrace{\big\|\mathbb{E}_k[\mathbf{q}_L^{(k)} - \mathbf{q}^{(k)}]\big\|_W^2}_{\coloneqq \mathrm{err}_k^{\mathrm{num}}}. \tag{4.2}$$

If (2.12a) and (2.12b) are satisfied, we can control both; the sampling error $\mathrm{err}_k^{\mathrm{sam}}$ and the numerical error $\mathrm{err}_k^{\mathrm{num}}$, either by increasing the level-dependent number of samples M_ℓ or by refining the mesh. Naturally, both measures come at an increased computational cost captured by (2.13) and (2.14). However, the multilevel structure allows for the construction of a sample sequence $\{M_\ell\}_{\ell=0}^L$, such that the computational cost C_ϵ (measured in memory or computing-time) to reach a desired accuracy of $0 < \epsilon < \mathrm{e}^{-1}$ is bounded by (cf. [6, 25])

$$\sqrt{\mathrm{err}^{\mathrm{MSE}}\big(E^{\mathrm{ML}}[\mathbf{q}_L^{(k)}]\big)} \overset{!}{<} \epsilon \quad \Rightarrow \quad \mathrm{C}_\epsilon\big(E^{\mathrm{ML}}[\mathbf{q}_L^{(k)}]\big) \lesssim \begin{cases} \epsilon^{-2} & \beta > \gamma, \\ \epsilon^{-2}(\log(\epsilon))^2 & \beta = \gamma, \\ \epsilon^{-2-(\gamma-\beta)/\alpha} & \beta < \gamma. \end{cases} \tag{4.3}$$

This stands in contrast to the combination of FE with standard MC methods which has the cost of $\mathrm{C}_\epsilon\big(E_M^{\mathrm{MC}}[\mathbf{q}_L^{(k)}]\big) \lesssim \epsilon^{-2-\gamma/\alpha}$, giving at least an improvement of β/α in the rate for the same accuracy [6]. The central idea of this paper is to leverage this improvement for the

batch estimation within an SGD method. Before we outline this approach in further detail in Section 5, we briefly state how to estimate $\mathrm{err}^{\mathrm{sam}}$ and $\mathrm{err}^{\mathrm{num}}$. This is done by (cf. [42, 7, 14])

$$\widehat{\mathrm{err}}_k^{\mathrm{num}} := \max_{\ell=1,\dots,L} \left(\frac{\|E_{M_\ell}^{\mathrm{MC}}[\mathbf{p}_\ell^{(k)}]\|_W}{(2^{\widehat{\alpha}}-1)2^{\widehat{\alpha}(L-\ell)}} \right)^2 \text{ with } \min_{(\widehat{\alpha},\widehat{c})} \sum_{\ell=1}^{L} \left(\log_2 \|E_{M_\ell}^{\mathrm{MC}}[\mathbf{p}_\ell^{(k)}]\|_W + \widehat{\alpha}\ell - \widehat{c} \right)^2 \tag{4.4}$$

thus $\widehat{\alpha}$ is determined by regression and $\widehat{\mathrm{err}}_k^{\mathrm{num}}$ follows from extrapolation arguments. The estimation of the sampling error is similar to computing a sample variance, i.e.,

$$\widehat{\mathrm{err}}_k^{\mathrm{sam}} := \sum_{\ell=0}^{L} \frac{s_\ell^2[\mathbf{p}_\ell^{(k)}]}{M_\ell(M_\ell-1)} \quad \text{with} \quad s_\ell^2[\mathbf{p}_\ell^{(k)}] := \sum_{m=1}^{M_\ell} \left\| \mathbf{p}_\ell^{(m,k)} - E_{M_\ell}^{\mathrm{MC}}[\mathbf{p}_\ell^{(k)}] \right\|_W^2. \tag{4.5}$$

5 Multilevel Stochastic Gradient Descent

With the previous recall of MLMC and BSGD, we now introduce one main contribution of this work, the Multilevel Stochastic Gradient Descent (MLSGD) method. We outline the algorithm in Section 5.1, present the first experimental results in Section 5.2 and analyze its convergence and complexity in Section 5.3.

5.1 Algorithm

To explain Algorithm 2, we first note its similarity to the previously discussed BSGD method outlined in Algorithm 1 and draw on knowledge from Section 4 to explain the differences and advantages. Again, we follow a function-wise explanation of the algorithm.

The MLSGD function. The `MLSGD` function serves as the entry point to Algorithm 2 and accepts an initial guess for $\mathbf{z}_L^{(0)}$, an appropriate sequence of step sizes $\{t_k\}_{k=0}^{K-1}$, and a multilevel batch $\{M_\ell\}_{\ell=0}^{L}$ as input arguments. Similar to Algorithm 1, the MLSGD method iteratively optimizes the control $\mathbf{z}_L^{(k)}$. However, instead of using (3.1), we employ the iteration scheme

$$\mathbf{z}_L^{(k+1)} \leftarrow \pi_Z\left(\mathbf{z}_L^{(k)} - t_k E^{\mathrm{ML}}[\mathbf{g}_L^{(k)}]\right) \quad \text{with} \quad E^{\mathrm{ML}}[\mathbf{g}_L^{(k)}] := \lambda \mathbf{z}_L^{(k)} - E^{\mathrm{ML}}[\mathbf{q}_L^{(k)}] \tag{5.1}$$

to approximate the solution to Problem 2.1. We note that the gradient estimation $E^{\mathrm{ML}}[\mathbf{g}_L^{(k)}]$ is now computed using $E^{\mathrm{ML}}[\mathbf{q}_L^{(k)}]$, as given through (4.1).

Multilevel batch estimation. To compute the adjoint $E^{\mathrm{ML}}[\mathbf{q}_L^{(k)}]$, the function `MultiLevelEstimation` is called in each optimization step, taking the current control $\mathbf{z}_L^{(k)}$ and a multilevel batch $\{M_\ell\}_{\ell=0}^{L}$ as input arguments. By (2.10), the adjoint and the state are computed with the `BatchEstimation` function, as in Algorithm 1, on the lowest level $\ell = 0$. For the higher levels $\ell = 1, \dots, L$, the estimation is performed for the level pair $(\ell, \ell-1)$ using the function `LevelPairEstimation`. This also motivates the definitions of $\mathbf{y}_{\ell,\ell-1}^{(m,k)}$, $\mathbf{u}_{\ell,\ell-1}^{(m,k)}$ and $\mathbf{q}_{\ell,\ell-1}^{(m,k)}$ in the top row of Algorithm 2.

Level pair batch estimation. Similar to the `BatchEstimation` function in Algorithm 1, the `LevelPairEstimation` function approximates the state (2.2) and adjoint (2.7) systems, taking the current control $\mathbf{z}_L^{(k)}$, the level-specific batch size M_ℓ, and the level ℓ as input arguments. Since the control $\mathbf{z}_L^{(k)}$ is always stored at the highest level L, the first step is to restrict it to levels ℓ and $\ell-1$ with the restriction operators $\mathrm{R}_L^\ell \colon V_L \to V_\ell$ and $\mathrm{R}_L^{\ell-1} \colon V_L \to V_{\ell-1}$, respectively. Additionally, the control must be distributed across the multiple processes executing the batch loop in parallel. We omit the details on the parallelization and refer to [13, 14] as well as to a short discussion in Section 5.2. Having the distributed and restricted controls $\mathbf{z}_{\ell,\ell-1}^{(k)}$, we can approximate the state and adjoint on both levels ℓ and $\ell-1$, taking the realizations $\mathbf{y}_{\ell,\ell-1}^{(m,k)}$ as input. Finally, the function returns with $\mathbf{p}_\ell^{(m,k)} = \mathbf{q}_\ell^{(m,k)} - \mathrm{P}_{\ell-1}^{\ell} \mathbf{q}_{\ell-1}^{(m,k)}$ the MC estimate

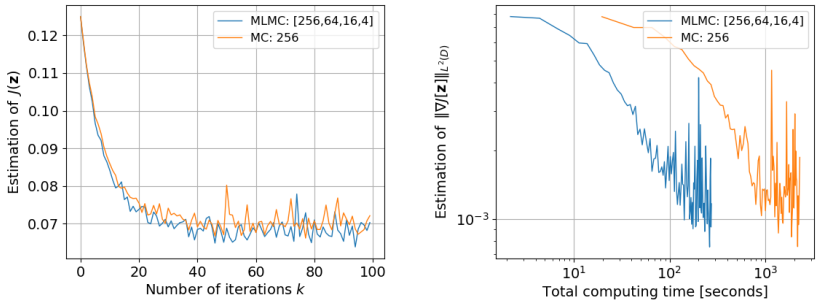


FIGURE 3. Comparison of MC and MLMC gradient estimation.

of the adjoint level difference as in (4.1), and with $\mathrm{Q}_\ell^{(m,k)} := \frac{1}{2}\|\mathbf{u}_\ell^{(m,k)} - \mathbf{d}\|_W^2$, $\mathrm{Y}_\ell^{(m,k)} := \mathrm{Q}_\ell^{(m,k)} - \mathrm{Q}_{\ell-1}^{(m,k)}$ for $\ell \geq 1$ and $\mathrm{Y}_0^{(m,k)} := \mathrm{Q}_0^{(m,k)}$ for $\ell = 0$ the estimate

$$J_{M_\ell}^{\mathrm{MC}}(\mathbf{z}_{\ell,\ell-1}^{(k)}) := \frac{1}{M_\ell}\sum_{m=1}^{M_\ell} \mathrm{Y}_\ell^{(m,k)}. \tag{5.2}$$

This enables the estimation for (2.1), then returned by the `MultiLevelEstimation` function

$$J^{\mathrm{ML}}(\mathbf{z}_L^{(k)}) := \sum_{\ell=0}^{L} J_{M_\ell}^{\mathrm{MC}}(\mathbf{z}_{\ell,\ell-1}^{(k)}) + \tfrac{\lambda}{2}\|\mathbf{z}_L^{(k)}\|_W^2. \tag{5.3}$$

In conclusion, Algorithm 2 utilizes a MLMC method as replacement for the batch estimator. Care has to be taken to ensure that the control is distributed and restricted to the correct data structures; and that the level pairs are solved with the same input realizations. The overall algorithm, however, is still strongly related to the BSGD method.

5.2 Experiments

Having introduced Algorithm 2, we present first experiments to evaluate the method. Particularly, we compare the MLSGD with the BSGD method and further investigate the influence of the multilevel batch $\{M_\ell\}_{\ell=0}^L$ at hand of the PDE example introduced in Section 2.3.

Comparing multilevel estimation and batch estimation. The layout of Figure 3 follows the figures in Section 3.2, illustrating a comparison between MLMC and MC gradient estimation. Both methods use the constant step size $t_k \equiv 100$, the same initial control $\mathbf{z}^{(0)} \equiv 0$, the same number of iterations $K = 100$, and the same number of CPUs $|\mathcal{P}| = 64$. The mesh resolution starts with $h_0 = 2^{-4}$ and ends with $h_L = 2^{-7}$ for the MLMC case, where as MC estimation operates as in Section 3.2 on $h_\ell = 2^{-7}$. Neither of the estimated objectives, $J^{\mathrm{ML}}(\mathbf{z}_L^{(k)})$ and $J_M^{\mathrm{MC}}(\mathbf{z}_L^{(k)})$ in Figure 3, shows a clear advantage over the iterations k, as both oscillate around the same value with a similar noise level. However, the plot of the gradient norms reveals a significant speedup—greater than factor 10 in this example—for the MLMC gradient estimation. The MLSGD method achieves a similar convergence rate and gradient quality by offloading variance reduction through batching on the lower discretization levels at significantly reduced computational cost. This serves as an initial indication that MLMC is a promising alternative to batch estimation.

Investigation of the multilevel batch size. Next, we examine the influence of the multilevel batch size $\{M_\ell\}_{\ell=0}^L$ illustrated in Figure 4. The blue lines in Figure 3 and Figure 4 correspond to the same batch with $h_0 = 2^{-4}$ and $h_L = 2^{-7}$, where as the orange and green lines in Figure 4 extend on $h_L = 2^{-8}$ and $h_L = 2^{-9}$, respectively. We note that larger multilevel batches yield

Algorithm 2 Multilevel Stochastic Gradient Descent (MLSGD)

`def` $\mathbf{y}_{\ell,\ell-1}^{(m)} \coloneqq (\mathbf{y}_{\ell}^{(m)}, \mathbf{y}_{\ell-1}^{(m)}),\ \mathbf{u}_{\ell,\ell-1}^{(m,k)} \coloneqq (\mathbf{u}_{\ell}^{(m,k)}, \mathbf{u}_{\ell-1}^{(m,k)}),\ \mathbf{q}_{\ell,\ell-1}^{(m,k)} \coloneqq (\mathbf{q}_{\ell}^{(m,k)}, \mathbf{q}_{\ell-1}^{(m,k)})$

`function MLSGD`$(\mathbf{z}_L^{(0)}, \{M_\ell\}_{\ell=0}^{L}, \{t_k\}_{k=0}^{K-1})$:

- `for` $k = 0, \dots, K-1$:
  - $E^{\mathrm{ML}}[\mathbf{q}_L^{(k)}],\ J^{\mathrm{ML}}(\mathbf{z}_L^{(k)}) \ \leftarrow$ `MultiLevelEstimation`$(\mathbf{z}_L^{(k)}, \{M_\ell\}_{\ell=0}^{L})$
  - $\mathbf{z}_L^{(k+1)},\ E^{\mathrm{ML}}[\mathbf{g}_L^{(k)}] \ \leftarrow$ `GradientDescent`$(\mathbf{z}_L^{(k)}, E^{\mathrm{ML}}[\mathbf{q}_L^{(k)}], t_k)$
- `return` $\mathbf{z}_L^{(K)}$

`function GradientDescent`$(\mathbf{z}_L^{(k)}, E^{\mathrm{ML}}[\mathbf{q}_L^{(k)}], t_k)$:

- $E^{\mathrm{ML}}[\mathbf{g}_L^{(k)}] \ \leftarrow \lambda \mathbf{z}_L^{(k)} - E^{\mathrm{ML}}[\mathbf{q}_L^{(k)}]$
- $\mathbf{z}_L^{(k+1)} \ \leftarrow \pi_Z\big(\mathbf{z}_L^{(k)} - t_k E^{\mathrm{ML}}[\mathbf{g}_L^{(k)}]\big)$
- `return` $\mathbf{z}^{(k+1)},\ E^{\mathrm{ML}}[\mathbf{g}_L^{(k)}]$

`function MultiLevelEstimation`$(\mathbf{z}_L^{(k)}, \{M_\ell\}_{\ell=0}^{L})$:

- // Solve state and adjont system on lowest level $\ell = 0$ (cf. Algorithm 1)
- $E_{M_0}^{\mathrm{MC}}[\mathbf{p}_0^{(k)}],\ J_{M_0}^{\mathrm{MC}}(\mathbf{z}_0^{(k)}) \leftarrow$ `BatchEstimation`$(\mathbf{z}_L^{(k)}, M_\ell)$
- `for` $\ell = 1, \dots, L$:
  - // Solve state and adjont system for level pair $(\ell, \ell-1)$
  - $E_{M_\ell}^{\mathrm{MC}}[\mathbf{p}_\ell^{(k)}],\ J_{M_\ell}^{\mathrm{MC}}(\mathbf{z}_{\ell,\ell-1}^{(k)}) \leftarrow$ `LevelPairEstimation`$(\mathbf{z}_L^{(k)}, M_\ell, \ell)$
- // Return result of multilevel sums as in (4.1) and (5.3)
- `return` $\sum_{\ell=0}^{L} \mathrm{P}_\ell^L E_{M_\ell}^{\mathrm{MC}}[\mathbf{p}_\ell^{(k)}],\ \ \sum_{\ell=0}^{L} J_{M_\ell}^{\mathrm{MC}}(\mathbf{z}_{\ell,\ell-1}^{(k)}) + \frac{\lambda}{2}\|\mathbf{z}_L^{(k)}\|_W^2$

`function LevelPairEstimation`$(\mathbf{z}_L^{(k)}, M_\ell, \ell)$:

- $\mathbf{z}_{\ell,\ell-1}^{(k)} \leftarrow (\mathrm{R}_L^{\ell}\mathbf{z}_L^{(k)},\ \mathrm{R}_L^{\ell-1}\mathbf{z}_L^{(k)})$ // Restrict and distribute control
- `for` $m = 1, 2, \dots, M_\ell$: // Run in parallel with optimal distr.
  - // Sampling method for $\omega \mapsto \mathbf{y}_{\ell,\ell-1}^{(m)}$
  - $\mathbf{y}_{\ell,\ell-1}^{(m)} \leftarrow [\qquad\qquad] \left\{ \quad \vdots \right.$
  - // Find states $\mathbf{u}_{\ell,\ell-1}^{(m,k)}$ to controls $\mathbf{z}_{\ell,\ell-1}^{(k)}$ and realizations $\mathbf{y}_{\ell,\ell-1}^{(m)}$
  - $\mathbf{u}_{\ell,\ell-1}^{(m,k)} \leftarrow [\mathbf{y}_{\ell,\ell-1}^{(m)},\ \ \mathbf{z}_{\ell,\ell-1}^{(k)}] \left\{ \begin{array}{l} \texttt{Find } \mathbf{u}_{\ell,\ell-1}^{(m,k)} \in V_{\ell,\ell-1} \texttt{ such that:} \\ \quad \mathcal{G}_{\ell,\ell-1}[\mathbf{y}_{\ell,\ell-1}^{(m)}]\ \mathbf{u}_{\ell,\ell-1}^{(m,k)} = \mathbf{z}_{\ell,\ell-1}^{(k)} \end{array} \right.$
  - // Find adjoints $\mathbf{q}_{\ell,\ell-1}^{(m,k)}$ to states $\mathbf{u}_{\ell,\ell-1}^{(m,k)}$ and realizations $\mathbf{y}_{\ell,\ell-1}^{(m)}$
  - $\mathbf{q}_{\ell,\ell-1}^{(m,k)} \leftarrow [\mathbf{y}_{\ell,\ell-1}^{(m)}, \mathbf{d} - \mathbf{u}_{\ell,\ell-1}^{(m,k)}] \left\{ \begin{array}{l} \texttt{Find } \mathbf{q}_{\ell,\ell-1}^{(m,k)} \in V_{\ell,\ell-1} \texttt{ such that:} \\ \quad \mathcal{G}_{\ell,\ell-1}^{*}[\mathbf{y}_{\ell,\ell-1}^{(m)}]\ \mathbf{q}_{\ell,\ell-1}^{(m,k)} = \mathbf{d} - \mathbf{u}_{\ell,\ell-1}^{(m,k)} \end{array} \right.$
- // Return result of estimators defined in (4.1) and (5.2)
- `return` $E_{M_\ell}^{\mathrm{MC}}[\mathbf{p}_\ell^{(k)}],\ \ J_{M_\ell}^{\mathrm{MC}}(\mathbf{z}_{\ell,\ell-1}^{(k)})$

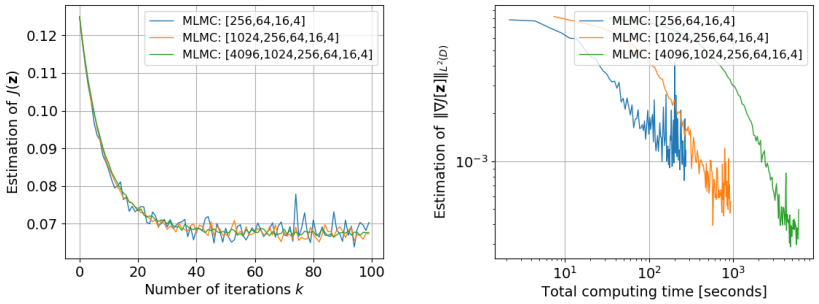


FIGURE 4. Comparison of different multilevel batch sizes $\{M_\ell\}_{\ell=0}^L$.

smoother and more robust estimates of the objective $J^{\mathrm{ML}}(\mathbf{z}_L^{(k)})$. The gradient norm plotted over computing time motivates a similar discussion as in Section 3.2, i.e., the larger the multilevel batch, the better is (2.6) satisfied due to the reduced variance, but this time also due to the reduced bias. Yet, the higher quality of the computational results naturally comes at increased cost.

Discussion and remaining challenges. Though the MLSGD method shows promising results in the numerical experiments, several questions remain as well as new challenges arise: (i) We note that picking the optimal multilevel batch size still depends upon the available computational resources. Smaller batches are still to be favored for small compute time budgets. In Section 6, we develop a hardware-aware method that automatically finds the optimal batch size for a given computational budget in CPU-time and memory. (ii) To incorporate the computational budgets, an adaptive strategy for the step size t_k, the multilevel batch $\{M_\ell\}_{\ell=0}^L$, the total number of iterations K and the largest level L has to be developed. Before we propose ways to approach (i) and (ii) algorithmically in Section 6, we examine the method's convergence and complexity behavior analytically in Section 5.3. (iii) Lastly, leveraging parallel computing resources within a multilevel setting is a non-trivial task. The reason is that the MC estimates on the lower levels are much more efficient with a sample parallelization, whereas the MC estimates on the higher levels often require a parallelization over the spatial domain $\mathcal{D}$. Yet, all estimates must be synchronized to compute the adjoint estimate in (4.1). To leverage the full potential of MLMC estimation without sacrificing parallel efficiency, careful algorithmic design is required which is solved with a multiindex approach for full field estimates as introduced in [14].

5.3 Analysis

Finally, within this section, we present comprehensive convergence and complexity analysis of the MLSGD method. The main result is given in Theorem 5.2, leveraging Lemma 5.1 as a key idea. Corollary 5.4 generalizes the convergence result on the objective and on the gradient.

Lemma 5.1. *There exists a multilevel batch $\{M_\ell\}_{\ell=0}^L$, such that the error $\mathbf{r}_L^{(k)}$ between the gradient $\nabla J(\mathbf{z}^{(k)})$ in step k and its estimation $E^{ML}[\mathbf{g}_L^{(k)}]$ in (5.1)*

$$\mathbf{r}_L^{(k)} := E^{ML}[\mathbf{g}_L^{(k)}] - \nabla J(\mathbf{z}^{(k)}) \tag{5.4}$$

as well as $\left\|\mathbf{z}_L^{(k)} - \mathbf{z}^{(k)}\right\|_{\mathrm{L}_k^2(\Omega,W)}$ can be bounded through $\epsilon_k > 0$, particularly

$$\left\|\mathbf{r}_L^{(k)}\right\|_{\mathrm{L}_k^2(\Omega,W)}^2 \leq 2\,\epsilon_k^2 \quad \textit{and} \quad \left\|\mathbf{z}_L^{(k)} - \mathbf{z}^{(k)}\right\|_{\mathrm{L}_k^2(\Omega,W)}^2 \leq c_{\mathcal{G}}\,(1-\theta)\,\epsilon_k^2, \tag{5.5}$$

with $\theta \in (0,1)$ and $c_{\mathcal{G}}$ from (2.15) at computational complexity

$$\mathrm{C}_k\left(E^{ML}[\mathbf{g}_L]\right) \lesssim \begin{cases} \epsilon_k^{-2} & \beta > \gamma, \\ \epsilon_k^{-2}(\log(\epsilon_k))^2 & \beta = \gamma, \\ \epsilon_k^{-2-(\gamma-\beta)/\alpha} & \beta < \gamma. \end{cases} \tag{5.6}$$

Proof. By $\nabla J(\mathbf{z}^{(k)}) = \lambda \mathbf{z}^{(k)} - \mathbb{E}_k[\mathbf{q}^{(k)}]$, (5.1) and the triangle inequality, we have

$$\begin{aligned} \left\|\mathbf{r}_L^{(k)}\right\|_{\mathrm{L}^2_k(\Omega,W)}^2 &= \mathbb{E}_k\left[\left\|\lambda \mathbf{z}_L^{(k)} - \lambda \mathbf{z}^{(k)} + \mathbb{E}_k[\mathbf{q}^{(k)}] - E^{\mathrm{ML}}[\mathbf{q}_L^{(k)}]\right\|_W^2\right] \\ &\leq 2\lambda^2 \left\|\mathbf{z}_L^{(k)} - \mathbf{z}^{(k)}\right\|_{\mathrm{L}^2_k(\Omega,W)}^2 + 2\left\|\mathbb{E}_k[\mathbf{q}^{(k)}] - E^{\mathrm{ML}}[\mathbf{q}_L^{(k)}]\right\|_{\mathrm{L}^2_k(\Omega,W)}^2 \end{aligned}$$

Since $\mathrm{err}_k^{\mathrm{MSE}} \coloneqq \mathbb{E}_k\left[\left\|\mathbb{E}_k[\mathbf{q}^{(k)}] - E^{\mathrm{ML}}[\mathbf{q}_L^{(k)}]\right\|_W^2\right]$, we have with the decomposition (4.2)

$$\begin{aligned} \tfrac{1}{2}\left\|\mathbf{r}_L^{(k)}\right\|_{\mathrm{L}^2_k(\Omega,W)}^2 \leq \sum_{\ell=0}^{L} M_\ell^{-1} \left\|\mathbf{p}_\ell^{(k)} - \mathbb{E}_k[\mathbf{p}_\ell^{(k)}]\right\|_{\mathrm{L}^2_k(\Omega,W)}^2 + \left\|\mathbb{E}_k[\mathbf{q}_L^{(k)} - \mathbf{q}^{(k)}]\right\|_W^2 \\ + \lambda^2 \left\|\mathbf{z}_L^{(k)} - \mathbf{z}^{(k)}\right\|_{\mathrm{L}^2_k(\Omega,W)}^2 \end{aligned}$$

Considering the left side of assumption (2.15) and applying Jensen's inequality gives

$$\tfrac{1}{2}\left\|\mathbf{r}_L^{(k)}\right\|_{\mathrm{L}^2_k(\Omega,W)}^2 \leq \sum_{\ell=0}^{L} M_\ell^{-1} \left\|\mathbf{p}_\ell^{(k)} - \mathbb{E}_k[\mathbf{p}_\ell^{(k)}]\right\|_{\mathrm{L}^2_k(\Omega,W)}^2 + (\lambda^2 c_{\mathcal{G}} + 1)\, \mathbb{E}_k\left[\left\|\mathbf{q}_L^{(k)} - \mathbf{q}^{(k)}\right\|_W^2\right]$$

Thus, with (2.12b) and (2.15) both error contributions are controlled and similar as in [6, 25], we can achieve with some bias-variance trade-off $\theta \in (0,1)$

$$\sum_{\ell=0}^{L} M_\ell^{-1} \left\|\mathbf{p}_\ell^{(k)} - \mathbb{E}_k[\mathbf{p}_\ell^{(k)}]\right\|_{\mathrm{L}^2_k(\Omega,W)}^2 \leq \theta \epsilon_k^2 \quad \text{and} \quad \mathbb{E}_k\left[\left\|\mathbf{q}_L^{(k)} - \mathbf{q}^{(k)}\right\|_W^2\right] \leq \frac{(1-\theta)}{(\lambda^2 c_{\mathcal{G}} + 1)}\, \epsilon_k^2$$

through an appropriate choice of $\{M_\ell^{(k)}\}_{\ell=0}^L$ such that (5.6) holds. □

Theorem 5.2 (Convergence and ϵ-Cost of MLSGD)**.** *There exists a sequence of multilevel batches $\left\{\{M_{k,\ell}\}_{\ell=0}^L\right\}_{k=0}^{K-1}$, step sizes $\{t_k\}_{k=0}^{K-1} \subset \mathbb{R}_{\geq 0}$, and some $\rho \in (0,1)$, such that*

$$e_K \coloneqq \left\|\mathbf{z}_L^{(K)} - \mathbf{z}^*\right\|_{\mathrm{L}^2(\Omega,W)} \quad \textit{converges linearly, i.e.,} \quad e_K \leq \mathcal{O}\left(\rho^K\right). \tag{5.7}$$

Further, reaching an error of $e_K < \epsilon$ smaller than accuracy $\epsilon > 0$ comes at the cost

$$\mathrm{C}_\epsilon \lesssim \begin{cases} \epsilon^{-2} & \beta > \gamma, \\ \epsilon^{-2}(\log(\epsilon))^2 & \beta = \gamma, \\ \epsilon^{-2-(\gamma-\beta)/\alpha} & \beta < \gamma. \end{cases}$$

Proof. The proof is organized in two steps, first we show the linear convergence and then we estimate the cost of the algorithm. We define for the further proof

$$e_{k+1} \coloneqq \left\|\mathbf{z}_L^{(k+1)} - \mathbf{z}^*\right\|_{\mathrm{L}^2_k(\Omega,W)}, \quad e_k \coloneqq \left\|\mathbf{z}_L^{(k)} - \mathbf{z}^*\right\|_{\mathrm{L}^2_k(\Omega,W)}.$$

(i). Adapting the arguments of [3] to the iteration scheme (5.1) gives

$$\begin{aligned} e_{k+1}^2 &= \big\|\pi_Z\big(\mathbf{z}_L^{(k)} - t_k E^{\mathrm{ML}}[\mathbf{g}_L^{(k)}]\big) - \mathbf{z}^*\big\|_{\mathrm{L}_k^2(\Omega,W)}^2 \\ &\le \big\|\mathbf{z}_L^{(k)} - t_k E^{\mathrm{ML}}[\mathbf{g}_L^{(k)}] - \mathbf{z}^*\big\|_{\mathrm{L}_k^2(\Omega,W)}^2 \\ &= e_k^2 - 2t_k\big\langle \mathbf{z}_L^{(k)} - \mathbf{z}^*, E^{\mathrm{ML}}[\mathbf{g}_L^{(k)}]\big\rangle_{\mathrm{L}_k^2(\Omega,W)} + t_k^2\big\|E^{\mathrm{ML}}[\mathbf{g}_L^{(k)}]\big\|_{\mathrm{L}_k^2(\Omega,W)}^2 \end{aligned}$$

By (5.4), it is $E^{\mathrm{ML}}[\mathbf{g}_L^{(k)}] = \mathbf{r}_L^{(k)} + \nabla J(\mathbf{z}^{(k)})$ and we can write

$$e_{k+1}^2 \le e_k^2 - 2t_k\big\langle \mathbf{z}_L^{(k)} - \mathbf{z}^*, \mathbf{r}_L^{(k)} + \nabla J(\mathbf{z}^{(k)})\big\rangle_{\mathrm{L}_k^2(\Omega,W)} + t_k^2\big\|\mathbf{r}_L^{(k)} + \nabla J(\mathbf{z}^{(k)})\big\|_{\mathrm{L}_k^2(\Omega,W)}^2.$$

Since J is μ-strongly convex (2.3) and $\nabla J(\mathbf{z}^*) - \nabla J(\mathbf{z}_L^{(k)}) + \nabla J(\mathbf{z}_L^{(k)}) = 0$, it follows

$$\begin{aligned} -\big\langle \mathbf{z}_L^{(k)} - \mathbf{z}^*, \nabla J(\mathbf{z}^{(k)})\big\rangle_{\mathrm{L}_k^2(\Omega,W)} &\le -\mu e_k^2 + e_k\big\|\nabla J(\mathbf{z}^{(k)}) - \nabla J(\mathbf{z}_L^{(k)}))\big\|_{\mathrm{L}_k^2(\Omega,W)}, \\ -\big\langle \mathbf{z}_L^{(k)} - \mathbf{z}^*, \mathbf{r}_L^{(k)}\big\rangle_{\mathrm{L}_k^2(\Omega,W)} &\le e_k\,\big\|\mathbf{r}_L^{(k)}\big\|_{\mathrm{L}_k^2(\Omega,W)}. \end{aligned}$$

Therefore, we can estimate

$$\begin{aligned} e_{k+1}^2 \le\ & e_k^2 + 2t_k\big(e_k\,\big\|\mathbf{r}_L^{(k)}\big\|_{\mathrm{L}_k^2(\Omega,W)} + e_k\big\|\nabla J(\mathbf{z}^{(k)}) - \nabla J(\mathbf{z}_L^{(k)}))\big\|_{\mathrm{L}_k^2(\Omega,W)} - \mu e_k^2\big) \\ & + 2t_k^2\big(\big\|\mathbf{r}_L^{(k)}\big\|_{\mathrm{L}_k^2(\Omega,W)}^2 + \big\|\nabla J(\mathbf{z}^{(k)})\big\|_{\mathrm{L}_k^2(\Omega,W)}^2\big) \end{aligned}$$

Further, by (2.5) and (2.6) we see that

$$\begin{aligned} \big\|\nabla J(\mathbf{z}^{(k)})\big\|_{\mathrm{L}_k^2(\Omega,W)}^2 &= \big\|\nabla J(\mathbf{z}^{(k)}) - \nabla J(\mathbf{z}^*)\big\|_{\mathrm{L}_k^2(\Omega,W)}^2 \le c_{\mathrm{Lip}}^2\big\|\mathbf{z}^{(k)} - \mathbf{z}^*\big\|_{\mathrm{L}_k^2(\Omega,W)}^2 \\ &\le 2c_{\mathrm{Lip}}^2\big(e_k^2 + \big\|\mathbf{z}^{(k)} - \mathbf{z}_L^{(k)}\big\|_{\mathrm{L}_k^2(\Omega,W)}^2\big) \end{aligned}$$

and since the true gradient in step k is given by $\nabla J(\mathbf{z}^{(k)}) = \lambda\mathbf{z}^{(k)} - \mathbb{E}_k[\mathbf{q}^{(k)}]$, it is

$$\big\|\nabla J(\mathbf{z}^{(k)}) - \nabla J(\mathbf{z}_L^{(k)}))\big\|_{\mathrm{L}_k^2(\Omega,W)} \le \big\|\mathbb{E}_k[\mathbf{q}^{(k)} - \mathbf{q}_L^{(k)}]\big\|_{\mathrm{L}_k^2(\Omega,W)} + \lambda\big\|\mathbf{z}^{(k)} - \mathbf{z}_L^{(k)}\big\|_{\mathrm{L}_k^2(\Omega,W)}$$

In conclusion, the estimate for e_{k+1}^2 is given by

$$\begin{aligned} e_{k+1}^2 \le\ & e_k^2 + 2t_k e_k\big\|\mathbf{r}_L^{(k)}\big\|_{\mathrm{L}_k^2(\Omega,W)} + 2t_k e_k\big\|\mathbb{E}_k[\mathbf{q}^{(k)} - \mathbf{q}_L^{(k)}]\big\|_{\mathrm{L}_k^2(\Omega,W)} \\ & + 2t_k e_k\lambda\big\|\mathbf{z}^{(k)} - \mathbf{z}_L^{(k)}\big\|_{\mathrm{L}_k^2(\Omega,W)} + 2t_k^2\big\|\mathbf{r}_L^{(k)}\big\|_{\mathrm{L}_k^2(\Omega,W)}^2 \\ & + 4t_k^2 c_{\mathrm{Lip}}^2\big\|\mathbf{z}^{(k)} - \mathbf{z}_L^{(k)}\big\|_{\mathrm{L}_k^2(\Omega,W)}^2 + 4t_k^2 c_{\mathrm{Lip}}^2 e_k^2 - 2t_k\mu e_k^2 \end{aligned}$$

Now, we apply Lemma 5.1 by choosing ϵ_k in every iteration k as $\eta\cdot e_k$ with a sufficiently small $\eta > 0$. Then, there exists in a sequence of multilevel batches $\big\{\{M_\ell^{(k)}\}_{\ell=0}^L\big\}_{k\in\mathbb{N}}$, such that

$$e_{k+1}^2 \le e_k^2\big((4c_{\mathrm{Lip}}^2(1+\eta^2 c_{\mathcal{G}}) + 2\eta^2)t_k^2 + 2(\sqrt{2}\eta + \eta + \sqrt{c_{\mathcal{G}}}\lambda\eta - \mu)t_k + 1\big) =: e_k^2\rho_k^2 \tag{5.8}$$

where we used $(1-\theta) \le 1$. By taking the square root of the above expression, it follows by induction that the error converges linearly, if $\rho_k < 1$ in each step. This can be achieved by choosing the step size t_k and the factor η, such that the quadratic expression of ρ_k in (5.8) is minimized and $t_k > 0$ holds. As a result, by taking the expectation and the tower property, the total error in the final step K is given by $\mathcal{O}(\rho^K)$, with $\rho \coloneqq \sup\{\rho_k \colon k = 1,\dots,K\}$.

(ii). We denote with C_k again the cost of the k-th descent. By Lemma 5.1, the summed cost C_ϵ to achieve an total error of $e_K < \epsilon$ is given by

$$\mathrm{C}_\epsilon = \sum_{k=0}^{K} \mathrm{C}_k \lesssim \sum_{k=0}^{K} \begin{cases} \epsilon_k^{-2} & \beta > \gamma, \\ \epsilon_k^{-2} \log(\epsilon_k)^2 & \beta = \gamma, \\ \epsilon_k^{-2-(\gamma-\beta)/\alpha} & \beta < \gamma. \end{cases}$$

Since we chose the multilevel batch such that $\eta^{-1}\,\epsilon_k$ equals the error in each step, and since $e_k = \mathcal{O}(\rho^k)$, we get similar to [22] for large $K \gg 1$ the asymptotic behavior

$$\mathrm{C}_\epsilon \lesssim \sum_{k=0}^{K} \begin{cases} \rho^{-2k} \\ \rho^{-2k}(k\log(\rho))^2 \\ \rho^{-2k-(\gamma-\beta)k/\alpha} \end{cases} \sim \begin{cases} \rho^{-2K-2} \\ K^2\,\rho^{-2K-2} \\ \rho^{-2(K+1)-(\gamma-\beta)(K+1)/\alpha} \end{cases} \sim \begin{cases} \epsilon^{-2} & \beta > \gamma, \\ \epsilon^{-2}(\log(\epsilon))^2 & \beta = \gamma, \\ \epsilon^{-2-(\gamma-\beta)/\alpha} & \beta < \gamma, \end{cases}$$

using the geometric series and $\epsilon = \mathcal{O}(\rho^K)$. □

Remark 5.3. (1) *Even though MLSGD improves the rate from $\mathcal{O}(K^{-1/2})$ to linear convergence $\mathcal{O}(\rho^K)$, as discussed in Section 3.3, similar results can be achieved with large enough batches and gradient aggregation methods. However, as pointed out in* [28] *and observed in Section 3.2, this comes with an increased computational cost. Our approach, leveraging the MLMC method, reduces the cost of the gradient estimation dramatically, as demonstrated numerically in Section 5.2 and now proven by Theorem 5.2.*

(2) *The parameter $\eta > 0$ has to be chosen small enough, that minimizing (5.8) is achieved with $t_k > 0$. Since $\eta < \frac{\mu}{\sqrt{2}+1+\sqrt{c_{\mathcal{G}}\lambda}}$ is independent of k, this suggests the usage of an adaptive algorithm that successively enlarges the multilevel batch. In particular, as the algorithm runs and e_k decreases, this leads with $\epsilon_k = \eta \cdot e_k$ to a decreasing target error ϵ_k and thus to an increased computational cost of each iteration.*

The following corollary motivates implementing the targeted multilevel error by $\epsilon_k = \eta \cdot \|E^{\mathrm{ML}}[\mathbf{g}_L^{(k)}]\|_W$. This idea is similar to [22] and later used in Algorithm 3.

Corollary 5.4. *There exists a sequence of multilevel batches $\big\{\{M_\ell^{(k)}\}_{\ell=0}^{L}\big\}_{k=0}^{K-1}$, step sizes $\{t_k\}_{k=0}^{K-1} \subset \mathbb{R}_{\geq 0}$, and some $\rho \in (0,1)$, such that*

$$\mathbb{E}\big[J(\mathbf{z}_L^{(K)}) - J(\mathbf{z}^*)\big] \leq \mathcal{O}(\rho^K) \quad \text{and} \quad \big\|E^{\mathrm{ML}}[\mathbf{g}_L^{(K)}]\big\|_{\mathrm{L}^2(\Omega,W)} \leq \mathcal{O}(\rho^K)\,.$$

Proof. The first expression follows from the strong μ-convexity (2.4) together with (2.5)

$$\begin{aligned} J(\mathbf{z}_L^{(K)}) - J(\mathbf{z}^*) &\leq \big\langle \nabla J[\mathbf{z}_L^{(K)}], \mathbf{z}_L^{(K)} - \mathbf{z}^* \big\rangle_W - \tfrac{\mu}{2}\big\|\mathbf{z}_L^{(K)} - \mathbf{z}^*\big\|_W^2 \\ &\leq (c_{\mathrm{Lip}} - \tfrac{\mu}{2})\big\|\mathbf{z}_L^{(K)} - \mathbf{z}^*\big\|_W^2 \end{aligned}$$

and by taking the expectation on both sides. The second expression follows from bounding $\big\|\nabla J(\mathbf{z}^{(k)})\big\|_{\mathrm{L}_k^2(\Omega,W)}$ and $\big\|\mathbf{r}_L^{(k)}\big\|_{\mathrm{L}_k^2(\Omega,W)}$ in (5.4) with similar arguments as in Theorem 5.2. Taking the expectation over all possible optimization trajectories concludes the result. □

6 Adaptivity and Budgeting

Having established the MLSGD method, particularly its convergence in Theorem 5.2 and its functionality in Algorithm 2, we now address the question of how to choose the multilevel batches $\big\{\{M_{k,\ell}\}_{\ell=0}^{L}\big\}_{k=0}^{K-1}$ and step sizes $\{t_k\}_{k=0}^{K-1}$ given computational constraints such as memory limitations Mem_0 and CPU-time budgets $|\mathcal{P}|\cdot \mathrm{T}_0$, T_0 being the reserved time budget on the cluster and $|\mathcal{P}|$ being the cardinality of the set of processing units. Building upon [13, 14], we formulate the following knapsack problem, which is then solved with the budgeted multilevel stochastic gradient descent (BMLSGD) method, adaptively determining the multilevel batches and step sizes.

Problem 6.1 (MLSGD Knapsack)**.** *Suppose the solution* $\mathbf{z}^*$ *to the optimal control problem* (2.1) *is given, find the optimal sequence of step sizes* $\{t_k\}_{k=0}^{K-1}$ *and multilevel batches* $\big\{\{M_{k,\ell}\}_{\ell=0}^{L_k}\big\}_{k=0}^{K-1}$*, such that the total error* e_K *is minimized, while staying within a CPU-time* $|\mathcal{P}|\cdot \mathrm{T}_0$ *and memory* Mem_0 *budget*

$$\min_{\left\{t_k,\{M_{k,\ell}\}_{\ell=0}^{L_k}\right\}_{k=0}^{K}} \left\|\mathbf{z}_L^{(K)} - \mathbf{z}^*\right\|_{\mathrm{L}^2(\Omega,W)} \tag{6.1a}$$

$$\sum_{k=0}^{K-1}\sum_{\ell=0}^{L_k}\sum_{m=1}^{M_{k,\ell}} \mathrm{C}^{\mathrm{CT}}\big((\mathbf{v}_\ell^{(m,k)},\mathbf{p}_\ell^{(m,k)})\big) \leq |\mathcal{P}|\cdot \mathrm{T}_0 \tag{6.1b}$$

$$\mathrm{C}^{\mathrm{Mem}}\big((\mathbf{v}_L^{(K-1)},\mathbf{p}_L^{(K-1)})\big) < \mathrm{Mem}_0. \tag{6.1c}$$

Problem 6.1 is an NP-hard integer resource allocation problem which we solve with distributed dynamic programming techniques utilizing an optimal policy to find the multilevel batches in each iteration of the optimization algorithm. To respect the cost constraints (6.1b) and (6.1c), we utilize assumptions (2.13) and (2.14). Before we present the BMLSGD method in Algorithm 3, we can conclude with similar arguments as in [13, 14] that the error of every feasible solution is bounded from below by the memory constraint and from above by the CPU-time budget.

Corollary 6.2 (Lower and Upper bound of MLSGD)**.** *The minimum* (6.1a) *is bounded through the imposed constraints* (6.1b) *and* (6.1c) *as*

$$\mathrm{Mem}_0^{-\alpha} < \left\|\mathbf{z}_L^{(K)} - \mathbf{z}^*\right\|_W \lesssim (1-\lambda_{\mathrm{p}})\mathrm{T}_0^{-\delta} + \lambda_{\mathrm{p}}(|\mathcal{P}|\cdot \mathrm{T}_0)^{-\delta}. \tag{6.2}$$

Here, $\delta = \min\left\{\frac{1}{2}, \frac{\alpha}{2\alpha+(\gamma-\beta)}\right\}$ *is the convergence rate with respect to the computational resources and* $\lambda_{\mathrm{p}} \in [0,1]$ *is the parallelizable percentage of the code.*

Proof. Applying the arguments of [13] for the right-hand side, i.e., combining the inverted complexity of Theorem 5.2 with Gustafson's law, and considering for the left-hand side that the smallest possible bias is determined through the maximal memory footprint as in [14], gives the result. □

6.1 Algorithm

Building upon Algorithm 1 and Algorithm 2, we outline again the method as functional pseudocode. Since Problem 6.1 is intractable, we use the estimators given in Section 4 in combination with the theory from Section 5.3. We define the global variables for the leftover time budget T_k and the total available memory Mem_k (we store the history of these values along the optimization steps k) and initialize them with the given T_0 and Mem_0 from Problem 6.1. We further set the constants for the error target reduction $\eta \in (0,1]$ and the bias-variance tradeoff $\theta \in (0,1)$ to start the algorithm.

The Init function. Starting the call stack in the `Init` function with an initial guess on $\mathbf{z}_L^{(0)}$, an initial multilevel batch $\{M_{0,\ell}\}_{\ell=0}^{L_0}$, and the initial step size t_0, the `MultiLevelEstimation` function and the `GradientDescent` function from Algorithm 2 are called. This gives the initial estimates on the adjoint $E^{\mathrm{ML}}[\mathbf{q}_L^{(0)}]$, on the initial objective $J^{\mathrm{ML}}(\mathbf{z}_L^{(0)})$ and with that, the gradient $E^{\mathrm{ML}}[\mathbf{g}_L^{(0)}]$ and the first updated control $\mathbf{z}_L^{(1)}$. The computational cost is dominated by the `MultiLevelEstimation` function, thus after its first call, the time budget and the leftover memory are updated with $\mathrm{T}_1 \leftarrow \mathrm{T}_0 - \mathrm{C}_0^{\mathrm{CT}}$ and $\mathrm{Mem}_1 \leftarrow \mathrm{Mem}_0 - \mathrm{C}_0^{\mathrm{Mem}}$, where the cost measurements $\mathrm{C}_k^{\mathrm{CT}}$ and $\mathrm{C}_k^{\mathrm{Mem}}$ are given by

$$\mathrm{C}_k^{\mathrm{CT}} = \sum_{\ell=0}^{L_k} \sum_{m=1}^{M_{k,\ell}} \mathrm{C}^{\mathrm{CT}}\big((\mathbf{v}_\ell^{(m,k)}, \mathbf{p}_\ell^{(m,k)})\big) \qquad \text{and} \qquad \mathrm{C}_k^{\mathrm{Mem}} \sim m_k h_{L_k}^{-\gamma_{\mathrm{Mem}}}. \tag{6.3}$$

We refer to [14] for a more detailed discussion on the time cost measurement in parallel and why the memory cost can be bounded through the highest level L_k following the assumption (2.14). Subsequently, we call the `BMLSGD` function with the new control $\mathbf{z}_L^{(1)}$ and the first error target $\epsilon_1 \leftarrow \eta \cdot \big\|E^{\mathrm{ML}}[\mathbf{g}_L^{(0)}]\big\|_W$ (cf. [13] for a discussion why η is relevant for the parallel efficiency).

The BMLSGD function. The `BMLSGD` function in Algorithm 3 acts as the main driver of the algorithm by recursively minimizing (6.1a), while enforcing the computational constraints (6.1b) and (6.1c). This involves checking the leftover time budget T_k in a guard clause of the function, and adapting the largest level L_k if the FE error, estimated with (4.4), is larger than a predefined fraction of the target error $\theta\,\epsilon_k$. To find the optimal multilevel batch $\{M_{k,\ell}^{\mathrm{opt}}\}_{\ell=0}^{L_k}$ in each iteration k, we apply techniques introduced in [7, 14] utilizing estimates from the previous optimization step $k-1$. Particularly, we use

$$M_{k,\ell}^{\mathrm{opt}} = \left\lceil \left(\sqrt{\theta}\epsilon_k\right)^{-2} \sqrt{\frac{s_\ell^2[\mathbf{p}_\ell^{(k-1)}]}{(M_{k-1,\ell}-1)\,\widehat{\mathrm{C}}_{k-1,\ell}^{\mathrm{CT}}}} \left(\sum_{\ell'=0}^{L_k} \sqrt{\frac{s_{\ell'}^2[\mathbf{p}_{\ell'}^{(k-1)}]\,\widehat{\mathrm{C}}_{k-1,\ell'}^{\mathrm{CT}}}{M_{k-1,\ell'}-1}} \right) \right\rceil \tag{6.4}$$

to get the optimal number of samples $M_{k,\ell}^{\mathrm{opt}}$ on each level ℓ where $s_\ell^2[\mathbf{p}_\ell^{(k-1)}]$ is the second order sum from (4.5). The estimate $\widehat{\mathrm{C}}_{k-1,\ell}^{\mathrm{CT}}$ is the averaged cost of the previous iteration $k-1$ on level ℓ, thus with the sample size $M_{k-1,\ell}$, it is

$$\widehat{\mathrm{C}}_{k-1,\ell}^{\mathrm{CT}} = \frac{1}{M_{k-1,\ell}} \sum_{m=1}^{M_{k-1,\ell}} \mathrm{C}^{\mathrm{CT}}\big((\mathbf{v}_\ell^{(m,k-1)}, \mathbf{p}_\ell^{(m,k-1)})\big).$$

Once the new multilevel batch $\{M_{k,\ell}^{\mathrm{opt}}\}_{\ell=0}^{L_k}$ is chosen, the `NotFeasible` function is called. This functions stops the computation if the new batch is expected to take longer than the leftover time budget T_k or, if, by appending a new level, the memory constraint is violated. Else, we call the `MultiLevelEstimation` function to compute new estimates on the adjoint $E^{\mathrm{ML}}[\mathbf{q}_L^{(k)}]$ and the objective $J^{\mathrm{ML}}(\mathbf{z}_L^{(k)})$ using the new multilevel batch $\{M_{k,\ell}^{\mathrm{opt}}\}_{\ell=0}^{L_k}$. We then determine the new control $\mathbf{z}_L^{(k+1)}$ by calling the `AdaptiveGradientDescent` function, update the budgets with (6.3), and start a new optimization step with the `BMLSGD` function taking the new control $\mathbf{z}_L^{(k+1)}$, and the new error target $\epsilon_{k+1} \leftarrow \eta \cdot \big\|E^{\mathrm{ML}}[\mathbf{g}_L^{(k)}]\big\|_W$ as input.

Feasibility check. Within the `NotFeasible` function, the upcoming computational cost is estimated and checked against the remaining budgets:

$$\underbrace{\sum_{\ell=0}^{L_k} M_{k,\ell}^{\mathrm{opt}} \widehat{\mathrm{C}}_{k-1,\ell}^{\mathrm{CT}} > \mathrm{T}_k}_{\text{new batch is too expensive in CPU-time}} \qquad \text{or} \qquad \underbrace{m_k h_{L_k}^{-\gamma_{\mathrm{Mem}}} > \mathrm{Mem}_k}_{\text{new batch is too expensive in memory}}$$

We recall that Mem_k is the leftover memory, i.e. the difference between the initial memory budget Mem_0 and the memory footprint of the last computational result. In either case, if the new multilevel batch is too expensive in CPU-time or memory, the algorithm stops and returns the current control as the solution to Problem 6.1.

Adaptive Gradient Descent. By Theorem 5.2, we do have to determine besides the optimal multilevel batch $\{M_{k,\ell}^{\mathrm{opt}}\}_{\ell=0}^{L_k}$ also the step size t_k in order to minimize (5.8). To this end, we use the adaptive step size techniques introduced in [31] in combination with the already computed MLMC estimates and the gradient $E^{\mathrm{ML}}[\mathbf{g}_L^{(k)}]$ used in (5.1). Particularly, we compute

$$t_k = \frac{\|E^{\mathrm{ML}}[\mathbf{g}_L^{(k)}]\|_W^2 - \widehat{\mathrm{err}}_k^{\mathrm{sam}}}{\widehat{c}_{\mathrm{Lip}} \|E^{\mathrm{ML}}[\mathbf{g}_L^{(k)}]\|_W^2} \quad \text{with} \quad \widehat{c}_{\mathrm{Lip}} = \frac{\|E^{\mathrm{ML}}[\mathbf{g}_L^{(k)}] - E^{\mathrm{ML}}[\mathbf{g}_L^{(k-1)}]\|_W}{\|t_{k-1} E^{\mathrm{ML}}[\mathbf{g}_L^{(k-1)}]\|_W}, \tag{6.5}$$

where $\widehat{\mathrm{err}}_{\mathrm{sam}}$ is determined with (4.5), and $\widehat{c}_{\mathrm{Lip}}$ is an estimate of the Lipschitz constant following (2.5) which takes the previous gradient $E^{\mathrm{ML}}[\mathbf{g}_L^{(k-1)}]$ from memory. With the new step size t_k, we finally compute the new control $\mathbf{z}_L^{(k+1)}$ following the update rule in (5.1) and return to the `BMLSGD` function again.

6.2 Experiments

We now present our final numerical results, evaluating the performance of Algorithm 3 on the PDE example from Section 2.3. In particular, we focus on estimating the convergence rate δ as defined in Corollary 6.2. Following the approach of [13], we estimate δ via linear regression on the logarithmic decay of the gradient norm over time:

$$\min_{(\widehat{\delta}, \widehat{c}_\delta)} \sum_k \left(\log_2 \left(\left\| E^{\mathrm{ML}}[\mathbf{g}_L^{(k)}] \right\|_W \right) - \widehat{\delta} \log_2(\mathrm{T}_k) + \widehat{c}_\delta \right)^2$$

All figures in this section display estimates of δ. To clarify the plots, we represent data points directly (excluding early iterations within the first 60 seconds) along with the corresponding linear fit.

Method comparison. Figure 5 presents a direct comparison between the BMLSGD and BSGD methods. To improve the BSGD performance, we used a decaying step size $t_k = t_0 \cdot K^{-0.5}$ satisfying (2.9) with $t_0 = 250$ and increased the iteration count to $K = 150$. With $h_\ell = 2^{-7}$ and $M_\ell = 256$, this configuration gave the best BSGD results within one hour of computation. Despite these optimizations, BMLSGD significantly outperforms BSGD on our example — achieving comparable results 18× faster and yielding errors 5× smaller for the same computational cost. As shown in Figure 5, BMLSGD also achieves a markedly better convergence rate of $\delta \approx 0.5$, compared to $\delta \approx 0.37$ for BSGD. This aligns with theoretical expectations: for BSGD, $\delta = \frac{\alpha}{2\alpha+\gamma}$ (cf. discussion in Sections 3.3 and 4), while for BMLSGD, Corollary 6.2 gives $\delta = \min\{\frac{1}{2}, \frac{\alpha}{2\alpha+(\gamma-\beta)}\}$. Finally, BMLSGD achieves these improvements with fewer iterations, lower bias, and reduced noise (see left plot of Figure 5).

Unlike in Sections 3.2 and 5.2, as well as the preceding discussion, Figures 6 and 7 additionally present results on multilevel statistics and highlight the decay of the gradient $\|E^{\mathrm{ML}}[\mathbf{g}_L^{(k)}]\|_W$ over time. The aim is to justify Assumption 2.3 by estimating the exponents $\alpha_{\mathbf{u}}, \alpha_{\mathbf{q}}, \beta_{\mathbf{v}}, \beta_{\mathbf{p}}, \gamma_{\mathrm{CT}}$, and γ_{Mem}. The top row in both figures displays the experimentally measured exponents: for

Algorithm 3 Budgeted Multilevel Stochastic Gradient Descent (BMLSGD)

`global const` $(\eta,\, \theta)^\top \leftarrow (0.9, 0.5)^\top$ // Tested choice for both values

`global` $(\mathrm{T}_k,\, \mathrm{Mem}_k)^\top \leftarrow (\mathrm{T}_0,\, \mathrm{Mem}_0)^\top$ // Initialize with time and memory budget

`function Init`$(\mathbf{z}_L^{(0)}, \{M_{0,\ell}\}_{\ell=0}^{L}, t_0)$: // Calls functions from Algorithm 2

$$\begin{cases} E^{\mathrm{ML}}[\mathbf{q}_L^{(0)}],\ J^{\mathrm{ML}}(\mathbf{z}_L^{(0)}) & \leftarrow & \texttt{MultiLevelEstimation}(\mathbf{z}_L^{(0)}, \{M_{0,\ell}\}_{\ell=0}^{L}) \\ \mathbf{z}_L^{(1)},\ E^{\mathrm{ML}}[\mathbf{g}_L^{(0)}] & \leftarrow & \texttt{GradientDescent}(\mathbf{z}_L^{(0)}, E^{\mathrm{ML}}[\mathbf{q}_L^{(0)}], t_0) \\ (\mathrm{T}_1,\, \mathrm{Mem}_1)^\top & \leftarrow & (\mathrm{T}_0 - \mathrm{C}_0^{\mathrm{CT}},\, \mathrm{Mem}_0 - \mathrm{C}_0^{\mathrm{Mem}})^\top \\ \texttt{return} & & \texttt{BMLSGD}(\mathbf{z}_L^{(1)},\, \eta \cdot \left\| E^{\mathrm{ML}}[\mathbf{g}_L^{(0)}] \right\|_W) \end{cases}$$

`function BMLSGD`$(\mathbf{z}_L^{(k)},\, \epsilon_k)$:

$$\begin{cases} \texttt{if}\ \mathrm{T}_k < 0.05\,\mathrm{T}_0\texttt{:} & \texttt{return}\ \mathbf{z}_L^{(k)} \\ \texttt{if}\ \widehat{\mathrm{err}}_{k-1}^{\mathrm{num}} \geq (1-\theta)\,\epsilon_k^2\texttt{:} & L_k \leftarrow L_{k-1} + 1 \\ \texttt{for}\ \ell = 0, \ldots, L_k\texttt{:} & M_{k,\ell}^{\mathrm{opt}} \leftarrow (6.4) \\ \texttt{if NotFeasible}(\{M_{k,\ell}^{\mathrm{opt}}\}_{\ell=0}^{L_k})\texttt{:} & \texttt{return}\ \mathbf{z}_L^{(k)} \end{cases}$$

$$\begin{cases} E^{\mathrm{ML}}[\mathbf{q}_L^{(k)}],\ J^{\mathrm{ML}}(\mathbf{z}_L^{(k)}) & \leftarrow & \texttt{MultiLevelEstimation}(\mathbf{z}_L^{(k)}, \{M_{k,\ell}^{\mathrm{opt}}\}_{\ell=0}^{L_k}) \\ \mathbf{z}_L^{(k+1)},\ E^{\mathrm{ML}}[\mathbf{g}_L^{(k)}] & \leftarrow & \texttt{AdaptiveGradientDescent}(\mathbf{z}_L^{(k)}, E^{\mathrm{ML}}[\mathbf{q}_L^{(k)}]) \\ (\mathrm{T}_{k+1},\, \mathrm{Mem}_{k+1})^\top & \leftarrow & (\mathrm{T}_k - \mathrm{C}_k^{\mathrm{CT}},\, \mathrm{Mem}_k - \mathrm{C}_k^{\mathrm{Mem}})^\top \\ \texttt{return} & & \texttt{BMLSGD}(\mathbf{z}_L^{(k+1)},\, \eta \cdot \left\| E^{\mathrm{ML}}[\mathbf{g}_L^{(k)}] \right\|_W) \end{cases}$$

`function NotFeasible`$(\{M_{k,\ell}^{\mathrm{opt}}\}_{\ell=0}^{L_k})$:

$$\begin{cases} \texttt{if}\ \sum_{\ell=0}^{L_k} M_{k,\ell}^{\mathrm{opt}} \widehat{\mathrm{C}}_{k-1,\ell}^{\mathrm{CT}} > \mathrm{T}_k\texttt{:} & \texttt{return true} \\ \texttt{if}\ m_k h_{L_k}^{-\gamma_{\mathrm{Mem}}} > \mathrm{Mem}_k\texttt{:} & \texttt{return true} \\ \texttt{else:} & \texttt{return false} \end{cases}$$

`function AdaptiveGradientDescent`$(\mathbf{z}_L^{(k)}, E^{\mathrm{ML}}[\mathbf{q}_L^{(k)}])$:

$$\begin{cases} E^{\mathrm{ML}}[\mathbf{g}_L^{(k)}] & \leftarrow & \lambda \mathbf{z}_L^{(k)} - E^{\mathrm{ML}}[\mathbf{q}_L^{(k)}] \\ \widehat{c}_{\mathrm{Lip}} & \leftarrow & \frac{\|E^{\mathrm{ML}}[\mathbf{g}_L^{(k)}] - E^{\mathrm{ML}}[\mathbf{g}_L^{(k-1)}]\|_W}{\|t_{k-1} E^{\mathrm{ML}}[\mathbf{g}_L^{(k-1)}]\|_W} & // \ t_{k-1}, E^{\mathrm{ML}}[\mathbf{g}_L^{(k-1)}] \text{ from memory} \\ t_k & \leftarrow & \frac{\|E^{\mathrm{ML}}[\mathbf{g}_L^{(k)}]\|_W^2 - \widehat{\mathrm{err}}_k^{\mathrm{sam}}}{\widehat{c}_{\mathrm{Lip}} \|E^{\mathrm{ML}}[\mathbf{g}_L^{(k)}]\|_W^2} & // \ \widehat{\mathrm{err}}_k^{\mathrm{sam}} \text{ estimated with (4.5)} \\ \mathbf{z}_L^{(k+1)} & \leftarrow & \pi_Z\big(\mathbf{z}_L^{(k)} - t_k E^{\mathrm{ML}}[\mathbf{g}_L^{(k)}]\big) \\ \texttt{return} & & \mathbf{z}^{(k+1)},\ E^{\mathrm{ML}}[\mathbf{g}_L^{(k)}] \end{cases}$$

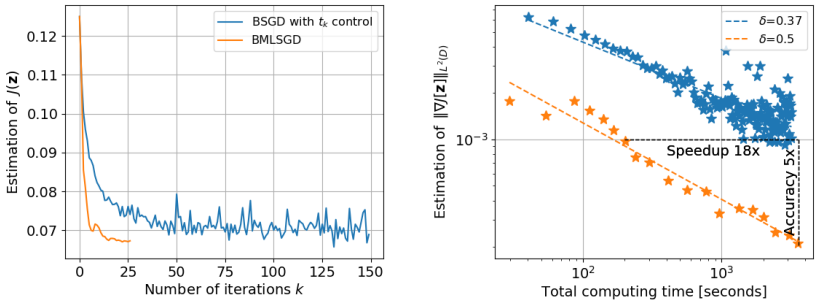


FIGURE 5. Comparison of Algorithm 1 with Algorithm 3.

the state (dash-dotted line) and the adjoint solutions (dashed line) in the left and center plots, respectively. The rightmost plot shows the measured computational cost exponents γ^{CT} and γ^{Mem} in a dual-axis format: the increasing solid line indicates average computational costs per level, while the bar plot represents memory costs per level. The horizontal lines in this plot mark the total memory footprint in megabytes (MB). The slight overhead of memory costs at the lowest level stems from initialization of the algorithm and the solvers.

Step size choice. Following the approach of [13], we assess the multilevel results to identify the optimal step size rule under a fixed computational budget through direct performance comparisons. The algorithm is initialized with the multilevel batch configuration: $(h_0, M_0)^\top = (2^{-4}, 64)^\top$, $(h_1, M_1)^\top = (2^{-5}, 16)^\top$, and $(h_2, M_2)^\top = (2^{-6}, 4)^\top$. Performance is evaluated using the same settings as in Figure 5. In Figure 6, we compare fixed step sizes $t_k \equiv 100$ and $t_k \equiv 150$ with the adaptive step size rule (6.5) using $t_0 = 200$. The center plot in the second row of Figure 6 shows that all three configurations result in a comparable load distribution across levels and remain within the 1-hour computational budget (indicated by the horizontal red line). The lower right plot indicates that the adaptive step size yields the best convergence rate and lowest gradient norm at the end of the optimization. In the context of BMLSGD, the adaptive rule utilizes each batch optimally, thereby enables larger batches that lead to improved parallel efficiency.

Node scaling. Finally, we demonstrate that the BMLSGD method scales well with increased computational resources (cf. Figure 7). To this end, we run the method with adaptive step size and $t_0 = 200$ on $|\mathcal{P}| = 64$ (1 node), $|\mathcal{P}| = 256$ (4 nodes), and $|\mathcal{P}| = 1024$ (16 nodes). Notably, the method leverages the additional resources to compute more samples—and, for $|\mathcal{P}| = 1024$, also includes an additional level (cf. lower left plot of Figure 7 showing the total number of samples M_ℓ over the optimization). As shown in the lower right plot of Figure 7, and similarly observed in [13], increasing computational resources improves the solution quality. The smaller gap between the green ($|\mathcal{P}| = 1024$) and orange ($|\mathcal{P}| = 256$) lines, compared to that between the orange and blue ($|\mathcal{P}| = 64$), suggests diminishing parallel efficiency with more nodes. This is due to the inherently unparallelizable portion λ_{p} of the code (see Corollary 6.2 and [13] for further details).

7 Outlook and Conclusion

In this paper, we introduced the MLSGD method and its budgeted variant. We showed that the method solves optimal control problems of the form (2.1) with a linear convergence rate in the number of optimization steps. The total error is controlled via multilevel Monte Carlo estimation, as established in Lemma 5.1, Theorem 5.2, and Corollary 5.4. Our numerical

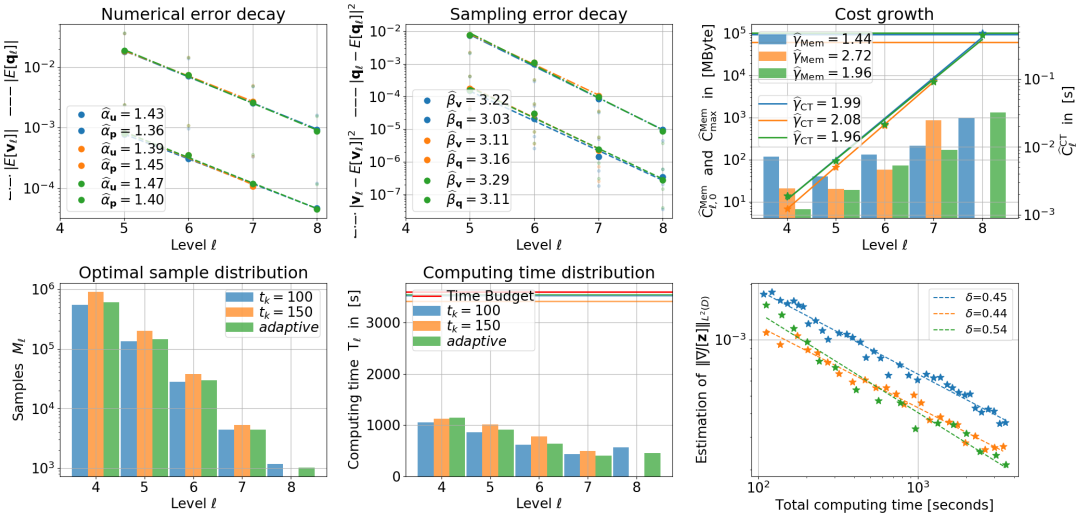


FIGURE 6. Comparison of $t_k \equiv 100$ and $t_k \equiv 150$ with the adaptive step size rule (6.5).

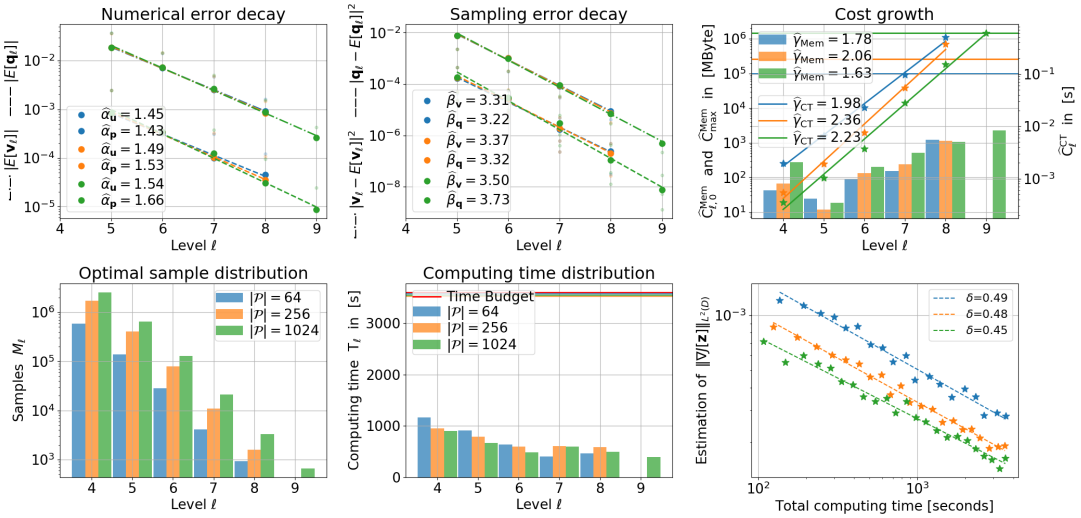


FIGURE 7. CPU-scaling experiment with $|\mathcal{P}| = 64$, $|\mathcal{P}| = 256$ and $|\mathcal{P}| = 1024$.

experiments in Section 5.2 and Section 6.2 clearly demonstrate the superior performance of MLSGD compared to the baseline BSGD method. In particular, the budgeted variant of Section 6 shows significant improvements in convergence speed, accuracy, parallel scalability, and robustness, while ensuring total error control relative to available CPU-time and memory, as shown in Corollary 6.2.

Future work may explore other PDEs, such as wave equations [13] or coupled PDEs [12], applications which are already supported by our software M++ [34]. The method is designed for high-dimensional problems and large computing clusters, where its inherent parallelism, based on [14], enables efficient usage of resource. It is thus applicable to three-dimensional problems, as illustrated by the outlook application in Figure 8. Algorithmically, the method can be

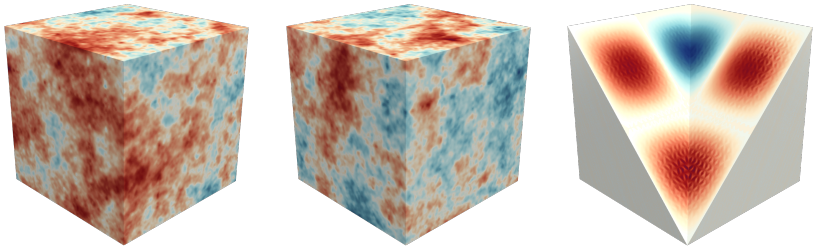


FIGURE 8. Outlook on three-dimensional random field sampling and optimal control.

extended in several directions, such as incorporating limited-memory BFGS methods [45] as an alternative to gradient descent, or combining it with QMC methods as in [10, 11]. Further improvements may also be achieved by refining the step size strategies and by using new averaging schemes [46]. We also suspect that the MLSGD method can excel in risk-averse PDE-constrained optimization using conditional value-at-risk (CVaR) estimation, as described in [47], potentially leveraging the multilevel techniques introduced in [48] to estimate the CVaR. Lastly, we want to point out again that the MLSGD method is not limited to PDE-constrained optimization in application. We note that simultaneously with our work, a related formulation to the MLSGD method appeared in [49], focusing on operator learning. We believe that the findings in our paper, e.g. how to realize adaptivity and parallel scalability, will improve future applications of the MLSGD method in machine learning.

Acknowledgments

We thank Christian Wieners, Sebastian Krumscheid and Robert Scheichl for their feedback and advice on the presented work. We further acknowledge the technical support by the National High-Performance Computing Center (NHR) at KIT and the possibility to access to the HoreKa supercomputer. LLMs were used to correct some spelling and grammar of this article. Funded by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) – Project-ID 258734477 – SFB 1173.

References

[1] C. Geiersbach and G. C. Pflug, "Projected stochastic gradients for convex constrained problems in Hilbert spaces," SIAM J. Optim., vol. 29, no. 3, pp. 2079–2099, 2019.
[2] C. Geiersbach and T. Scarinci, "A stochastic gradient method for a class of nonlinear pde-constrained optimal control problems under uncertainty," Journal of Differential Equations, vol. 364, pp. 635–666, 2023.
[3] C. Geiersbach and W. Wollner, "A stochastic gradient method with mesh refinement for PDE-constrained optimization under uncertainty," SIAM J. Sci. Comput., vol. 42, no. 5, pp. A2750–A2772, 2020.
[4] M. Martin, S. Krumscheid, and F. Nobile, "Complexity analysis of stochastic gradient methods for PDE-constrained optimal control problems with uncertain parameters," ESAIM Math. Model. Numer. Anal., vol. 55, no. 4, pp. 1599–1633, 2021.
[5] S. C. Toraman and H. Yücel, "A stochastic gradient algorithm with momentum terms for optimal control problems governed by a convection–diffusion equation with random diffusivity," Journal of Computational and Applied Mathematics, vol. 422, p. 114919, 2023.
[6] K. A. Cliffe, M. B. Giles, R. Scheichl, and A. L. Teckentrup, "Multilevel Monte Carlo methods and applications to elliptic PDEs with random coefficients," Comput. Vis. Sci., vol. 14, no. 1, pp. 3–15, 2011.
[7] M. B. Giles, "Multilevel Monte Carlo methods," Acta Numer., vol. 24, pp. 259–328, 2015.
[8] D. P. Kouri, "A multilevel stochastic collocation algorithm for optimization of PDEs with uncertain coefficients," SIAM/ASA J. Uncertain. Quantif., vol. 2, no. 1, pp. 55–81, 2014.

[9] F. Nobile and T. Vanzan, "A combination technique for optimal control problems constrained by random pdes," SIAM/ASA Journal on Uncertainty Quantification, vol. 12, no. 2, pp. 693–721, 2024.
[10] P. A. Guth, V. Kaarnioja, F. Y. Kuo, C. Schillings, and I. H. Sloan, "A quasi-Monte Carlo method for optimal control under uncertainty," SIAM/ASA J. Uncertain. Quantif., vol. 9, no. 2, pp. 354–383, 2021.
[11] P. A. Guth, V. Kaarnioja, F. Y. Kuo, C. Schillings, and I. H. Sloan, "Parabolic pde-constrained optimal control under uncertainty with entropic risk measure using quasi-monte carlo integration - numerische mathematik," SpringerLink, Mar 2024.
[12] N. Baumgarten, A Fully Parallelized and Budgeted Multi-level Monte Carlo Framework for Partial Differential Equations. PhD thesis, Karlsruher Institut für Technologie (KIT), 2023.
[13] N. Baumgarten, S. Krumscheid, and C. Wieners, "A fully parallelized and budgeted multilevel monte carlo method and the application to acoustic waves," SIAM/ASA Journal on Uncertainty Quantification, vol. 12, no. 3, pp. 901–931, 2024.
[14] N. Baumgarten, R. Kutri, and R. Scheichl, "A budgeted multi-level monte carlo method for full field estimates of multi-pde problems," 2025.
[15] L. Chen, M. Xiong, J. Ming, and X. He, "Efficient mini-batch stochastic gradient descent with centroidal voronoi tessellation for pde-constrained optimization under uncertainty," Physica D: Nonlinear Phenomena, vol. 467, p. 134216, 2024.
[16] C. Geiersbach, T. Suchan, and K. Welker, "Optimization of piecewise smooth shapes under uncertainty using the example of navier-stokes flow," arXiv preprint arXiv:2308.07742, 2023.
[17] R. Kutri and R. Scheichl, "Dirichlet-neumann averaging: The dna of efficient gaussian process simulation," arXiv preprint arXiv:2412.07929, 2024.
[18] R. Caflisch and Y. Yang, Adjoint Monte Carlo Method, pp. 461–505. Cham: Springer Nature Switzerland, 2024.
[19] A. A. Ali, E. Ullmann, and M. Hinze, "Multilevel monte carlo analysis for optimal control of elliptic pdes with random coefficients," SIAM/ASA Journal on Uncertainty Quantification, vol. 5, no. 1, pp. 466–492, 2017.
[20] G. Ciaramella, F. Nobile, and T. Vanzan, "A multigrid solver for pde-constrained optimization with uncertain inputs," Journal of Scientific Computing, vol. 101, no. 1, p. 13, 2024.
[21] P. A. Guth and A. Van Barel, "Multilevel quasi-monte carlo for optimization under uncertainty," Numerische Mathematik, vol. 154, no. 3-4, pp. 443–484, 2023.
[22] A. Van Barel and S. Vandewalle, "Robust optimization of pdes with random coefficients using a multilevel monte carlo method," SIAM/ASA Journal on Uncertainty Quantification, vol. 7, no. 1, pp. 174–202, 2019.
[23] H. Robbins and S. Monro, "A Stochastic Approximation Method," The Annals of Mathematical Statistics, vol. 22, no. 3, pp. 400 – 407, 1951.
[24] N. Frikha, "Multi-level stochastic approximation algorithms," The Annals of Applied Probability, vol. 26, no. 2, pp. 933 – 985, 2016.
[25] M. B. Giles, "Multilevel Monte Carlo path simulation," Oper. Res., vol. 56, no. 3, pp. 607–617, 2008.
[26] S. Weissmann, A. Wilson, and J. Zech, "Multilevel optimization for inverse problems," in Conference on Learning Theory, pp. 5489–5524, PMLR, 2022.
[27] S. Weissmann and J. Zech, "On the mean-field limit for stein variational gradient descent: stability and multilevel approximation," arXiv preprint arXiv:2402.01320, 2024.
[28] L. Bottou, F. E. Curtis, and J. Nocedal, "Optimization methods for large-scale machine learning," SIAM review, vol. 60, no. 2, pp. 223–311, 2018.
[29] S. Khirirat, H. R. Feyzmahdavian, and M. Johansson, "Mini-batch gradient descent: Faster convergence under data sparsity," in 2017 IEEE 56th Annual Conference on Decision and Control (CDC), pp. 2880–2887, IEEE, 2017.
[30] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," arXiv preprint arXiv:1412.6980, 2014.
[31] F. Köhne, L. Kreis, A. Schiela, and R. Herzog, "Adaptive step sizes for preconditioned stochastic gradient descent," 2024.
[32] B. Polyak and A. Juditsky, "Acceleration of stochastic approximation by averaging," SIAM Journal on Control and Optimization, vol. 30, pp. 838–855, 07 1992.
[33] O. Shamir and T. Zhang, "Stochastic gradient descent for non-smooth optimization: Convergence results and optimal averaging schemes," vol. 28, pp. 71–79, 17–19 Jun 2013.
[34] N. Baumgarten and C. Wieners, "The parallel finite element system M++ with integrated multilevel preconditioning and multilevel Monte Carlo methods," Comput. Math. Appl., vol. 81, pp. 391–406, 2021.
[35] C. Wieners, D. Corallo, D. Schneiderhan, L. Stengel, H. D. N. Pham, and N. Baumgarten, "Mpp 3.5.0," 2025.

[36] M. Hinze, R. Pinnau, M. Ulbrich, and S. Ulbrich, Optimization with PDE Constraints. Mathematical Modelling: Theory and Applications, Springer, 2009.
[37] G. J. Lord, C. E. Powell, and T. Shardlow, An Introduction to Computational Stochastic PDEs. Cambridge Texts in Applied Mathematics, Cambridge University Press, 2014.
[38] G. Pflug, Optimization of Stochastic Models. The Kluwer international series in engineering and computer science, Kluwer Acad. Publ., 1996.
[39] A. Barth, C. Schwab, and N. Zollinger, "Multilevel Monte Carlo Finite Element Method for elliptic PDEs with stochastic coefficients," Numerische Mathematik, vol. 119, no. 1, pp. 123–161, 2011.
[40] J. Charrier, R. Scheichl, and A. L. Teckentrup, "Finite element error analysis of elliptic PDEs with random coefficients and its application to multilevel Monte Carlo methods," SIAM J. Numer. Anal., vol. 51, no. 1, pp. 322–352, 2013.
[41] A. L. Teckentrup, R. Scheichl, M. B. Giles, and E. Ullmann, "Further analysis of multilevel Monte Carlo methods for elliptic PDEs with random coefficients," Numer. Math., vol. 125, no. 3, pp. 569–600, 2013.
[42] N. Collier, A.-L. Haji-Ali, F. Nobile, E. von Schwerin, and R. Tempone, "A continuation multilevel Monte Carlo algorithm," BIT, vol. 55, no. 2, pp. 399–432, 2015.
[43] F. Lindgren, H. Rue, and J. Lindström, "An explicit link between gaussian fields and gaussian markov random fields: the stochastic partial differential equation approach," Journal of the Royal Statistical Society Series B: Statistical Methodology, vol. 73, no. 4, pp. 423–498, 2011.
[44] C. Bierig and A. Chernov, "Convergence analysis of multilevel monte carlo variance estimators and application for random obstacle problems," Numerische Mathematik, vol. 130, pp. 579–613, 2015.
[45] D. C. Liu and J. Nocedal, "On the limited memory bfgs method for large scale optimization," Mathematical programming, vol. 45, no. 1, pp. 503–528, 1989.
[46] F. Köhne and A. Schiela, "An exponential averaging process with strong convergence properties," arXiv preprint arXiv:2505.10605, 2025.
[47] D. P. Kouri and T. M. Surowiec, "Risk-averse pde-constrained optimization using the conditional value-at-risk," SIAM Journal on Optimization, vol. 26, no. 1, pp. 365–396, 2016.
[48] Q. Ayoul-Guilmard, S. Ganesh, S. Krumscheid, and F. Nobile, "Quantifying uncertain system outputs via the multi-level monte carlo method- distribution and robustness measures," International Journal for Uncertainty Quantification, vol. 13, no. 5, 2023.
[49] J. Rowbottom, S. Fresca, P. Lio, C.-B. Schönlieb, and N. Boullé, "Multi-level monte carlo training of neural operators," arXiv preprint arXiv:2505.12940, 2025.