Bi-fidelity stochastic gradient descent for structural optimization under uncertainty

Subhayan De · Kurt Maute · Alireza Doostan

Received: 27 November 2019 / Accepted: 9 June 2020 / Published online: 3 August 2020
© Springer-Verlag GmbH Germany, part of Springer Nature 2020

Abstract
The presence of uncertainty in material properties and geometry of a structure is ubiquitous. The design of robust engineering structures, therefore, needs to incorporate uncertainty in the optimization process. Stochastic gradient descent (SGD) method can alleviate the cost of optimization under uncertainty, which includes statistical moments of quantities of interest in the objective and constraints. However, the design may change considerably during the initial iterations of the optimization process which impedes the convergence of the traditional SGD method and its variants. In this paper, we present two SGD based algorithms, where the computational cost is reduced by employing a low-fidelity model in the optimization process. In the first algorithm, most of the stochastic gradient calculations are performed on the low-fidelity model and only a handful of gradients from the high-fidelity model is used per iteration, resulting in an improved convergence. In the second algorithm, we use gradients from the low-fidelity models to be used as control variate, a variance reduction technique, to reduce the variance in the search direction. These two bi-fidelity algorithms are illustrated first with a conceptual example. Then, the convergence of the proposed bi-fidelity algorithms is studied with two numerical examples of shape and topology optimization and compared to popular variants of the SGD method that do not use low-fidelity models. The results show that the proposed use of a bi-fidelity approach for the SGD method can improve the convergence. Two analytical proofs are also provided that show linear convergence of these two algorithms under appropriate assumptions.

Keywords Bi-fidelity method · Optimization under uncertainty · Stochastic gradient descent · Control variate · Stochastic average gradient (SAG) · Stochastic variance reduced gradient (SVRG)

1 Introduction
In simulation-based engineering, models, often in the form of discretized (partial) differential equations, are used for purposes such as analysis, design space exploration, uncertainty quantification, and design optimization. In the context of structural optimization, such as shape and topology optimization, the models need to be simulated many times throughout the optimization process [96]. Structures are often subjected to uncertainties in the material properties, geometry, and external loads [17,44,65]. Hence, for robust design of these structures, such uncertainties must be accounted for in the optimization process. The most commonly used method to compute the stochastic moments of the design criteria for optimization under uncertainty (OuU) is random sampling based Monte Carlo approach. In this approach, at every iteration, statistics calculated from a number of random samples are used as the objective and constraints for the optimization. However, the number of random samples often needs to be large to get a small approximation error. As a result, this approach increases the computational burden even further as one needs to solve the governing equations many times at every iteration of the optimization [23,25,26,87]. Stochastic collocation [4,71] or polynomial chaos expansion [38,99] methods can also be utilized to estimate these statistics, but the required number of random samples increases rapidly with the number of optimization parameters. Note that, sparse polynomial chaos expansions [10,28,29,41,42] can be used to reduce the computational cost. However, for design opti-
mization problems, where uncertainty is represented by a large number of random variables the computational cost may remain unbearable.

In deterministic optimization problems, $\theta \in \mathbb{R}^{n\theta}$ denotes the vector of design parameters and the objective function $f(\theta) : \mathbb{R}^{n\theta} \to \mathbb{R}$, e.g., strain energy of a structure, depends on $\theta$. For constrained optimization problems, let $g(\theta) : \mathbb{R}^{n\theta} \to \mathbb{R}^n$ be $n_g$ real-valued constraint functions, e.g., allowable mass of a structure. The constraints are satisfied if $g_i(\theta) \leq 0$ for $i = 1, \ldots, n_g$. Hence, the optimization problem can be written as

$$
\min_{\theta} \ f(\theta) \text{ subject to } g_i(\theta) \leq 0 \text{ for } i = 1, \ldots, n_g. \tag{1}
$$

In the presence of uncertainty, a reformulation of the optimization problem in (1) is generally used. Let $\xi \in \mathbb{R}^{n_\xi}$ be the vector of random variables, with known probability distribution function, characterizing the system uncertainty. The objective function $f(\theta; \xi) : \mathbb{R}^{n\theta} \times \mathbb{R}^{n_\xi} \to \mathbb{R}$ now also depends on the realized values of $\xi$. Similarly, for constrained optimization problems, $n_g$ real-valued constraint functions $g(\theta; \xi) : \mathbb{R}^{n\theta} \times \mathbb{R}^{n_\xi} \to \mathbb{R}^{n_\xi}$ in general depend on $\xi$. The optimization problem is defined using the expected risk $R(\theta) = \mathbb{E}[f(\theta; \xi)]$ and expected constraint value $C_i(\theta) = \mathbb{E}[g_i(\theta; \xi)]$ as follows [18,23,88,101]

$$
\min_{\theta} \ R(\theta) \text{ subject to } C_i(\theta) \leq 0 \text{ for } i = 1, \ldots, n_g. \tag{2}
$$

where $\mathbb{E}[\cdot]$ denotes the mathematical expectation of its argument.

The stochastic gradient descent methods illustrated in this paper utilize evaluations of the gradients of $R(\theta)$ and $C(\theta)$. In optimization, a combination of these gradients defines a direction along which a search is performed. Each of the investigated methods modifies these directions with the goal to improve stability in the optimization so that a converged solution is reached more reliably (i.e., the same converged solution is reached for multiple runs of the optimization algorithms) in a smaller number of optimization steps.

### 1.1 Multi-fidelity models with applications to OuU

In most engineering problems (or applications), multiple models are often available to describe the system. Some of these models are able to describe the behavior with a higher level of accuracy but are generally associated with high computational cost; in the sequel referred to as high-fidelity models. Models with lower computational cost, on the other hand, are often (not always) less accurate and termed low-fidelity models. For a structural system analyzed by the finite element method, low-fidelity models can be obtained, for instance, using coarser grid discretizations of the governing equations. Multi-fidelity methods exploit the availability of these different models to accelerate design optimization, parametric studies, and uncertainty quantification; see, e.g., Fernández-Godino et al. [35], Peherstorfer et al. [78], and the references therein. In these methods, most of the computation is performed using the low-fidelity models and the high-fidelity models are utilized to correct the low-fidelity predictions.

The use of multi-fidelity models is especially helpful in optimization, where one needs to solve the governing equations multiple times [74]. Booker et al. [11] used pattern search to construct low-fidelity surrogate models. Forrester et al. [37] used co-kriging [53,66,80] for constructing surrogates that are then used for design optimization of an aircraft wing. Similar applications of multi-fidelity models for optimization of aerodynamic design are performed in Keane [51], Huang et al. [47], Choi et al. [20], and Robinson et al. [82]. Reduced order models constructed using Padé approximation, Shanks transformation, Krylov subspace methods, derivatives of eigenmodes with respect to design variables, and proper orthogonal decomposition are used in Chen et al. [19], Kirsch [55], Hurtado [48], Sandridge and Haftka [89], and Weickum et al. [98] to perform optimization of structural systems. Eldred and Dunlavy [31] compared different data-fit, multi-fidelity, and reduced order models to construct surrogates for optimization. Yamazaki et al. [100] used a gradient enhanced kriging to perform design optimization as well as uncertainty quantification at the optimal design point. Another optimization method known as space mapping [5–7,57] uses the low-fidelity models to solve an approximate optimization problem, where the input parameter space is mapped onto a different space to construct multi-fidelity models. Fischer et al. [36] used a Bayesian approach for estimating weights of the low-fidelity models to be used for optimization in a multi-fidelity optimization setting.

Recently, the multi-fidelity approach has also been applied for efficient uncertainty quantification. There, the low-fidelity models are used to reduce the computational cost associated with the solution of governing equations of large-scale physical systems in the presence of high-dimensional uncertainty [27,34,43,56,67,70,75,77–79,95].

For OuU, Jin et al. [49] used different metamodeling approaches, e.g., polynomial regression, kriging [64] to ease the computational burden of the optimization iterations. Kroo et al. [58] employed multi-fidelity models for multiobjective optimization, where these objectives provide a balance between performance and risk associated with the uncertainty in the problem. Keane [52] used co-kriging for robust design optimization with an application to shape optimization of a gas-turbine blade. Allaire et al. [1] used a Bayesian approach for risk based multidisciplinary optimization, where models of multiple fidelity are used to merge
information on uncertainty. In multidisciplinary optimization, Christensen [21] increased the fidelity of the model for a particular discipline from which the contribution to the uncertainty of the quantity of interest is large. Eldred and Elman [32] and Padron et al. [73] used stochastic expansion methods, e.g., stochastic collocation and polynomial chaos expansion methods to construct high- and low-fidelity surrogate models for design OuU. March and Wilcox [61–63] proposed a multi-fidelity based trust-region algorithm for optimization using gradients from low-fidelity models only. Ng and Willcox [69] used control variate approach for the design of an aircraft wing under uncertainty.

1.2 Stochastic gradient descent methods

Gradient descent methods [15,72] for solving (1) are the preferred choice if \( f(\theta) \) and \( g(\theta) \) are differentiable with respect to \( \theta \). To solve the optimization problem under uncertainty in (2), a stochastic version of gradient descent [81] that has a smaller per iteration computational cost compared to a Monte Carlo approach with large number of random samples can be used. Recently, the stochastic gradient descent (SGD) method has seen increasing use in training of neural networks [12], where the number of optimization variables is large. In this method, the gradients used at every iteration are obtained using either only one or a small number of random samples of \( \xi \). However, the standard SGD method converges slowly [68]. Hence, to improve the convergence various modifications to the standard SGD method have been proposed recently. Among these, adaptive gradient (AdaGrad) [30] and adaptive moment (Adam) [54] retard the movement in directions with historically large gradient magnitudes and are useful for problems that lack convexity. Adadelta [102] removes the need for an explicitly specified learning rate, i.e., step size; however, small initial gradients affect this algorithm adversely [22]. Another variant of SGD method, the stochastic average gradient (SAG) algorithm [84] updates a single gradient using one random sample of the uncertain parameters per iteration and keeps the rest same as in the last iteration. Then, the optimization parameters are updated using an average of all computed gradients. However, for design optimization, this approach of using past gradients may lead to poor convergence [22]. The stochastic variance reduced gradient (SVRG) algorithm [50] uses a control variate [83] to reduce the variance of the stochastic gradients. This approach, as introduced in Sect. 2.1, can also suffer from poor convergence in OuU if the same control variate is used for a large number of iterations [22,24].

In this paper, to achieve a better convergence and to ease the computational burden of OuU, we formulate two bi-fidelity based variants of the SAG and SVRG algorithms to overcome their respective shortcomings. We use the word bi-fidelity instead of multi-fidelity as we are only using one high- and one low-fidelity model. The first algorithm, bi-fidelity stochastic average gradient (BF-SAG), evaluates most of the gradients using a low-fidelity model and then applies a gradient descent step using an average gradient. In the second algorithm, bi-fidelity stochastic variance reduced gradient (BF-SVRG), we propose a control variate approach, where the mean of the control variate is estimated using only low-fidelity gradient evaluations resulting in the reduction of the computational cost. In addition, we use the correlation between the high- and low-fidelity gradients to reduce the variance in the estimated gradients. Further, we prove linear convergence of these two proposed algorithms for strongly convex objectives and gradients that are Lipschitz continuous.

We illustrate the proposed bi-fidelity algorithms using three numerical examples. For the first example, we choose a simple fourth order polynomial as the high-fidelity model and a second-order approximation of it as the low-fidelity model. This example shows that the use of a low-fidelity model allows for evaluating more gradients per iteration using the same computational budget and thereby leads to a faster convergence. In the second example, we use an example of a square plate with a hole subjected to uniaxial tension. We optimize the shape of the hole to minimize the maximum principal stress in the plate. For the third example, we apply the proposed algorithms to a topology optimization problem using the solid isotropic material with penalisation (SIMP) method [8,9,94]. In all examples, we observe convergence improvements of the proposed bi-fidelity strategies over their standard counterparts, e.g., SAG and SVRG.

The rest of the paper is organized as follows. In the next section, we briefly discuss some popular variants of the SGD method. In Sect. 3, we introduce the proposed bi-fidelity based optimization algorithms with convergence properties and computational cost analyses. In Sect. 4, we illustrate various aspects of the algorithms with three numerical examples. Finally, we conclude our paper with a brief discussion of future research directions of bi-fidelity based SGD methods.

2 Background

In this section, we discuss the stochastic gradient descent method and its variants, namely, stochastic average gradient (SAG) and stochastic variance reduced gradient (SVRG). A related concept of variance reduction using a control variate is also briefly discussed.

2.1 Stochastic gradient descent (SGD) method and its variants

With the SGD method, for unconstrained problems the expected risk is minimized as mentioned in Sect. 1. For
constrained optimization problems, an unconstrained formulation of (2) can be used employing a penalty formulation and constraint violation defined as \( g_j^+(\theta; \xi_k) \leq 0 \) and \( g_j^+(\theta; \xi) \) for \( g_j(\theta; \xi) > 0, j = 1, \ldots, n_g \). The optimization problem is then formulated as follows

\[
\min_{\theta} J(\theta) = R(\theta) + \sum_{j=1}^{n_g} \kappa_j C_j(\theta),
\]

where the objective \( J(\theta) \) is a combination of the expected risk \( R(\theta) \) and the expected squared constraint violations \( C_j(\theta) := \mathbb{E}[(g_j^+(\theta; \xi))^2] \) for \( j = 1, \ldots, n_g \). In (3), \( \kappa \) is a user-specified penalty parameter vector. For large values of \( \kappa \), (3) has similar solutions to (2) [60].

The basic SGD method uses a single realization of \( \xi \) from its set of \( N \) realizations to perform the update on \( \theta \) at the \( k \)th iteration utilizing the gradient \( \nabla_{\theta} f(\theta_k; \xi_k) \) [13]. For a constrained optimization problem, the search direction \( h_k \) is estimated as the combination of the gradients of the cost function \( f(\theta_k; \xi) \) and the square of constraint violations \( (g_j^+(\theta_k; \xi))^2 \) as

\[
h_k := h(\theta_k; \xi_k) = \nabla_{\theta} f(\theta_k; \xi_k) + \sum_{j=1}^{n_g} \kappa_j \nabla_{\theta} (g_j^+(\theta_k; \xi_k))^2,
\]

where \( \xi_k \) is selected uniformly at random from its set of \( N \) realizations. The parameter update is then applied as follows

\[
\theta_{k+1} = \theta_k - \eta h_k,
\]

where \( \eta \) is the step size, also known as the learning rate. Note that, for a large step size the standard SGD algorithm and also its variants discussed next may diverge whereas very small step size will result in a slow convergence. Other than a constant learning rate used in this paper, a scheduling of the learning rate [91] to gradually lower the step size as iteration count increases can also be applied to reduce the oscillations at the end of the optimization occurring due to the stochastic nature of the search direction. However, a detailed study of different learning rate schedule strategies is beyond the scope of this paper. Also, a rescaling of the parameters may help in convergence if these parameters vary by many orders of magnitude. The steps of the standard SGD method are illustrated in Algorithm 1. Following (5) and (6), the SGD method performs only one gradient calculation per iteration; hence, its computational cost per iteration is relatively small. However, the descent direction \( V J \) is not followed at every iteration. Instead, the descent is achieved in expectation as the expectation of the stochastic gradient is the same as the gradient of the objective \( J(\theta) \). As a result, the convergence of the SGD method can be very slow [12]. A straightforward extension of the SGD method is to use a small batch of random samples to compute the search direction \( h_k \). This version is known as mini-batch gradient descent [13,86]. In the past few years, several modifications of the SGD method have been proposed to improve its convergence [13]. Two of them, namely, the stochastic average gradient (SAG) and the stochastic variance reduced gradient (SVRG) algorithms that are relevant to this paper are discussed next.

### Algorithm 1: Stochastic gradient descent [13]

Given \( \eta \).

Initialize \( \theta_1 \).

for \( k = 1, 2, \ldots \) do

Compute \( h_k := h(\theta_k; \xi_k) \). [see Eq. (5)]

Set \( \theta_{k+1} \leftarrow \theta_k - \eta h_k \).

end for

### 2.2 Stochastic average gradient (SAG) algorithm

One popular variant of the SGD method is the SAG algorithm [84], which updates the gradient information for one random sample at every iteration and keeps the old gradients for other samples. The parameters are then updated using (6) with the search direction \( h_k \) defined as

\[
h_k = \frac{1}{N} \sum_{i=1}^{N} d_{k,i};
\]

\[
d_{k,i} = \begin{cases} h(\theta_k; \xi_i) & \text{if } i = t \in \{1, 2, \ldots, N\}; \\ d_{k-1,i} & \text{otherwise}, \end{cases}
\]

where \( t \) is selected uniformly at random from \( \{1, 2, \ldots, N\} \) and at the start of the algorithm \( d_{0,i} = \theta \) for \( i = 1, \ldots, N \). This use of previous gradient information accelerates the convergence of the optimization compared to the standard SGD method defined in Algorithm 1. However, relying on past gradients can impede convergence rate and/or stability for OoU as the design often changes drastically during the early iterations of the optimization process [22]. In this paper, we use a batch version of this algorithm, where \( N_b > 1 \) gradients are updated at every iteration (see Algorithm 2).
2.3 Variance reduction using control variates

In this subsection, we briefly discuss control variates, a variance reduction technique that we use in one of our proposed algorithms. It has also been used in the SVRG algorithm, a variant of the SGD method, described in Sect. 2.4. A control variate can be used to estimate the expected value of a random variable \( X \) via Monte Carlo averaging, while reducing the variance of the estimate [40, 83]. Here, another random variable \( Y \), known as the control variate, is introduced such that it is correlated with \( X \), is cheaper to simulate than \( X \), and either the expected value of \( Y \), \( \mathbb{E}[Y] \), is known or can be estimated accurately and relatively cheaply. Using \( Y \), \( \mathbb{E}[Y] \), and the standard Monte Carlo simulation, the expected value of

\[
Z = X - \alpha (Y - \mathbb{E}[Y]),
\]

is estimated as an unbiased estimator of \( \mathbb{E}[X] \). In (8), \( \alpha \) is a control variate parameter and when set to

\[
\alpha^* = \frac{\sigma_{XY}}{\sigma_Y^2},
\]

the sample average estimate of \( \mathbb{E}[Z] \) achieves the minimum mean squared error. In (9), \( \sigma_{XY} \) is the covariance between \( X \) and \( Y \) and \( \sigma_Y^2 \) is the variance of \( Y \). When \( \alpha = \alpha^* \) is used in (8) to estimate \( \mathbb{E}[Z] \) with \( N \) Monte Carlo samples of \( X \) and \( Y \), the variance of sample average of \( Z \) is given by

\[
\frac{1}{N}(1 - \rho_{XY}^2)\sigma_X^2,
\]

where \( \rho_{XY} = \frac{\sigma_{XY}}{\sigma_X \sigma_Y} \) is the correlation between \( X \) and \( Y \), and \( \sigma_X^2 \) is the variance of \( X \). Since, \(-1 \leq \rho_{XY} \leq 1\), the reduction in variance of the estimate based on \( Z \) can be seen when compared to the variance of the standard Monte Carlo estimate of \( \mathbb{E}[X] \) given by

\[
\frac{1}{N} \sigma_X^2.
\]

In an ideal situation, \( \rho_{XY} \to 1 \) and as a result \( \frac{1}{N}(1 - \rho_{XY}^2)\sigma_X^2 \to 0 \). A bi-fidelity algorithm proposed in Sect. 3.2 employs this variance reduction technique, where \( X \) and \( Y \) consist of high- and low-fidelity model gradients, respectively.

For random vectors \( X \) and \( Y \), (8) is replaced by

\[
Z = X - \alpha (Y - \mathbb{E}[Y]),
\]

where \( \alpha \) is a coefficient matrix. The optimal value of the coefficient matrix \( \alpha \) that minimizes the trace of the covariance matrix of \( Z \) is given by

\[
\alpha^* = \nabla_Y^{-1} C_{XY},
\]

where \( \nabla_Y \) is the covariance matrix of the control variate \( Y \) and \( C_{XY} \) is the cross-covariance between \( X \) and \( Y \).

2.4 Stochastic variance reduced gradient (SVRG) algorithm

The second variant of the SGD method that we present here is the SVRG algorithm [50]. In this algorithm, a variance reduction method is introduced by maintaining a parameter estimate \( \theta_{prev} \) at every inner iteration that is updated only during the outer iteration. Using this parameter estimate \( \theta_{prev} \) and \( N_h \) samples of \( \xi \), the mean of \( h(\theta_{prev}; \xi) \) is estimated as

\[
\tilde{h}(\theta_{prev}) = \frac{1}{N_h} \sum_{i=1}^{N_h} h(\theta_{prev}; \xi_i).
\]

Note that, in (12) \( N_h \) can be smaller than \( N \) in (7). Next, the update rule in (6) is applied with the search direction \( h_k \) defined as

\[
h_k = h(\theta_k; \xi_i) - h(\theta_{prev}; \xi_i) + \tilde{h}(\theta_{prev})
\]

for a chosen \( i \) uniformly at random, i.e., \( h(\theta_{prev}; \xi) \) is used here as a control variate with \( \alpha = \mathbf{I} \) in (10), where \( \mathbf{I} \) is the identity matrix. These steps are illustrated in Algorithm 3.

---

**Algorithm 2: Batch Implementation of SAG**

Given \( \eta \) and \( N_h \),

1. Initialize \( \theta_1 \).
2. Initialize \( d_{0,i} = 0 \) for \( i = 1, \ldots, N \).
3. for \( k = 1, 2, \ldots \) do
   1. Draw \( [t_i]_{i=1}^{N_h} \) uniformly at random from \( [1, \ldots, N] \).
   2. for \( i = 1, 2, \ldots, N \) do
      1. if \( i \in [t_i]_{i=1}^{N_h} \) then
         1. Compute \( d_{k,i} := h(\theta_k; \xi_i) \)
         2. else
            1. Compute \( d_{k,i} := d_{k-1,i} \).
      2. end if
   3. end for
4. \( \theta_{k+1} \leftarrow \theta_k - \frac{\eta}{N_h} \sum_{i=1}^{N_h} d_{k,i} \). [see Eq. (7)]
5. end for

---

**Algorithm 3: SVRG [50]**

Given \( \eta \), \( m \), and \( N_h \),

1. Initialize \( \theta_1 \).
2. for \( j = 1, 2, \ldots \) do (outer iteration)
   1. Set \( \theta_{prev} = \theta_j \).
   2. Set \( \tilde{h}(\theta_{prev}) = \frac{1}{N_h} \sum_{i=1}^{N_h} h(\theta_{prev}; \xi_i) \).
   3. Set \( \theta_{prev} \).
   4. for \( k = 1, 2, \ldots, m \) do (inner iteration)
      1. Uniformly at random choose \( t \in [1, 2, \ldots, N_h] \).
      2. Set \( \theta_{k+1} \rightarrow \theta_k - \eta \left[ h(\theta_k; \xi_t) - h(\theta_{prev}; \xi_t) + \tilde{h}(\theta_{prev}) \right] \). [see Eq. (13)]
      3. end for
6. \( \theta_j \leftarrow \theta_{m+1} \).
7. end for
3 Methodology: proposed algorithms

In this section, inspired by the SAG and SVRG algorithms, we propose two variants of the SGD method using a combination of high- and low-fidelity gradient evaluations.

3.1 Bi-fidelity stochastic average gradient (BF-SAG) algorithm

Similar to batch implementation of the SAG algorithm, at every iteration of the proposed BF-SAG algorithm, we update $N_h$ gradients using the high-fidelity model. In addition, we update $N_l \gg N_h$ gradients using the low-fidelity model. Unlike in the SAG algorithm, by using many low-fidelity model evaluations to update most of the gradients, in addition to the high-fidelity gradients, we reduce the dependency on previous designs as in the optimization process designs may go through drastic changes over a few iterations. Similar to the SAG algorithm in (7), parameter update in (6) is performed at the $k$th iteration using $N$ stochastic gradients $d_{k,i}$. Specifically, the search direction is defined as

$$h_k = \frac{1}{N} \sum_{i=1}^{N} d_{k,i};$$

$$d_{k,i} = \begin{cases} h_{\text{low}}(\theta_k; \xi_i) & \text{if } i \in \{t_l\}_{l=1}^{N_l}; \\ h_{\text{high}}(\theta_k; \xi_i) & \text{if } i \in \{t_h\}_{h=1}^{N_h}; \\ d_{k-1,i} & \text{otherwise}, \end{cases}$$

where $t_l$ and $t_h$ are selected uniformly at random from \{1, 2, \ldots, N\} with $\{t_l\} \cap \{t_h\} = \emptyset$. A more involved strategy where $N_h$ and $N_l$ depend on the iteration number can also be implemented for faster convergence. However, a detailed investigation of this strategy is beyond the scope of the current paper. The implementation of these steps is summarized in Algorithm 4.

3.1.1 Computational cost

Let the ratio of the computational effort for a low-fidelity model compared to a high-fidelity model be $\gamma < 1$. Note that, $\gamma$ may include the cost of generating a high-fidelity gradient estimate from a low-fidelity gradient, e.g., interpolating a gradient computed from a coarse grid model on a fine grid. The computational cost in terms of the cost of high-fidelity gradient evaluations for $N_l$ number of iterations of the BF-SAG algorithm can be given by

$$C_{\text{BF-SAG}} := N_l \left[ N_h + \gamma N_l \right].$$

Next, we compare the per-iteration cost of the BF-SAG algorithm relative to that of a batch implementation of the SAG algorithm, where we update $N_h'$ gradients per iteration. The ratio of their respective per-iteration cost is $\frac{N_h + \gamma N_l}{N_h'}$. In general, $N_h' > N_h$ and $\gamma$ is a very small number leading to per-iteration cost efficiency. Note that, the per-iteration costs used to evaluate the ratio do not necessarily lead to similar accuracy in a fixed number of iterations.

3.1.2 Convergence

In this subsection, we present a result for linear convergence of the proposed BF-SAG algorithm. The corresponding assumptions and result that are presented in the following theorem are inspired from the results of Schmidt et al. [90].

**Theorem 1** Assume the objective function $J(\theta)$ obtained from the low- and high-fidelity models are strongly convex with constants $\mu_{\text{low}}$ and $\mu_{\text{high}}$, respectively. Also, assume that the corresponding gradients are Lipschitz continuous with constants $L_{\text{low}}$ and $L_{\text{high}}$, respectively. Let $\theta^* = \text{argmin}_{\theta} J(\theta)$ and initialize the gradient history vector $d$ to zero. For some constants $\mu$, $L^2 > 0$ that depend on the constants $\mu_{\text{low}}$, $\mu_{\text{high}}$, and $L_{\text{low}}^2$ and $L_{\text{high}}^2$, respectively, Algorithm 4 achieves a linear convergence as

$$\mathbb{E} [\|\theta_{k+1} - \theta^*\|^2 | \theta_k] \leq (1 - \mu^2 / L^2)^k \|\theta_0 - \theta^*\|^2,$$

where $\theta_{k+1}$ is the parameter vector at $(k+1)$th iteration.

The proof of this theorem along with the details of (16) and the definitions of $\mu_{\text{low}}$, $\mu_{\text{high}}$, $L_{\text{low}}$, and $L_{\text{high}}$ are given in Appendix A. Note that, the structural optimization settings that we consider here do not satisfy the conditions of this theorem, e.g., strong convexity. However, the empirical results presented in Sect. 4 illustrate convergence for the non-convex problems considered in this paper.

---

**Algorithm 4:** Bi-fidelity stochastic average gradient (BF-SAG)

Given $\theta_0$, $\eta$, $N_l$, $N_h$, and $N_l$.

Initialize $\theta_1 = \theta_0$ and $d = 0$.

for $k = 1, 2, \ldots, N_l$ do

Draw $N_l + N_h$ samples $\{t_l\}_{l=1}^{N_l}$ uniformly from \{1, \ldots, N\}.

Define $\{t_h\}_{h=1}^{N_h} \equiv \{t_l\}_{l=1}^{N_l}$ and $\{t_h\}_{h=N_l+1}^{N_l+N_h}$.

for $i = 1, 2, \ldots, N$ do

if $i \in \{t_l\}_{l=1}^{N_l}$ then

Compute $d_{k,i} := h_{\text{low}}(\theta_k; \xi_{t_l})$.

else if $i \in \{t_h\}_{h=1}^{N_h}$ then

Compute $d_{k,i} := h_{\text{high}}(\theta_k; \xi_{t_h})$.

else

Set $d_{k,i} = d_{k-1,i}$.

end if

end for

$\theta_{k+1} \leftarrow \theta_k - \eta \sum_{i=1}^{N} d_{k,i}$.

end for

---

(Springer)
3.2 Bi-fidelity stochastic variance reduced gradient (BF-SVRG)

In our second algorithm, we exploit the availability of low- and high-fidelity model gradients by using a control variate similar to its use in the SVRG algorithm [50], resulting in a bi-fidelity extension of SVRG named here BF-SVRG. In particular, we employ the gradient $\mathbf{h}_{\text{low}}$ evaluated using the low-fidelity model and a previous estimate of the parameters $\theta_{\text{prev}}$ as a control variate ($\mathbf{Y}$ in (10)). To estimate the mean of the control variate $\hat{\mathbf{h}}_{\text{low}}$, we use $N_l$ random samples as follows:

$$\hat{\mathbf{h}}_{\text{low}} = \frac{1}{N_l} \sum_{i=1}^{N_l} \mathbf{h}_{\text{low}}(\theta_{\text{prev}}; \xi_i).$$  \hspace{1cm} (17)$$

Note that, the number of random samples $N_l$ required to estimate $\hat{\mathbf{h}}_{\text{low}}$ can be large [85] but the low-fidelity model is used, keeping the cost small. Also, as in the standard SVRG (Algorithm 3), $\theta_{\text{prev}}$ is updated only at every $m$ inner iteration and the same $\hat{\mathbf{h}}_{\text{low}}$ is kept for all these inner iterations. At every inner iteration, $N_h \ll N_l$ random variables are used to estimate gradients using the high-fidelity model. Note that, these $N_h$ random variables are not necessarily a subset of the $N_l$ random samples used in (17). The estimated gradients are used to evaluate the mean as follows:

$$\hat{\mathbf{h}}_{\text{high}} = \frac{1}{N_h} \sum_{b=1}^{N_h} \mathbf{h}_{\text{high}}(\theta_b; \xi_b).$$  \hspace{1cm} (18)$$

The gradient descent step in (6) is performed with the search direction defined as

$$\mathbf{h}_k = \hat{\mathbf{h}}_{\text{high}} - \alpha \sum_{b=1}^{N_h} \left( \mathbf{h}_{\text{low}}(\theta_{\text{prev}}; \xi_b) - \hat{\mathbf{h}}_{\text{low}} \right).$$  \hspace{1cm} (19)$$

where $\alpha$ is a coefficient matrix and the same set of random samples $\{\xi_b\}_{b=1}^{N_h}$ from (18) is used. The estimation of optimal coefficient matrix using (11) requires computation of the inverse of the covariance matrix of $\mathbf{h}_{\text{low}}(\theta_{\text{prev}}; \xi)$. To avoid this computationally expensive step, herein we assume $\alpha$ is a diagonal matrix with optimal diagonal entries given by [97]

$$\alpha^*_i = \frac{C_{ii}}{\mathbb{V}_{ii}}, \quad i = 1, \ldots, n_\theta,$$  \hspace{1cm} (20)$$

where $\mathbb{V}$ is the covariance matrix of $\mathbf{h}_{\text{low}}(\theta_{\text{prev}}; \xi)$ and $C$ is the cross-covariance between $\mathbf{h}_{\text{high}}(\theta_b; \xi)$ and $\mathbf{h}_{\text{low}}(\theta_{\text{prev}}; \xi)$. We use the $N_h$ random samples used in (18) and (19) to compute the diagonal entries of $\alpha$ as follows:

$$\alpha^*_{ii} = \sum_{b=1}^{N_h} \left( \mathbf{h}_{\text{high}}(\theta_b; \xi_b) - \hat{\mathbf{h}}_{\text{high}} \right) \odot \left( \mathbf{h}_{\text{low}}(\theta_{\text{prev}}; \xi_b) - \hat{\mathbf{h}}_{\text{low}} \right)_{ii}, \quad i = 1, \ldots, n_\theta,$$  \hspace{1cm} (21)$$

where $\odot$ denotes a Hadamard product (i.e., elementwise multiplication). Here, the total number of low-fidelity gradient evaluation is $(mN_h + N_l)$. Hence, when we increase $N_h$, i.e., the number of high-fidelity gradient evaluation, the total number of low-fidelity gradient evaluation also increases. Note that, the SVRG algorithm uses the high-fidelity gradient from the past iteration as the control variate and an identity matrix as the $\alpha$. On the other hand, the proposed BF-SVRG algorithm uses the low-fidelity gradient as the control variate and uses a diagonal coefficient matrix $\alpha$, which is not necessarily an identity matrix. Also, there are other efficient strategies for random sampling to estimate the control variate as discussed in Gorodetsky et al. [39]. However, a detailed study on the implementation of these strategies inside the proposed BF-SVRG algorithm is beyond the scope of this paper. Note that, similar to the BF-SAG algorithm, we can also develop strategies to vary $N_h$ and $N_l$ over iterations. A description of these steps is shown in Algorithm 5.

3.2.1 Computational cost

From Algorithm 5, the computational cost of the BF-SVRG algorithm can be given by

$$C_{\text{BF-SVRG}} = N_{\text{olt}} \left[ \gamma N_l + \gamma m N_h + m N_h \right]$$

$$= N_{\text{olt}} \left[ \gamma N_l + (\gamma + 1) m N_h \right],$$  \hspace{1cm} (22)$$

where $N_{\text{olt}}$ is the number of outer iterations and $\gamma$ is the ratio of the computational cost of the low- over high-fidelity gradient evaluations, as in Sect. 3.1. Next, we compare the per outer iteration cost of the BF-SVRG algorithm with the SVRG algorithm, where both use $m$ inner iterations, but the SVRG algorithm uses $N'_h$ gradient evaluations to estimate the mean of the control variate $\hat{\mathbf{h}}(\theta_{\text{prev}})$ in (12) and a batch of $N'_h$ random samples to estimate $\mathbf{h}(\theta_b; \xi)$ and $\mathbf{h}(\theta_{\text{prev}}; \xi)$ in (13). Using (22), the ratio of per outer iteration cost of the BF-SVRG and SVRG algorithms is given by $\frac{N_{\text{olt}}}{N_{\text{olt}}}$, where $N'_h \approx N_l \gg N_h$ and $N''_h \approx N_h$, thus the BF-SVRG algorithm leads to cost efficiency. Note that, the per-iteration costs used to evaluate the ratio do not necessarily lead to similar accuracy in a fixed number of iterations.

3.2.2 Convergence

In this subsection, we present a theorem to show that the convergence rate of the proposed BF-SVRG algorithm is linear and depends on the correlation between the high- and low-fidelity gradients. The corresponding assumptions and result
that are presented in the following theorem are inspired from the results of Schmidt et al. [90], Johnson and Zhang [50].

**Theorem 2** Assume the objective function $J(\theta)$ obtained from high-fidelity models is strongly convex with a constant $\mu_{\text{high}}$ and the corresponding gradients are Lipschitz continuous with constant $L_{\text{high}}$. Let $\theta^* = \arg\min_{\theta} J(\theta)$. For some constant $\delta \geq 1$, Algorithm 5 with $m$ inner iterations and $j$ outer iterations achieves a convergence rate as

$$
\mathbb{E}[\|\tilde{\theta}_j - \theta^*\|^2 | \theta_0] \leq \left(1 - \frac{\mu_{\text{high}}^2}{2L_{\text{high}}^2 \delta}\right)^{jm} \|\theta_0 - \theta^*\|^2, \quad (23)
$$

where $\tilde{\theta}_j$ is the value of the parameter after $j$ outer iterations.

The proof of this theorem is given in Appendix B. We note that the constant $\delta$ depends on the correlation between high- and low-fidelity gradients at a given $\theta$ and has a minimum value of 1 as defined in Appendix B. A higher correlation between the high- and low-fidelity gradients leads to a $\delta$ closer to 1, hence a tighter bound on $\mathbb{E}[\|\tilde{\theta}_j - \theta^*\|^2 | \theta_0]$. Similar to Theorem 1, the structural optimization settings that we consider here do not satisfy the conditions of this theorem, e.g., strong convexity. However, the empirical results presented in Sect. 4 illustrate the convergence for non-convex problems considered in this paper.

**Remark 1** As observed from the results of Theorem 2 and Appendix B, the convergence of BF-SVRG depends on the correlation between the low- and high-fidelity gradients, which is not known in advance. In practice, the degree of correlation may be computed using pilot runs, where a relatively small number of low- and high-fidelity simulations are performed to compute the gradients and their correlation.

### 4 Numerical examples

Three numerical examples are used to illustrate the proposed algorithms in this section. Example I uses a polynomial regression problem to show the working principles and demonstrate the convergence rates provided in Theorems 1 and 2. Examples II and III are chosen from shape and topology optimization. In particular, Example III(a) is a stochastic counterpart of the most commonly used example in topology optimization [92]. We then add another load with uncertain direction to this problem in Example III(b) to show a case where the SVRG algorithm fails. Then, in Example III(c), we increase the dimension of the random vector $\xi$ to 101 to demonstrate that our proposed algorithms perform well even for high stochastic dimensions.

#### 4.1 Example I: polynomial regression

To show the working principle of the proposed algorithms, we first consider the following fourth order polynomial as the high-fidelity model

$$
y_{\text{high}}(x) = 0.5 + x + x^2 + 1.25x^3 + 0.5x^4, \quad (24)
$$

where $x \in [-1, 1]$. The low-fidelity model near $x_0 \in [-1, 1]$ is given by a second order Taylor series expansion, i.e.,

$$
y_{\text{low}}(x; x_0) = y_{\text{high}}(x_0) + \frac{d y_{\text{high}}}{dx} \bigg|_{x_0} (x - x_0) + \frac{1}{2} \frac{d^2 y_{\text{high}}}{dx^2} \bigg|_{x_0} (x - x_0)^2. \quad (25)
$$

We seek to approximate $y_{\text{high}}(x)$ in a fourth order monomial basis as

$$
y_{\text{pred}}(x, \theta) = \sum_{i=0}^{4} \theta_i x^i, \quad \theta_i \in \mathbb{R}, \quad (26)
$$

via the standard least squares regression over the unknown coefficients $\theta = [\theta_0, \theta_1, \theta_2, \theta_3, \theta_4]^T$. To this end, we generate $N = 1000$ noisy measurements of $y_{\text{high}}$ using

$$
y_{\text{obs}}(x_i) = y_{\text{high}}(x_i) + e_i, \quad i = 1, 2, \ldots, N, \quad (27)
$$

where $x_i$ is selected at equal interval from $[-1, 1]$ and $e_i$ are independent, zero-mean normal random variables with variance 0.25. We then try to find an optimal $\theta$ in (26) by solving the unconstrained optimization problem

$$
\min_{\theta} J(\theta) = \frac{1}{N} \sum_{i=1}^{N} \left( y_{\text{obs}}(x_i) - y_{\text{pred}}(x_i, \theta) \right)^2. \quad (28)
$$

Springer
4.1.1 Results

We implement the proposed algorithms for this problem, where the low-fidelity model is defined using a second-order Taylor series expansion about \( x_0 = 0.01 \) (see (25) and Fig. 1). The optimization is performed with an initial guess \( \theta_0 = [2, 2.5, 2.5, 4.5, 1.5]^T \). Figure 2 shows a comparison of mean squared error (MSE) values for the SAG and BF-SAG algorithms with \( \eta = 0.3 \). Note that, when the total number of gradient evaluations per iteration is the same, i.e., \( N_l + N_h = 50 \), the performance of the BF-SAG algorithm is similar or a little worse. However, most of the gradient evaluations are performed with the low-fidelity model, thus making the BF-SAG algorithm cheaper. On the other hand, if we increase the number of low-fidelity gradient evaluations to 225, the BF-SAG algorithm reduces the variations in the MSE. Note that, the number of high-fidelity gradient evaluation in the BF-SAG algorithm at every iteration is still smaller than that of the SAG algorithm. This advantage will be exploited in the rest of the numerical examples in this paper.

Similarly, we compare the SVRG algorithm with the BF-SVRG algorithm in Fig. 3 with \( \eta = 0.3 \) and \( m = 20 \). When using the same number of gradient evaluations in each outer iteration (320 + 20 × (1 + 1) = 360 for SVRG and 200 + 20 × (4 + 4) = 360 for BF-SVRG in Fig. 3), the BF-SVRG algorithm removes the large oscillations in the SVRG algorithm. However, the BF-SVRG algorithm uses significantly smaller number of high-fidelity gradients. Note that, the large oscillations in SVRG is due to the reliance on mean of control variate estimated from past high-fidelity gradients. Further, the SVRG algorithm assumes these past gradients are fully-correlated to the current ones and uses \( \alpha = I \). The convergence of the BF-SVRG algorithm is further improved by using more low- and high-fidelity gradient evaluations to estimate the diagonal entries of the coefficient matrix \( \alpha \) as shown in Fig. 3.

When compared to the BF-SAG algorithm, the BF-SVRG algorithm with \( N_l = 200 \) and \( N_h = 16 \) has similar convergence to the BF-SAG algorithm with \( N_l = 225 \) and \( N_h = 25 \) and both show small oscillations in the objective after some initial iterations as shown in Fig. 4. The relative expected error in the estimates of the optimization parameters estimated using 100 independent runs of the optimization...
algorithms is shown in Fig. 5, which shows that the proposed algorithms have a linear convergence after a few initial iterations. Note that, a linear convergence of the algorithms is not shown for the next two examples as the true value of the parameters \( \theta^* \) are unknown in those two examples. Figure 6 further shows that for three arbitrarily chosen initial guesses the effect of initial guess on the optimization is negligible.

4.2 Example II: shape optimization of a plate with a hole

The second example is concerned with optimizing the shape of an elastic 2D square plate of dimension 20 \( \times \) 20 (in self-consistent units) with a hole located at the center; see Fig. 7. The plate is subjected to a uni-axial uniform stress, \( \sigma_a \), and the goal is to minimize the maximum principal stress, i.e., the stress intensity factor. The shape of the hole is described in the polar coordinate \((r, \phi)\) with the center of the coordinate system placed at the center of the plate, i.e., in Cartesian coordinates every point on the hole boundary is described using \( x = r \cos \phi \) and \( y = r \sin \phi \). The radius of the hole is described in a harmonic basis,

\[
\begin{align*}
r &= r_0 + \tau \sum_{i=1}^{d} \frac{1}{\sqrt{i}} \left[ \theta_i^s \sin(i\phi) + \theta_i^c \cos(i\phi) \right]; \\
-1 \leq \theta_i^s, \theta_i^c \leq 1, \quad i = 1, \ldots, d,
\end{align*}
\]  

where \( \theta = [\theta_1^s, \ldots, \theta_d^s, \theta_1^c, \ldots, \theta_d^c]^T \) is the vector of optimization variables. The parameters in (29) are set to \( r_0 = 1 \), \( \tau = 0.15 \), and \( d = 3 \). Note that, these values will not result in a negative radius. We further add a contribution from the deviation of the area of the hole from a circle of radius one to the objective to avoid converging to the solution of a hole with much smaller radius. The optimization problem is given by

\[\text{minimize} \quad \text{Area}_h - 1\]
\[
\min_\theta J(\theta) = \mathbb{E} \left[ \frac{\sigma_{\text{max}}(\theta)}{\sigma_a} \right] + \lambda (\pi - A_{\text{hole}}(\theta))
\]

subject to \(-1 \leq \theta_j \leq 1, \ j = 1, \ldots, n_\theta\).

where \(\sigma_{\text{max}}(\theta)\) is the maximum value of the principal stress in the plate; \(A_{\text{hole}}(\theta)\) is the area of the hole; and we choose \(\lambda = 50\), which proved sufficient to keep the solution from converging to a hole with smaller radius. While a large value for \(\lambda\) ensures that the shape of the hole is close to a circle with unit radius this may result in gradients of the area of the hole dominating the gradients of the principal stress. The box constraint on \(\theta\) is applied here by restricting any parameter update to be within \([-1, 1]\), i.e., if the update puts the parameter outside of \([-1, 1]\) we replace the parameter with \(-1\) or \(1\), respectively. We assume uncertainty in the tensile stress \(\sigma_a\) applied at the two ends of the plate, which we model as

\[
\sigma_a(\xi_\sigma) = \sigma_0(1 + 0.5\xi_\sigma),
\]

where \(\xi_\sigma\) is a standard normal random variable and \(\sigma_0 = 1\) is a constant. Further, the elastic modulus of the plate \(E\) and Poisson’s ratio \(\nu\) are assumed uncertain and given by

\[
\begin{align*}
E(\xi_E) &= E_0(1 + 0.05\xi_E); \\
\nu(\xi_\nu) &= \nu_0(1 + 0.01\xi_\nu),
\end{align*}
\]

where \(\xi_E\) is a standard normal random variable truncated on one side to keep \(E(\xi_E) \in (0, \infty)\); \(\xi_\nu\) is a standard normal random variable truncated at both sides to get \(\nu(\xi_\nu) \in (0.3, 0.5)\); \(E_0 = 1000\); and \(\nu_0 = 0.3\). To compute the gradients of \(J(\theta)\) we use a differentiable approximation of \(\sigma_{\text{max}}/\sigma_a \approx (\sum_i (\sigma_i/\sigma_0)^p)^{1/p}\) for a large (even) \(p\), where \(\sigma_i\) are elemental principal stresses [46] and the summation is over all elements in the finite element model. In our numerical experiments, we set \(p = 200\).

To solve for the stress in the plate we use the finite element package FEniCs [2,59]. We employ forward finite difference for its simplicity with a step size of 0.01 to compute the gradients of \(J(\theta)\). Note that, a large step size may result in incorrect gradient information, whereas much smaller step size will result in numerical issues. In this example, a step size smaller than 0.01 produces the same gradient values and hence we do not refine the step size further. Note that, only a quarter of the plate is analyzed leveraging the symmetry of the problem. A low-fidelity model with \(\sim 260\) degrees of freedom is constructed using a coarse mesh while a high-fidelity model with \(\sim 10250\) degrees of freedom uses refined mesh around the hole. For a circular hole with a radius of \(r = 1\), the coarse mesh gives a relative difference of 9.1369% in \(\sigma_{\text{max}}/\sigma_a\) as compared to the fine mesh. Figure 8 shows the typical meshes used for both high- and low-fidelity models. We remesh each of these models once for every \(\theta\) but keep the total number of degrees of freedom approximately same.

Here, we assume the computational cost \(CT\) for a model with \(N_\beta\) degrees of freedom is proportional to \(N_\beta^\beta\), where \(\beta\) depends on the solver used. Hence, we can write

\[
\gamma = \frac{CT_{\text{low}}}{CT_{\text{high}}} = \frac{N_{d,\text{low}}^\beta}{N_{d,\text{high}}^\beta},
\]

where \(CT_{\text{low}}\) and \(CT_{\text{high}}\) are computational costs for a low- and a high-fidelity gradient evaluations, respectively, including any interpolation costs; and \(N_{d,\text{low}}\) and \(N_{d,\text{high}}\) are degrees of freedom for a low- and a high-fidelity model, respectively.

Solving the plate problem with an algebraic multigrid solver and using the wall-clock data for meshes with different number of degrees of freedom, we determine via data fitting a \(\gamma\) value approximately of 0.015 in this example. Note that, \(\gamma\) changes slightly as we remesh but the total number of degrees of freedom remains almost same and hence the change in \(\gamma\) is insignificant. The cost calculations are performed on a
4.2.1 Results

We use the hole shown in Fig. 12 (green dashed line) generated with \( \theta^s = [-0.1660, 0.4406, -0.9998]^T \) and \( \theta^c = [-0.3953, -0.7065, -0.8153]^T \) as the initial guess for all the optimization algorithms. Figure 9 compares the performance of the SAG and the BF-SAG algorithms for this example with a learning rate \( \eta = 0.02 \), which proved sufficient to achieve a converged solution. In our implementation of the SAG algorithm, we use \( N_h = 10 \), i.e., 10 high-fidelity gradient evaluations per iteration to keep the computational cost reasonable on a desktop computer. In the BF-SAG algorithm, at every iteration, we use \( N_h = 5 \) and \( N_l = 5 \), i.e., we update five gradients using the high-fidelity model as before but for the other five gradients we use the low-fidelity model. The evolution of the objective for the SAG and the BF-SAG algorithms is shown in Fig. 9a, which shows similar performance for both of these algorithms. Figure 9b further shows that if we normalize the computational cost in terms of the high-fidelity model evaluation, we can reach the optimum.
Objective using a fraction of the cost in the BF-SAG algorithm compared to the SAG algorithm, which only uses the high-fidelity gradients.

Next, we compare the SVRG and the proposed BF-SVRG algorithms for a learning rate $\eta = 0.02$, where we keep the number of gradient evaluations for every outer iteration the same. Here, we consider the SVRG algorithm with $N_h = 15$ and the BF-SVRG algorithm with $N_l = 5$ and $N_h = 2$.

We use inner iteration $m = 5$, which requires 25 gradient evaluations for every outer iteration for each of these algorithms. Figure 10a shows that we obtain a similar convergence, as the iteration progresses. However, in terms of the computational cost of evaluating the high-fidelity models, the BF-SVRG algorithm features a faster convergence as shown in Fig. 10b. Hence, the use of the BF-SAG and BF-SVRG algorithms are effective in reducing the computational cost of

---

**Fig. 12** Optimized shape of the hole using the proposed bi-fidelity algorithms. Initial shape is shown using a dashed line. Note that, two-fold symmetry is assumed here.

**Fig. 13** Schematic of the beam problem of Example III (a) (design domain is shown as the shaded region).

**Fig. 14** The steps to estimate the gradients using the low-fidelity model are shown here. The density variables $\rho$ for the finer mesh (high-fidelity model) are first mapped onto the coarser mesh (low-fidelity model). Then the coarser mesh is used to calculate the gradients. Finally, the coarse grid gradients are mapped again onto the finer mesh using a cubic spline interpolation.
For the third numerical example, we consider topology optimization of a beam under uncertain load. The design domain is simply supported on the bottom left and right ends with a span of $L$ and is subjected to an uncertain point load $2P$ at the mid span. The schematic for this problem is shown in Fig. 13. Using the symmetry of the problem, we only consider one-half of the span as our optimization domain. The uncertainty in the load $P$ is given by

$$P(\xi) = P_0(1 + 0.5\xi),$$  \hspace{1cm} (34)$$

where $\xi$ is a uniform random variable in $[0, 1]$ and $P_0 = 1$ is a constant.

We optimize the material distribution inside the optimization domain by minimizing a combination of the compliance (i.e., strain energy) and the mass subject to satisfying the equilibrium equations [92,94]. We divide the design domain $\Omega$ into a large number of non-overlapping elements $\{\Omega_i\}_{i=1}^{N_e}$ using a finite element approach, where $\{v_i\}_{i=1}^{N_e}$ are the corresponding volumes and $N_e$ is the total number of elements used. We use the solid isotropic material with penalization (SIMP) approach [9,94,103] to formulate this topology optimization problem, and use some parts of the widely-used 99 line topology optimization code in Sigmund [92]. In SIMP, the material properties are interpolated by a power-law model in terms of the density $\rho$ of a fictitious porous material, e.g.,
\[
E(\rho_i) = \rho_i^{\beta_P} E_0; \quad 0 < \rho_i \leq 1; \quad i = 1, 2, \ldots, N_e, \tag{35}
\]

where \(\beta_P\) is a penalization parameter and \(E_0\) is the bulk material’s elastic modulus. For the formulation of the optimization problem considered here and \(\beta_P > 1\), intermediate densities are penalized as compared to densities closer to zero or one. We use \(\beta_P = 3\) in the present work. To avoid a checkerboard design, we use filtered values of the design variables \(\theta\) to define the material density \(\rho\) [3,14,16,93]. This density filter is applied to the \(e\)th element as follows

\[
\rho_e = \frac{1}{\sum_{i=1}^{N_e} H_i} \sum_{i=1}^{N_e} H_i \theta_i, \tag{36}
\]

where the weight \(H_i = \max(0, r_f - d_{ie})\) is the difference between a filter size \(r_f\) and the distance \(d_{ie}\) between the centers of \(i\)th and \(e\)th elements. Herein, we use 1.5 times the element width as \(r_f\). Further use of projections may be needed to achieve a discrete 0–1 design [94]. However, we do not use any such projection in this paper. We write the optimization problem as

\[
\min_{\theta} J(\theta) = \mathbb{E} \left[ \sum_{i=1}^{N_e} \int_{\Omega_i} W\left( u_i(\rho_i(\theta); \xi), \rho_1(\theta); \xi \right) dV_i \right] + \lambda \sum_{i=1}^{N_e} v_i \theta_i
\]

subject to \(K(\rho(\theta); \xi) u(\rho(\theta); \xi) = f(\xi)\);  
\(0 \leq \rho_i(\theta) \leq 1\), for \(i = 1, \ldots, N_e\). \(\tag{37}\)
where the objective $J(\cdot)$ is the expected value of the integral of the strain energy density $W(\cdot; \cdot; \cdot)$ plus a contribution from the total mass of the structure; $K(\cdot; \cdot)$ is the stiffness matrix; $f(\cdot)$ is the external force vector; $\lambda$ is the weighting factor for the contribution of the total mass to the objective. The strain energy density depends on the displacement $u(\cdot; \cdot; \cdot)$ and the material density $\rho$. The displacement $u(\cdot; \cdot; \cdot)$ in turn depends on the uncertain variable $\xi$. The topology optimization problems are highly convex with multiple solutions. In this paper, we compare two designs using their respective objective values and conclude both of them as acceptable if they have similar objectives.

We construct the high-fidelity model by dividing the domain of optimization (i.e., an area of $L/2 \times L/6$) into $120 \times 40$ quadrilateral elements, which results in 4800 optimization variables. For the low-fidelity model, we use $60 \times 20$ of the same type of elements with 1200 optimization variables.

To estimate the gradients using the low-fidelity model, we follow the steps shown in Fig. 14. First, we map the density variable $\rho$ from the high-fidelity mesh to the low-fidelity one by averaging, i.e., a restriction like operation. Note that, we have one element in the low-fidelity mesh in place of four in the high-fidelity mesh. Next, we perform the calculation of the gradients using the low-fidelity mesh. Finally, we map the gradients to the elements in the high-fidelity mesh using a cubic spline interpolation, i.e., a prolongation like operation. Note that, these restriction and prolongation operations are different than used in a multigrid scheme [45]. While for the configurations considered here the mapping is simple and can be computed analytically, it can be generalized to any mesh configurations using a proper projection operator, such as an $L^2$ minimization. Averaging the computation time over 10 runs measured using cputime shows that for a MATLAB implementation of the finite element solver and the proposed
minimize compliance and a contribution from total mass

$$P(\xi) + P_0 \phi(\xi)$$

L/2

L/8

L/6

Fig. 21 Schematic for the beam problem in Example III (b) (design domain is shown as the shaded region). Only one-half of the beam is shown because of the symmetry.

(a) Final design using the SAG algorithm with $N_h = 25$.

(b) Final design using the BF-SAG algorithm with $N_l = 20$ and $N_h = 5$.

(c) Final design using the BF-SAG algorithm with $N_l = 95$ and $N_h = 5$.

Fig. 22 Final designs for Example III (b) as obtained from the SAG and the proposed BF-SAG algorithms. Note that, both algorithms use the same number of gradient evaluations per iteration in (a) and (b). However, the BF-SAG algorithm only uses 5 high-fidelity gradients compared to 25 in the SAG algorithm. In (c), we use more low-fidelity gradients per iteration.

mapping scheme the cost of such a gradient estimate is 10.47 times cheaper than that of direct calculation of the high-fidelity gradients. This leads to the cost ratio of low- and high-fidelity gradient evaluations $\gamma = 0.096$. Note that, the restriction and prolongation costs make up 6.76% of the total low-fidelity gradient calculation cost. We use the same desktop computer as in the previous example to compute the costs.

4.3.1 Results

We study the proposed algorithms for this example with a learning rate $\eta = 0.05$ and $\lambda = 0.25$ in (37). The final designs obtained from the SAG and BF-SAG algorithms with different number of low-fidelity gradient solves are shown in Fig. 15. The first two designs are similar and exhibit a truss-like topology as in the deterministic optimization problem in Sigmund [92]. The final design (Fig. 15c) that uses more low-fidelity gradient solves has a smaller mass compared to the other two designs.

The objective is plotted in Fig. 16a for these two algorithms. The result shows that the performance of the BF-SAG algorithm (solid blue curve) is comparable to the SAG algorithm (dotted green curve), when we use the same number of finite element solves per iteration. Further, if we increase the number of low-fidelity solves per iteration, we can improve the convergence as shown by the (red) dash-dotted curve. In terms of the mass of the structure, from Fig. 16b, we see that the BF-SAG algorithm with more low-fidelity gradient solves produces a structure that has a significantly smaller mass but similar total objective; note that the total objective includes a contribution from the mass. The BF-SAG algorithm with $N_l = 95$ and $N_h = 5$ is also run for three different seeds for the random number generator to see the effect of stochasticity in the proposed algorithm. Figure 17 shows that the designs are mostly similar. However, all of them achieve similar objectives.

Next, we compare the SVRG and BF-SVRG algorithms for this example. Again, we obtain similar truss-like designs (see Fig. 18). The plot of the objective in Fig. 19a shows that the SVRG algorithm diverges initially as the design changes substantially at the beginning of the optimization. One possible reason might be the poor approximation of the control variate (using $\theta_{prev}$ in Algorithm 3) that leads to a poor design and large compliance values. The use of more low-fidelity gradient samples along with the calculation of an optimal $\alpha$ in (19) improves the convergence of the solution and avoids large objective values. The BF-SVRG algorithm also leads to smaller variations in the objective and hence an improved variance reduction. Similar observations can be made from Fig. 19b for the mass ratio. These results suggest that the use of the BF-SAG and BF-SVRG algorithms can improve the convergence of the OuU problem when compared to their single fidelity counterparts. Again, to see the effect of stochasticity in the final design we run the BF-SVRG algorithm with $N_l = 20$ and $N_h = 8$ using three different seeds for the random number generator. Figure 20 shows that we obtain almost identical designs with similar objectives.
4.4 Example III (b): topology optimization of a beam under uncertain load magnitude and direction

We next consider the same beam problem as in Example III (a) but add another load at a distance \( L/8 \) from the mid-span when only one-half of the beam is considered due to symmetry; see Fig. 21. While the magnitude of this force, \( P_0 = 1 \) is deterministic, its direction \( \phi \) relative to the beam’s longitudinal axis is assumed random and given by

\[ \phi(\xi_\phi) = \frac{\pi}{4} + \xi_\phi, \]

where \( \xi_\phi \) is a uniform random variable in \([−\pi/8, \pi/8]\). Hence, in this example, we have two uncertain parameters — \( \xi \) in (34) and \( \xi_\phi \) in (38). Due to the addition of another load as compared to Example III (a), the compliance value in this example increases. This results in significantly different optimal designs. Also, this example shows a case where the SVRG algorithm fails to converge.

4.4.1 Results

In this example, we use \( \lambda = 0.25 \) in the objective function (see (37)) and a learning rate \( \eta = 0.05 \). The final designs obtained using the SAG and the proposed BF-SAG algorithms are shown in Fig. 22. The designs differ significantly in the mass they use. Note that, the mass contributes to the objective and larger mass increases the objective value but gives a smaller compliance. As a result, we reach different locally optimum designs. Figure 23a and b show that the design obtained using the BF-SAG algorithm with \( N_l = 95 \) and \( N_h = 5 \) has the smallest objective value but uses more mass. The BF-SAG algorithm with a similar number of gradient evaluations per iteration performs slightly worse than the SAG algorithm that uses only high-fidelity gradients. Interestingly, the SAG algorithm uses more high-fidelity gradients but does not converge to a design that matches the performance of the BF-SAG algorithm. We further investigate this by studying the effect of the seed of the random number generator in this example, where we run the optimization for
three different seeds. Figures 24 and 26 illustrate that the designs obtained from the BF-SAG algorithm with $N_l = 20$ and $N_h = 5$ and the SAG algorithm with $N_h = 25$ varies significantly over these runs. Their objectives are also very different. However, Fig. 25 shows that when we increase the number of low-fidelity gradient evaluation to $N_l = 95$ for the BF-SAG algorithm, the variations in the design and objective are minimal.

Further, in this example, the SVRG algorithm fails to converge. The BF-SVRG algorithm, on the other hand, produces meaningful designs as shown in Fig. 27a and b. Initially, the designs undergo drastic changes – in terms of the compliance and objective – as can be seen in Fig. 28. Since the SVRG algorithm uses a control variate of the gradient based on past design parameters this control variate is poorly correlated with the gradient at the current iteration, especially during the initial iterations. To see this, we run the SVRG algorithm, where we estimate the coefficient $\alpha$ using 4 random samples per inner iteration. In this scenario, the SVRG algorithm converges and produces a design similar to the BF-SVRG designs.

4.5 Example III (c): design of beam under uncertain load and material property

In this example, we consider the same beam problem as in Example III (a) with uncertainty in the load magnitude and material property, which are modeled as random variables. The objective is to design a beam that minimizes the compliance subject to stress and displacement constraints. The design variables are the cross-sectional area of the beam.

The optimization problem is formulated as follows:

$$
\min_{\mathbf{x}} \mathcal{C}(\mathbf{x})
$$

subject to

$$
\mathbf{g}(\mathbf{x}) = \mathbf{0},
$$

$$
\mathbf{h}(\mathbf{x}) \leq \mathbf{0},
$$

where $\mathcal{C}(\mathbf{x})$ is the compliance of the beam, $\mathbf{g}(\mathbf{x}) = \mathbf{0}$ are the stress constraints, and $\mathbf{h}(\mathbf{x}) \geq \mathbf{0}$ are the displacement constraints. $\mathbf{x}$ represents the design variables.

The stochastic nature of the problem is handled by Monte Carlo sampling, where the material and load properties are sampled from their respective probability distributions.

The optimization is performed using the BF-SAG algorithm, which is an adaptation of the SAG algorithm for optimization under uncertainty. The algorithm iteratively updates the design by solving a series of low-fidelity problems, where the gradients are calculated using only a subset of the samples. The algorithm is given by:

1. Initialize $\mathbf{x}_0$ and set $k = 0$.
2. For each iteration $k$:
   - Sample $\mathbf{n}$ random points from the probability distributions.
   - Solve a low-fidelity problem to update the design variables.
   - Update the parameters of the algorithm.
3. Stop when convergence criteria are met.

The optimization is illustrated in Figs. 24 and 25, which show the design evolution over iterations. The designs obtained are shown in Figs. 26 and 27a and b, respectively. The objectives for these three designs are, however, not similar.
Three different designs obtained using the SAG algorithm with $N_h = 25$ but with different seeds for the random number generator. The objectives for these three designs are, however, not similar (see (34)). We further assume that the elastic modulus of the material $E_0$ in (35) is uncertain and modeled by a lognormal random field,

$$E_0(x_1, x_2) = E_{\text{min}} + \exp[z(x_1, x_2)],$$

(39)

where $E_{\text{min}} = 0.1$ is used and $z(x_1, x_2)$ is a zero-mean Gaussian random field with a covariance function

$$\mathbb{E}[z(x_1, x_2)z(y_1, y_2)] = \sigma^2 \exp \left( -\frac{|x_1 - y_1|}{l_1} - \frac{|x_2 - y_2|}{l_2} \right).$$

(40)
Here, \( l_1 = l_2 = L/10 \) and \( \sigma = 2 \) are used. The random field \( z(x_1, x_2) \) is expressed using a Karhunen-Loève expansion truncated at \( N_{\text{max}} \) terms as follows,

\[
z(x_1, x_2) = \sum_{i=1}^{N_{\text{max}}} \sqrt{\lambda_i} \xi_i \psi_i(x_1, x_2),
\]

where \( \lambda_i \) are eigenvalues and \( \psi_i(x_1, x_2) \) are eigenfunctions of the covariance function \((40)\); \( \xi_i \) are independent standard normal random variables; \( N_{\text{max}} = 100 \) is selected to capture 92.86% of total variance of \( z \). Figure 29 shows four realizations of the modulus of elasticity \( E_0 \) over the design domain. The uncertainty in the load magnitude is assumed same as in (34). Hence, the dimension of the uncertain parameter vector \( \xi \) is 101 in this example, whereas dimension of the optimization variable vector \( \theta \) is 4800 as before.

4.5.1 Results

We study the proposed algorithms using a learning rate \( \eta = 0.05 \) and \( \lambda = 0.25 \) in (37). The final designs from the SAG and BF-SAG algorithms are shown in Fig. 30. It can be noted from the figure that the SAG algorithm fails to find a meaningful design and results in a higher value of objective. In the BF-SAG algorithm, if we use more low-fidelity gradient evaluations per iteration, we obtain a design (Fig. 30c) that utilizes more mass but reaches the performance of the BF-SAG algorithm with smaller low-fidelity gradient evaluations per iteration. This is further evident from the plots of objective and mass ratio for these three cases as shown in Fig. 31. The figure also shows that the BF-SAG algorithm with \( N_l = 95 \) and \( N_h = 5 \) produces smaller variations in the
5 Conclusions

In the presence of uncertainty, the cost of design optimization of structures increases many folds. Methods like polynomial chaos expansion or stochastic collocation help in reducing the cost but in the presence of high-dimensional uncertainty their costs increase rapidly as well. In this paper, to alleviate this computational burden of OuU, we propose a bi-fidelity approach with stochastic gradient descent type methods, where most of the gradients are estimated using a low-fidelity model. The gradients are then incorporated into two distinct stochastic gradient descent algorithms. In the first algorithm, we use an average of the gradients, where most of them are updated using the low-fidelity model. In the second algorithm, we use a control variate based on gradients calculated using the low-fidelity model to reduce the variance in the stochastic gradients. Linear convergence of these proposed algorithms in ideal conditions are proved. The efficacy of these algorithms is shown using three numerical examples. After studying the proposed algorithms with a conceptual problem, we optimize the shape of a hole in a square plate to minimize the maximum principal stress in the plate. In the third example, we apply the proposed algorithms to a topology optimization problem involving uncertainties in load and material properties with the number of uncertain parameters reaching 101. These examples show that using the proposed algorithms we successfully leverage a low-fidelity model to

Fig. 31 Reduction of objective and mass ratio during the optimization process for two configurations of the BF-SAG algorithm and one configuration of the SAG algorithm in Example III (c). Note that the faster convergence to the optimum can be achieved by using more low-fidelity gradients and only a handful of high-fidelity gradients per iteration.

Fig. 32 Final designs for Example III (c) as obtained from the SVRG and the proposed BF-SVRG algorithms. Note that, in (a) and (b), both algorithms use the same number of gradient evaluations per outer iteration (see Algorithm 5)
Assume \( \theta_k \) is the vector of optimization parameters after \( k \) iterations of Algorithm 4. \( \nabla_{\theta_k}^\text{low}(\theta_k) \) and \( \nabla_{\theta_k}^\text{high}(\theta_k) \) are gradients of the objective with respect to \( \theta_k \) using the low- and high-fidelity models, respectively. Under the assumption of strong convexity\(^1\) of low-fidelity and high-fidelity objectives, \( (\theta_k - \theta^*)^T \nabla_{\theta_k}^\text{low}(\theta_k) \geq \mu_{\text{low}} \| \theta_k - \theta^* \|^2 \), \( (\theta_k - \theta^*)^T \nabla_{\theta_k}^\text{high}(\theta_k) \geq \mu_{\text{high}} \| \theta_k - \theta^* \|^2 \), (42)

where \( \mu_{\text{low}} \) and \( \mu_{\text{high}} \) are constants. Similarly, if the low- and high-fidelity gradients are Lipschitz continuous, 
\[
\| \nabla_{\theta_k}^\text{low}(\theta_k) \| \leq L_{\text{low}}^2 \| \theta_k - \theta^* \|^2 ,
\| \nabla_{\theta_k}^\text{high}(\theta_k) \| \leq L_{\text{high}}^2 \| \theta_k - \theta^* \|^2 ,
\]
(43)

where \( L_{\text{low}}, L_{\text{high}} \) are the Lipschitz constants for low- and high-fidelity gradients, respectively. The parameters are updated in Algorithm 4 using 
\[
\theta_{k+1} = \theta_k - \eta \hat{h}_k.
\]
(44)

The expected value of the search direction \( \hat{h}_k \) at iteration \( k \) is 
\[
\mathbb{E}[\hat{h}_k|\theta_k] = p_l \nabla_{\theta_k}^\text{low}(\theta_k) + p_h \nabla_{\theta_k}^\text{high}(\theta_k) + (1-p_l-p_h)\mathbf{d}_{k-1},
\]
(45)

where \( p_l = N_l/N \) and \( p_h = N_h/N \).

Next, we evaluate the following expectation 
\[
\mathbb{E}[\| \theta_{k+1} - \theta^* \|^2 | \theta_k] = \mathbb{E}[\| \theta_{k+1} - \theta^* - \eta \hat{h}_k \|^2 | \theta_k]
\]
\[
= \| \theta_k - \theta^* \|^2 - 2\eta (\theta_k - \theta^*)^T \mathbb{E}[\hat{h}_k|\theta_k]
\]
\[
+ \eta^2 \mathbb{E}[\| \hat{h}_k \|^2 | \theta_k]
\]
\[
\leq \| \theta_k - \theta^* \|^2 - 2\eta (\theta_k - \theta^*)^T \mathbb{E}[\hat{h}_k|\theta_k]
\]
\[
+ \eta^2 L^2 \| \theta_k - \theta^* \|^2 ,
\]
(46)

where \( L^2 = \max\left\{ p_l (1-p_l-p_h) (k-j) L_{\text{low}}^2 \frac{\| \theta_k - \theta^* \|^2}{\| \theta_k - \theta^* \|^2} , p_h (1-p_l-p_h) (k-j) L_{\text{high}}^2 \frac{\| \theta_k - \theta^* \|^2}{\| \theta_k - \theta^* \|^2} \right\} \) for \( j = 1, \ldots, k \). Using the strong convexity property of \( J(\theta) \), 
\[
\mathbb{E}[\| \theta_{k+1} - \theta^* \|^2 | \theta_k] \leq (1 - 2\eta \mu + \eta^2 L^2) \| \theta_k - \theta^* \|^2 ,
\]
\[
\mathbb{E}[\| \theta_{k+1} - \theta^* \|^2 | \theta_0] \leq (1 - \mu^2 / L^2)^k \| \theta_0 - \theta^* \|^2 ,
\]
(47)

where \( \mu = \min\left\{ p_l (1-p_l-p_h) (k-j) \frac{\mu_{\text{low}} \| \theta_j - \theta^* \|^2}{\| \theta_j - \theta^* \|^2} , p_h (1-p_l-p_h) (k-j) \frac{\mu_{\text{high}} \| \theta_j - \theta^* \|^2}{\| \theta_j - \theta^* \|^2} \right\} \) for \( j = 1, \ldots, k \); and learning rate is chosen as \( \eta = \mu / L^2 \) subject to \( \mu^2 / L^2 \leq 1 \). This completes the proof of Theorem 1.

The constants \( \mu \) and \( L^2 \) in (47) are affected by the parameter update history as mentioned in Sect. 3.1. To see this, let us define

\( L_{\text{low}}^2 \) is convex.
\[ c_{\min}^k = \min_j \left\{ (1 - p_l - p_h)^{(k-j)} \| \theta_j - \theta^* \|^2 \right\}; \]
\[ c_{\max}^k = \max_j \left\{ (1 - p_l - p_h)^{(k-j)} \| \theta_j - \theta^* \|^2 \right\}; \]  \tag{48}

Hence, the constants \( \mu \) and \( L^2 \) can be written as
\[ \mu = \frac{c_{\min}^k}{\| \theta_k - \theta^* \|^2} \min \left\{ p_l \mu_{low}, p_h \mu_{high} \right\}; \]
\[ L^2 = \frac{c_{\max}^k}{\| \theta_k - \theta^* \|^2} \max \left\{ p_l L_{low}^2, p_h L_{high}^2 \right\}. \]  \tag{49}

Note that, if \( p_l \) and \( p_h \) are fixed \( \mu \) depends on \( c_{\min}^k \), i.e., on
\[ \min \left\{ (1 - p_l - p_h)^{(k-j)} \| \theta_j - \theta^* \|^2 \right\} \] for \( j = 1, \ldots, k \). Further, \( (1 - p_l - p_h)^{(k-j)} \) increases with \( j \) since \( 1 - p_l - p_h < 1 \) but \( \| \theta_j - \theta^* \|^2 \) depends on the parameter update history \( \{ \theta_j \}_{j=1}^k \). Similarly, \( L^2 \) depends on \( c_{\max}^k \) and in turn on \( \{ \theta_j \}_{j=1}^k \). Hence, the parameter update history affects \( \mu \) and \( L^2 \).

**B Proof of Theorem 2**

Using the assumption of strong convexity of objective obtained from the high-fidelity models,
\[ (\theta_k - \theta^*)^T h_{\text{high}}(\theta_k) \geq \mu_{\text{high}} \| \theta_k - \theta^* \|^2, \]  \tag{50}

where \( \mu_{\text{high}} \) is a constant. Similarly, if the high-fidelity gradients are Lipschitz continuous
\[ \| h_{\text{high}}(\theta_k) \|^2 \leq L_{\text{high}}^2 \| \theta_k - \theta^* \|^2, \]  \tag{51}

where \( L_{\text{high}} \) is the Lipschitz constant. For the inner iterations, we can evaluate the following expectation
\[ \mathbb{E}[\| \theta_{k+1} - \theta^* \|^2 | \theta_k, \theta_{prev}] = \mathbb{E} \left[ \left\| \theta_k - \theta^* - \eta \left( \frac{\alpha}{N_h} \sum_{i=1}^{N_h} h_{\text{low}}(\theta_{prev}; \xi_i) - \widehat{h}_{\text{low}} \right) \right\|^2 \right] \]
\[ = \| \theta_k - \theta^* \|^2 - 2\eta (\theta_k - \theta^*)^T \widehat{h}_{\text{high}}(\theta_k) + \eta^2 \| \widehat{h}_{\text{high}}(\theta_k) \|^2 \]
\[ + \eta^2 \sum_{q=1}^{n_q} \text{Var} \left( \widehat{h}_{\text{high},q} \frac{\alpha_q}{N_h} \sum_{i=1}^{N_h} (h_{\text{low},q}(\theta_{prev}; \xi_i) - \widehat{h}_{\text{low},q}) \right) \]
\[ - \| \widehat{h}_{\text{low},q} \|^2 \| \theta_k, \theta_{prev} \). \]  \tag{52}

where \( h_q \) is the gradient with respect to \( \theta_q \) and \( \text{Var}(\cdot) \) denotes variance of its argument. Note that, if \( \widehat{h}_{\text{low}} = \mathbb{E}[h_{\text{low}}(\theta; \xi)] \) exactly,
\[ \text{Var} \left( \widehat{h}_{\text{high},q} - \frac{\alpha_q}{N_h} \sum_{i=1}^{N_h} (h_{\text{low},q}(\theta_{prev}; \xi_i) - \widehat{h}_{\text{low},q}) \right) \theta_k, \theta_{prev} \]
\[ = \frac{1}{N_h} (1 - \rho_{hl,q}^2) \text{Var}(h_{\text{high},q}(\theta_k; \xi)), \]  \tag{53}

where \( \alpha_{q,q} = \text{Cov}(h_{\text{low},q}(\theta_{prev}; \xi), h_{\text{high},q}(\theta_k; \xi))/\text{Var}(h_{\text{low},q}(\theta_{prev}; \xi)) \) and the correlation coefficient \( \rho_{hl,q} = \text{Cov}(h_{\text{low},q}(\theta_{prev}; \xi), h_{\text{high},q}(\theta_k; \xi))/\sqrt{\text{Var}(h_{\text{low},q}(\theta_{prev}; \xi)) \text{Var}(h_{\text{high},q}(\theta_k; \xi))} \). On the other hand, if we use \( N_l \) samples to estimate \( \widehat{h}_{\text{low}} \), i.e., \( \widehat{h}_{\text{low}} = \frac{1}{N_l} \sum_{i=1}^{N_l} h_{\text{low}}(\theta_{prev}; \xi_i) \) then we can write
\[ \text{Var} \left( \widehat{h}_{\text{high},q} - \frac{\alpha_q}{N_h} \sum_{i=1}^{N_h} (h_{\text{low},q}(\theta_{prev}; \xi_i) - \widehat{h}_{\text{low},q}) \right) \theta_k, \theta_{prev} \]
\[ = \frac{1}{N_h} \left( 1 - \frac{\rho_{hl,q}^2}{1 + N_h/N_l} \right) \text{Var}(h_{\text{high},q}(\theta_k; \xi)), \]  \tag{54}

where the coefficient \( \alpha_{q,q} \) is obtained by minimizing the mean-squared error in \( \widehat{h}_{\text{high},q} \) [33,76], i.e.,
\[ \alpha_{q,q} = \frac{\text{Cov}(h_{\text{low},q}(\theta_{prev}; \xi), h_{\text{high},q}(\theta_k; \xi))}{\text{Var}(h_{\text{low},q}(\theta_{prev}; \xi))} \left( \frac{1}{1 + N_h/N_l} \right), \]  \tag{55}

and the correlation coefficient \( \rho_{hl,q} \) is same as before. Next, let us assume
\[ \frac{1}{N_h} \left( 1 - \frac{\rho_{hl,q}^2}{1 + N_h/N_l} \right) \text{Var}(h_{\text{high},q}(\theta_k; \xi)) \leq L_{\text{high}}^2 \delta_{k,q} \| \theta_k - \theta^* \|^2 \]  \tag{56}

for some constants \( \delta_{k,q} \). Further, assume \( \delta_k = \max\{1, \delta_{k,q}\} \) for \( q = 1, \ldots, n_\theta \). Hence,
\[ \mathbb{E}[\| \theta_{k+1} - \theta^* \|^2 | \theta_k, \theta_{prev}] \leq (1 - 2\eta \mu_{\text{high}} + \eta^2 L_{\text{high}}^2) \| \theta_k - \theta^* \|^2 \]
\[ + \eta^2 L_{\text{high}}^2 \delta_k \| \theta_k - \theta^* \|^2 \]
\[ \leq (1 - 2\eta \mu_{\text{high}} + 2\eta^2 L_{\text{high}}^2 \delta_k) \| \theta_k - \theta^* \|^2 \]  \tag{57}

\[ \copyright \ Springer \]
At $k$th inner iteration let us use the learning rate $\eta = \frac{\mu_{\text{high}}}{2L_{\text{high}}\delta_k}$. This leads to

$$E[\|\theta_{k+1} - \theta^\star\|^2 | \theta_{\text{prev}}] \leq \left( 1 - \frac{\mu_{\text{high}}}{2L_{\text{high}}\delta_k} \right)^k \|\theta_{\text{prev}} - \theta^\star\|^2,$$

(58)

$$E[\|\tilde{\theta}_j - \theta^\star\|^2 | \theta_0] \leq \left( 1 - \frac{\mu_{\text{high}}}{2L_{\text{high}}\delta} \right)^j \|\theta_0 - \theta^\star\|^2,$$

(59)

where $\delta = \min\{\delta_i\}_{i=1}^k$ and $\theta_1 = \theta_{\text{prev}}$. Similarly, for $j$th outer iteration,

where $\delta = \min\{\delta_i\}_{i=1}^j$ subjected to $\frac{\mu_{\text{high}}}{2L_{\text{high}}\delta} \leq 1$ and this proves Theorem 2.

Note that, if $\rho_{\text{hl}}$ is close to 1, i.e., the low- and the high-fidelity models are highly correlated, then $\delta_k$ can be assumed small. This implies that $\delta$ will be close to 1 and, thus, we can use a larger learning rate $\eta$. This, in turn, leads to a smaller right hand in (59) and a tighter bound on $E[\|\tilde{\theta}_j - \theta^\star\|^2 | \theta_0]$.

References

1. Allaire D, Willcox K, Toupet O (2010) A Bayesian-based approach to multifidelity multidisciplinary design optimization. In: 13th AIAA/ISSMO multidisciplinary analysis optimization conference. p 9183
2. Alnæs MS, Blechta J, Hake J, Johansson A, Logg A, Richardson C, Ring J, Rognes ME, Wells GN (2015) The FENICS project version 1.5. Arch Numer Softw 3(100):9–23
3. Andereassen E, Clausen A, Schevenels M, Lazarov BS, Sigmund O (2011) Efficient topology optimization in Matlab using 88 lines of code. Struct Multidiscip Optim 43(1):1–16
4. Babuška I, Hake J, Jørgens M, Jørgens ME, Wells GN (2015) The FENICS project version 1.5. Arch Numer Softw 3(100):9–23
5. Bakr MH, Bandler JW, Madsen K, Søndergaard J (2000) Review of the space mapping approach to engineering optimization and modeling. Optim Eng 1(3):241–276
6. Bakr MH, Bandler JW, Madsen K, Søndergaard J (2001) An introduction to the space mapping technique. Optim Eng 2(4):369–384
7. Bandler JW, Biernacki RM, Chen SH, Grobelny PA, Hemmers RH (1994) Space mapping technique for electromagnetic optimization. IEEE Trans Microw Theory Tech 42(12):2536–2544
8. Bendse MP (1989) Optimal shape design as a material distribution problem. Struct Optim 1(4):193–202
9. Blatman G, Sudret B (2010) An adaptive algorithm to build up sparse polynomial chaos expansions for stochastic finite element analysis. Probab Eng Mech 25(2):183–197
10. Booker AJ, Dennis JE, Frank PD, Serafini DB, Torczon V, Trosset MW (1999) A rigorous framework for optimization of expensive functions by surrogates. Struct Optim 17(1):1–13
11. Bottou L (2010) Large-scale machine learning with stochastic gradient descent. In: Proceedings of COMPSTAT’2010. Springer, pp 177–186
12. Bottou L, Curtis FE, Nocedal J (2018) Optimization methods for large-scale machine learning. SIAM Rev 60(2):223–311
13. Bourdin B (2001) Filters in topology optimization. Int J Numer Methods Eng 50(9):2143–2158
14. Boyle S, Vandenbergh L (2004) Convex optimization. Cambridge University Press, Cambridge
15. Brooks TE, Tortorelli DA (2001) Topology optimization of nonlinear elastic structures and compliant mechanisms. Comput Methods Appl Mech Eng 190(26–27):3443–3459
16. Bucker WM (2008) Uncertainty in structural engineering. Pract Period Struct Des Construct 13(1):24–30
17. Calafiore GC, Dabbene F (2008) Optimization under uncertainty with applications to design of truss structures. Struct Multidiscip Optim 35(3):189–200
18. Chen SH, Yang XW, Wu BS (2000) Static displacement reanalysis of structures using perturbation and pade approximation. Comput Methods Appl Mech Eng 16(2):75–82
19. Choi S, Alonso JJ, Kroo IM, Wintzer M (2008) Multifidelity design optimization of low-boom supersonic jets. J Airrc 45(1):106–118
20. Christensen DE (2012) Multifidelity methods for multidisciplinary design under uncertainty. Master’s thesis, Massachusetts Institute of Technology
21. De S, Hampton J, Maute K, Doostan A (2020) Topology optimization under uncertainty using a stochastic gradient-based approach. Struct Multidiscip Optim (accepted)
22. De S, Wojtkiewicz SF, Johnson EA (2017) Efficient optimal design and design-under-uncertainty of passive control devices with application to a cable-stayed bridge. Struct Control Health Monit 24(2):e1846
23. Defazio A, Bottou L (2018) On the ineffectiveness of variance reduced optimization for deep learning. ArXiv preprint arXiv:1812.04529
24. Diwekar U (2008) Optimization under uncertainty. In: Introduction to applied optimization. Springer, pp 1–54
25. Diwekar UM, Kalagnanam JR (1997) Efficient sampling technique for optimization under uncertainty. AIChE J 43(2):440–447
26. Doostan A, Geraci G, Iaccarino G (2016) A bi-fidelity approach for uncertainty quantification of heat transfer in a rectangular ribbed channel. In: ASME turbo expo 2016: turbomachinery technical conference and exposition. American Society of Mechanical Engineers, p V02CT4A031
27. Doostan A, Owhadi H (2011) A non-adapted sparse approximation of PDE with stochastic inputs. J Comput Phys 230(8):3015–3034
28. Doostan A, Owhadi H, Lashgari A, Iaccarino G (2009) Non-adapted sparse approximation of PDEs with stochastic inputs. Technical report annual research brief, Center for Turbulence Research, Stanford University
29. Duchi J, Hazan E, Singer Y (2011) Adaptive subgradient methods for online learning and stochastic optimization. J Mach Learn Res 12(7):2121–2159
30. Eldred MS, Elman HC (2011) Design under uncertainty employing stochastic expansion methods. Int J Uncertain Quantif 1(2):119–146
33. Fairbanks HR, Doostan A, Ketelsen C, Iaccarino G (2017) A low-rank control variate for multilevel Monte Carlo simulation of high-dimensional uncertain systems. J Comput Phys 341:121–139
34. Fairbanks HR, Jofre L, Geraci G, Iaccarino G, Doostan A (2018) Bi-fidelity approximation for uncertainty quantification and sensitivity analysis of irradiated particle-laden turbulence. ArXiv preprint arXiv:1808.05742
35. Fernández-Godino MG, Park C, Kim N-H, Haftka RT (2016) Review of multi-fidelity models. ArXiv preprint arXiv:1609.07196
36. Fischer CC, Grandhi RV, Beran PS (2017) Bayesian low-fidelity correction approach to multi-fidelity aerospace design. In: 58th AIAA/ASCE/AHS/ASC structures, structural dynamics, and materials conference, p 0133
37. Forrester AI, Sóbester A, Keane AJ (2007) Multi-fidelity optimization via surrogate modelling. Proc R Soc Lond A Math Phys Eng Sci 463(2088):3251–3269
38. Ghanem RG, Spanos PD (2003) Stochastic finite elements: a spectral approach. Dover publications, New York
39. Gorodetsky AA, Geraci G, Eldred MS, Jakeman JD (2020) A generalized approximate control variate framework for multifidelity uncertainty quantification. J Comput Phys 408:109257
40. Hammersley J (2013) Monte Carlo methods. Springer, Berlin
41. Hampton J, Doostan A (2016) Compressive sampling methods for sparse polynomial chaos expansions. Handbook of uncertainty quantification, pp 1–29
42. Hampton J, Doostan A (2018) Basis adaptive sample efficient polynomial chaos (BASE-PC). J Comput Phys 371:20–49
43. Hampton J, Fairbanks HR, Narayan A, Doostan A (2018) Practical error bounds for a non-intrusive bi-fidelity approach to parametric/stochastic model reduction. J Comput Phys 368:315–332
44. Hasselman T (2001) Quantification of uncertainty in structural dynamic models. J Aerosp Eng 14(4):158–165
45. Henson VE, Briggs WL, McCormick SF (2000) A multigrid tutorial. Society for Industrial and Applied Mathematics, Philadelphia
46. Holmberg E, Torstenfelt B, Klarbring A (2013) Stress constrained topology optimization. Struct Multidiscip Optim 48(1):33–47
47. Huang D, Allen TT, Notz WI, Miller RA (2006) Sequential kriging optimization using multiple-fidelity evaluations. Struct Multidiscip Optim 32(5):369–382
48. Hurtado JE (2002) Reanalysis of linear and nonlinear structures using iterated Shanks transformation. Comput Methods Appl Mech Eng 191(37–38):4215–4229
49. Jin R, Du X, Chen W (2003) The use of metamodelling techniques for optimization under uncertainty. Struct Multidiscip Optim 25(2):99–116
50. Johnson R, Zhang T (2013) Accelerating stochastic gradient descent using predictive variance reduction. In: Advances in neural information processing systems, pp 315–323
51. Keane A (2003) Wing optimization using design of experiment, response surface, and data fusion methods. J Airvic 40(4):741–750
52. Keane AJ (2012) Cokriging for robust design optimization. AIAA J 50(11):2351–2364
53. Kennedy MC, O’Hagan A (2001) Bayesian calibration of computer models. J R Stat Soc Ser B Stat Methodol 63(3):425–464
54. Kimura D, Ba J (2014) Adam: a method for stochastic optimization. ArXiv preprint arXiv:1412.6980
55. Kirsch U (2000) Combined approximations—a general reanalysis approach for structural optimization. Struct Multidiscip Optim 25(2):99–116
56. Koutsourelakis P-S (2009) Accurate uncertainty quantification using inaccurate computational models. SIAM J Sci Comput 31(5):3274–3300
57. Koziel S, Tesfahunegn Y, Amrit A, Leifsson LT (2016) Rapid multi-objective aerodynamic design using co-kriging and space mapping. In: 57th AIAA/ASCE/AHS/ASC structures, structural dynamics, and materials conference, p 0418
58. Kuo I, Willcox K, March A, Haas A, Rajanarayan D, Kays C (2010) Multifidelity analysis and optimization for supersonic design. Technical report CR-2010-216874, NASA
59. Logg A, Mardal K-A, Wells GN et al (2012) Automated solution of differential equations by the finite element method. Springer, Berlin
60. Luenberger DG, Ye Y (1984) Linear and nonlinear programming, vol 2. Springer, Berlin
61. March A, Willcox K (2012a) Constrained multifidelity optimization using model calibration. Struct Multidiscip Optim 46(1):93–109
62. March A, Willcox K (2012b) Provably convergent multifidelity optimization algorithm not requiring high-fidelity derivatives. AIAA J 50(5):1079–1089
63. March A, Willcox K, Wang Q (2011) Gradient-based multifidelity optimisation for aircraft design using Bayesian model calibration. Aeronaut J 115(1174):729–738
64. Martin JD, Simpson TW (2005) Use of kriging models to approximate deterministic computer models. AIAA J 43(4):853–863
65. Maute K, Petitl CL (2006) Uncertainty quantification and design under uncertainty of aerospace systems. Struct Infrastruct Eng 2(3–4):159–159
66. Myers DE (1982) Matrix formulation of co-kriging. J Int Assoc Math Geol 14(3):249–257
67. Narayan A, Gittelson C, Xiu D (2014) A stochastic collocation algorithm with multifidelity models. SIAM J Sci Comput 36(2):A495–A521
68. Nemirovski A, Juditsky A, Lan G, Shapiro A (2009) Robust stochastic approximation approach to stochastic programming. SIAM J Optim 19(4):1574–1609
69. Ng LW-T, Willcox KE (2014) Multifidelity approaches for optimization under uncertainty. Int J Numer Methods Eng 100(10):746–772
70. Ng LW-T, Eldred M (2012) Multifidelity uncertainty quantification using non-intrusive polynomial chaos and stochastic collocation. In: 53rd AIAA/ASCE/AHS/ASC structures, structural dynamics and materials conference 20th AIAA/ASME/AHS adaptive structures conference 14th AIAA, p 1852
71. Nobile F, Tempone R, Webster CG (2008) A sparse grid stochastic collocation method for partial differential equations with random input data. SIAM J Numer Anal 46(5):2309–2345
72. Nocedal J, Wright S (2006) Numerical optimization. Springer, Berlin
73. Padron AS, Alonso JJ, Eldred MS (2016) Multi-fidelity methods in aerodynamic robust optimization. In: 18th AIAA non-deterministic approaches conference, p 0680
74. Park C, Haftka RT, Kim NH (2017) Remarks on multi-fidelity surrogates. Struct Multidiscip Optim 55(3):1029–1050
75. Parussini L, Venturi D, Perdikaris P, Karniadakis GE (2017) Multi-fidelity Gaussian process regression of random fields. J Comput Phys 336:36–50
76. Pasupathy R, Schweizer BW, Taaffe MR, Wang J (2012) Control-variate estimation using estimated control means. IIE Trans 44(5):381–385
77. Peherstorfer B, Cui T, Marzouk Y, Willcox K (2016) Multifidelity and multifidelity optimization algorithm not requiring high-fidelity derivatives. AIAA J 50(5):1079–1089
78. Peherstorfer B, Willcox K, Gunzburger M (2018) Survey of multifidelity models. ArXiv preprint arXiv:1808.05742
79. Perdikaris P, Venturi D, Royset JO, Karniadakis GE (2015) Multifidelity modelling via recursive co-kriging and Gaussian–Markov random fields. Proc R Soc A Math Phys Eng Sci 471(2179):20150018
80. Qian PZ, Wu CJ (2008) Bayesian hierarchical modeling for integrating low-accuracy and high-accuracy experiments. Technometrics 50(2):192–204
81. Robbins H, Monro S (1951) A stochastic approximation method. Ann Math Stat 22:400–407
82. Robinson T, Eldred M, Willcox K, Haimes R (2008) Surrogate-based optimization using multifidelity models with variable parameterization and corrected space mapping. AIAA J 46(11):2814–2822
83. Ross SM (2013) Simulation, 5th edn. Academic Press, Cambridge
84. Roux NL, Schmidt M, Bach FR (2012) A stochastic gradient method with an exponential convergence rate for finite training sets. In: Advances in neural information processing systems, pp 2663–2671
85. Rubinstein RY, Kroese DP (2016) Simulation and the Monte Carlo method, vol 10. Wiley, New York
86. Ruder S (2016) An overview of gradient descent optimization algorithms. ArXiv preprint arXiv:1609.04747
87. Sahinidis NV (2004) Optimization under uncertainty: state-of-the-art and opportunities. Comput Chem Eng 28(6–7):971–983
88. Sandgren E, Cameron TM (2002) Robust design optimization of structures through consideration of variation. Comput Struct 80(20–21):1605–1613
89. Sandridge CA, Haftka RT (1989) Accuracy of eigenvalue derivatives from reduced-order structural models. J Guid Control Dyn 12(6):822–829
90. Schmidt M, Le Roux N, Bach F (2017) Minimizing finite sums with the stochastic average gradient. Math Program 162(1–2):83–112
91. Senior A, Heigold G, Ranzato M, Yang K (2013) An empirical study of learning rates in deep neural networks for speech recognition. In: 2013 IEEE international conference on acoustics, speech and signal processing. IEEE, pp 6724–6728
92. Sigmund O (2001) A 99 line topology optimization code written in Matlab. Struct Multidiscip Optim 21(2):120–127
93. Sigmund O (2007) Morphology-based black and white filters for topology optimization. Struct Multidiscip Optim 33(4–5):401–424
94. Sigmund O, Maute K (2013) Topology optimization approaches: a comparative review. Struct Multidiscip Optim 48(6):1031–1055
95. Skinner RW, Doostan A, Peters EL, Evans JA, Jansen KE (2019) Reduced-basis multifidelity approach for efficient parametric study of NACA airfoils. AIAA J 57:1481–1491
96. Spillers WR, MacBain KM (2009) Structural optimization. Springer, Berlin
97. Wang C, Chen X, Smola AJ, Xing EP (2013) Variance reduction for stochastic gradient optimization. In: Advances in neural information processing systems, pp 181–189
98. Weickum G, Eldred M, Maute K (2006) Multi-point extended reduced order modeling for design optimization and uncertainty analysis. In: 47th AIAA/ASME/ASCE/AHS/ASC structures, structural dynamics, and materials conference 14th AIAA/ASME/AHS adaptive structures conference 7th, p 2145
99. Xiu D, Karniadakis GE (2002) The Wiener-Askey polynomial chaos for stochastic differential equations. SIAM J Sci Comput 24(2):619–644
100. Yamazaki W, Rumpfkeil M, Mavriplis D (2010) Design optimization utilizing gradient/hessian enhanced surrogate model. In: 28th AIAA applied aerodynamics conference, p 4363
101. Zhang C, Friswell M, Mottershead J (2005) A review of robust optimal design and its application in dynamics. Comput Struct 83(4–5):315–326
102. Zeiler MD (2012) ADADelta: an adaptive learning rate method. ArXiv preprint arXiv:1212.5701
103. Zhou M, Rozvany G (1991) The COC algorithm, part II: topological, geometrical and generalized shape optimization. Comput Methods Appl Mech Eng 89(1):309–336

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