SIAM/ASA J. UNCERTAINTY QUANTIFICATION
Vol. 7, No. 1, pp. 174–202



# Robust Optimization of PDEs with Random Coefficients Using a Multilevel Monte Carlo Method*

Andreas Van Barel† and Stefan Vandewalle†

**Abstract.** This paper addresses optimization problems constrained by partial differential equations with uncertain coefficients. In particular, the robust control problem and the average control problem are considered for a tracking type cost functional with an additional penalty on the variance of the state. The expressions for the gradient and Hessian corresponding to either problem contain expected value operators. Due to the large number of uncertainties considered in our model, we suggest evaluating these expectations using a multilevel Monte Carlo (MLMC) method. Under mild assumptions, it is shown that this results in the gradient and Hessian corresponding to the MLMC estimator of the original cost functional. Furthermore, we show that the use of certain correlated samples yields a reduction in the total number of samples required. Two optimization methods are investigated: the nonlinear conjugate gradient method and the Newton method. For both, a specific algorithm is provided that dynamically decides which and how many samples should be taken in each iteration. The cost of the optimization up to some specified tolerance $\tau$ is shown to be proportional to the cost of a gradient evaluation with requested root mean square error $\tau$. The algorithms are tested on a model elliptic diffusion problem with lognormal diffusion coefficient. An additional nonlinear term is also considered.

**Key words.** robust optimization, stochastic PDEs, multilevel Monte Carlo, optimal control, uncertainty, gradient, Hessian

**AMS subject classifications.** 35Q93, 65C05, 65K10, 49M05, 49M15

**DOI.** 10.1137/17M1155892

## 1. Introduction.

We consider the optimization of a tracking type cost functional constrained by a partial differential equation (PDE) containing uncertain coefficients. The goal is to find an optimum that is satisfactory in a broad parameter range and that is as insensitive as possible to parameter uncertainties. To that end we solve the so-called robust control problem, in which the expected value of the cost functional is optimized. Other problem formulations that take into account the uncertainties can be found in [8, 7, 2, 23, 26]. They differ mainly in computational cost and in the robustness of the obtained optimum. Several techniques to solve the robust control problem have been described previously, in particular, stochastic collocation methods [36, 9, 38, 11, 10] and stochastic Galerkin schemes [36, 27]. These are based on earlier methods for simulation problems [3, 4, 42, 41, 33]. These methods are mainly used

*Received by the editors November 7, 2017; accepted for publication (in revised form) November 19, 2018; published electronically January 30, 2019.

http://www.siam.org/journals/juq/7-1/M115589.html

**Funding:** This research was supported by project IWT/SBO EUFORIA, "Efficient Uncertainty Quantification for Optimization in Robust Design of Industrial Applications" (IWT-140068), of the Agency for Innovation by Science and Technology, Flanders, Belgium. The work of the first author was also supported by a Ph.D. fellowship of the Research Foundation, Flanders, Belgium.

†Department of Computer Science, KU Leuven, Celestijnenlaan 200A, 3001 Heverlee, Belgium (andreas.vanbarel@cs.kuleuven.be, stefan.vandewalle@cs.kuleuven.be).

for relatively small stochastic dimensions, because the amount of collocation points increases rapidly with the dimension. Furthermore, Galerkin schemes may run into memory problems. Many techniques sample the problem in some way and use a multigrid solver on the resulting equations. This effectively comes down to taking the same number of samples on all levels in the multigrid hierarchy. Fundamentally different is the method proposed by Kouri [21], in which the multigrid optimization (MG/OPT) framework [31, 28] is applied to a hierarchy of stochastic discretizations. "Finer" levels correspond to taking a larger number of sample points in the stochastic space. Finally, Newton methods have also been applied successfully to stochastic problems; see, e.g., [30].

The computation of the gradient and the Hessian vector product corresponding to the robust optimization problem entail the solution of a system of PDEs containing uncertain coefficients and expected value operators. Due to the large number of uncertainties considered in our work, we propose to evaluate these expected values using a multilevel Monte Carlo (MLMC) method. This is motivated by the recent developments in MLMC methods for the simulation of elliptic [12, 37, 17] and other [5] PDEs with uncertain coefficients. The MLMC method reduces the computational cost by taking most samples on coarse grids and refining the resulting estimate using fewer samples on finer grids. This idea is mainly responsible for the substantial performance increase of our method w.r.t. methods that implicitly take the same number of samples on every grid. Recently, an MLMC method was proposed to solve the pathwise control problem [2], which consists of calculating the average of many optimal control solutions for different realizations of the PDE constraints. However, the resulting control is not guaranteed to be robust.

The method described in this paper solves the robust control problem. It retains the positive aspects of some of the previously described methods while avoiding some of the drawbacks. In particular, our method uses a different number of samples on different spatial discretization levels, it limits memory use by only storing a few samples of the state at any given time, and it reduces cost by adapting the precision (and thus the amount of samples) to the current stage of the optimization process; see also [22]. Furthermore, the method dynamically choses the number of samples such that a solution satisfying a requested tolerance on the gradient norm of the original (unsampled) problem can be obtained. The method can also deal with an additional cost functional term for the variance on the state, as in [36]. The method is especially suited for a large number of stochastic dimensions. If the samples are carefully taken, the resulting calculated gradient and Hessian are shown to be exact for some cost functional. Under mild conditions, this cost functional is equal to the one calculated using MLMC. Furthermore, we demonstrate that it is possible to have cheaper samples if correlated samples are allowed. This requires a slight extension of the classic MLMC theory. For the problems in this paper, the effect of the correlations is such that fewer samples are required.

The paper is structured as follows. In section 2, it is shown that the robust control formulation and the average control formulation are essentially equivalent for the tracking type cost functional with additional variance term. Section 3 introduces the model PDE, describes the properties of the stochastic variables, and explains how they are sampled. Expressions for the gradient and Hessian are derived in section 4. The proposed optimization methods follow the so-called reduced approach, i.e., the state is eliminated. Because the state is stochastic, the alternative would imply storing all realizations of the state in memory, which

we want to avoid. Section 5 summarizes the existing MLMC theory and provides details on how function valued quantities of interest can be dealt with. Section 6 applies the MLMC method on the equations derived in section 4. Section 7 investigates two optimization methods: the gradient based nonlinear conjugate gradient (NCG) method and the Newton method. For both, a specific algorithm is provided that dynamically decides which and how many samples should be used in each iteration. The cost of the optimization up to some specified tolerance $\tau$ on the gradient norm is shown to be proportional to the cost of a gradient evaluation with requested root mean square error (RMSE) $\tau$. The algorithms are tested on a model elliptic diffusion problem with lognormal diffusion coefficient in section 8. An additional nonlinear term is also considered. Finally, we end with some concluding remarks in section 9.

**2. Cost functional.** Let $(\Omega, \mathcal{A}, \mu)$ denote a probability space. The sample space $\Omega$ contains all possible realizations $\omega$ of the random influence. Its dimension is the stochastic dimension of the problem and may be infinite. $\mathcal{A}$ is the set of all events (subsets of $\Omega$) and $\mu$ is a measure that maps events in $\mathcal{A}$ to probabilities in $[0, 1]$. The expected value operator, the variance operator and the standard deviation operator, of a stochastic variable $k$ are denoted as follows:

$$\mathbb{E}[k] = \int_\Omega k \, \mathrm{d}\mu(\omega), \qquad \mathbb{V}[k] = \mathbb{E}[(k - \mathbb{E}[k])^2] = \mathbb{E}[k^2] - \mathbb{E}[k]^2, \qquad \mathbb{S}[k] = \sqrt{\mathbb{V}[k]}.$$

Assume a domain $D \subset \mathbb{R}^d$ on which the state $y$, some target state $y_D$, and the control $u$ are defined. In this paper, we consider the stochastic equivalent to the following classical deterministic goal function of tracking type:

$$J_{\mathrm{det}}(y, u) = \|y - y_D\|^2 + \alpha \|u\|^2 \tag{1}$$

with $\alpha > 0$. The norm $\|.\|$ denotes the $L^2$-norm in $D$ induced by the classical inner product $(.,.)$ in $L^2(D)$. Consider now the case where, due to uncertainties in the state equations, $y$ is stochastic. The cost functional can then be made deterministic again in several ways [2, 7]. The robust control problem attempts to minimize the mean of the cost functional, yielding

$$J_{\mathrm{rob}}(y, u) = \mathbb{E}[\|y - y_D\|^2] + \gamma \|\mathbb{S}[y]\|^2 + \alpha \|u\|^2. \tag{2}$$

The term $\|\mathbb{S}[y]\|^2 = \int_D \mathbb{V}[y] \, \mathrm{d}x$ was added because it is desirable to have a control for which the state is more accurately known, leading to a risk averse optimum. Note that the first term minimizes the expected distance to the target function $y_D$, which is not the same as minimizing the distance of the expected state to the target function. The latter is called the *average control* cost functional

$$J_{\mathrm{av}}(y, u) = \|\mathbb{E}[y] - y_D\|^2 + \gamma' \|\mathbb{S}[y]\|^2 + \alpha \|u\|^2. \tag{3}$$

Both cost functionals can easily be shown to be convex. Moreover, for $L^2$-norm tracking, we can prove that both are essentially equivalent.

Theorem 2.1 (equivalence of robust and average control). *Assume* $\|\mathbb{S}[y]\| \neq 0$*; then* $J_{rob} = J_{av}$ *iff* $\gamma' = 1 + \gamma$.

*Proof.* By switching the order of integration, we have

$$\|\mathbb{S}[y]\|^2 = \int_D \sqrt{\mathbb{E}[(y-\mathbb{E}[y])^2]}^2 \,\mathrm{d}x = \int_D \int_\Omega (y-\mathbb{E}[y])^2 \,\mathrm{d}\mu(\omega)\,\mathrm{d}x = \mathbb{E}[\|y-\mathbb{E}[y]\|^2].$$

We can now write

$$\begin{aligned}\mathbb{E}[\|y-y_D\|^2] &= \mathbb{E}[\|\mathbb{E}[y]-y_D+y-\mathbb{E}[y]\|^2]\\ &= \mathbb{E}[\|\mathbb{E}[y]-y_D\|^2]+\mathbb{E}[\|y-\mathbb{E}[y]\|^2]+\mathbb{E}[2(\mathbb{E}[y]-y_D, y-\mathbb{E}[y])]\\ &= \|\mathbb{E}[y]-y_D\|^2+\|\mathbb{S}[y]\|^2.\end{aligned}$$

The quantity $\mathbb{E}[y]-y_D$ is deterministic. Hence, the last term drops out because $\mathbb{E}[y-\mathbb{E}[y]]=0$. It is now clear that

$$J_{\mathrm{rob}}(y,u) = \|\mathbb{E}[y]-y_D\|^2+(1+\gamma)\|\mathbb{S}[y]\|^2+\alpha\|u\|^2 = J_{\mathrm{av}}(y,u)$$

iff $\gamma' = 1+\gamma$. ∎

In [36] both robust and average control cost functionals are considered. Theorem 2.1 explains why two seemingly different problems produced the same result.[1] The robust control cost functional (2) will be denoted simply as $J$ in the remainder of this paper.

**3. Model problem PDE constraint.** The method that we shall propose does not assume any specific PDE. However, to make matters more concrete and to simplify some expressions, we will focus our exposition on an elliptic model problem. Consider an object occupying the spatial domain $D = [0,1]^d \subset \mathbb{R}^d$ and denote its boundary by $\partial D$. The temperature distribution on $D$ constitutes the state $y$. The control $u$ is a heat source (or sink) on $D$ which we assume to be constant in time. The heat conduction coefficient is a stochastic field $k : D\times\Omega \to \mathbb{R} : (x,\omega) \mapsto k(x,\omega)$. With Dirichlet boundary conditions, the system equations are now described by the following PDE with random coefficients:

$$\begin{aligned}-\nabla\cdot(k(x,\omega)\nabla y(x,\omega)) &= \beta(x)u(x) \quad \text{on } D,\\ y(x,\omega) &= 0 \qquad\qquad\quad \text{on } \partial D.\end{aligned} \tag{4}$$

The coefficient $\beta(x)$ allows us to constrain the control input to a subset of $D$, by setting it to 1 if $x$ is in the subset and 0 otherwise; see, e.g., [40]. The variables belong to the function spaces:

$$u \in L^2(D),\ y \in H_0^1(D)\otimes L^2(\Omega),\ k \in L_+^\infty(D)\otimes L^2(\Omega),\ \beta \in L^\infty(D).$$

The symbol $\otimes$ denotes the tensor product. The subscript $+$ indicates the subset of functions that are positive almost everywhere.

**3.1. Stochastic field $k$.** We assume a *lognormal field* $k(x,\omega) \triangleq \exp(z(x,\omega))$ with $z$ a Gaussian field. We take $\mathbb{E}[z]=0$ and use the common assumption of an exponential covariance [12, 18]

$$C_z(x_1,x_2) = \mathrm{Cov}[z(x_1,\omega), z(x_2,\omega)] = \sigma^2 \exp\Big(-\frac{\|x_1-x_2\|_1}{\lambda}\Big) \tag{5}$$

[1] Rosseel and Wells [36], p. 18, Table 1, first and fourth problems under "unknown mean control."

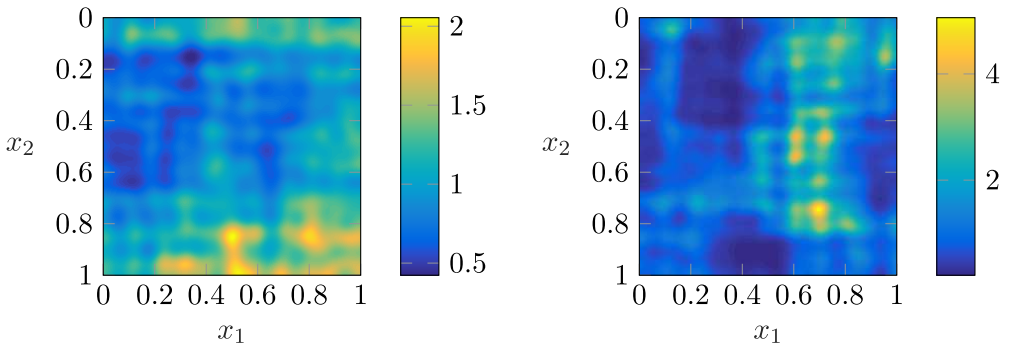



**Figure 1.** *Realizations of the lognormal field $k$ for the 2D case with $\lambda = 0.3$, $n_{KL} = 500$. Left: $\sigma^2 = 0.1$. Right: $\sigma^2 = 0.5$.*

with $\sigma^2$ the variance of the field and $\lambda$ the correlation length. Samples of $z$ can be generated starting from the Karhunen–Loève (KL) expansion [29, 19] of $z$:

$$z(x,\omega) = \mathbb{E}[z(x,\omega)] + \sum_{n=1}^{\infty} \sqrt{\theta_n}\xi_n(\omega) f_n(x); \tag{6}$$

see, e.g., [9, 10, 12, 18, 11]. The KL expansion is the unique expansion of the above form that minimizes the total mean square error (MSE) if the expansion is truncated to a fixed finite number of terms [16]. In this paper we confine ourselves to the choice $\lambda = 0.3$ and choose $n_{\mathrm{KL}} = 500$ terms, capturing 94% of the variance for a two-dimensional (2D) problem. A typical realization for two values of $\sigma$ is found in Figure 1.

The accurate and efficient generation of samples is not the main topic of this paper. Other sampling techniques such as circulant embedding [15, 18] may provide some computational advantages over the KL expansion.

## 4. Optimality conditions.

This section derives the optimality conditions for the model problem. The constraint is denoted here by $c(y,u) = \nabla \cdot (k\nabla y) + \beta u = 0$, without explicit dependence on $k$ or $\omega$. It provides a relation between $y \in H_0^1(D) \otimes L^2(\Omega)$ and $u \in L^2(D)$ for all realizations of $\omega$. All inputs $u$ are assumed to be admissible, i.e., for every $u$, $c(y,u) = 0$ can be uniquely solved for $y$.

### 4.1. General expressions.

The optimality conditions can be derived starting from the Lagrangian

$$\mathcal{L}(y,u,p) = J(y,u) + (p, c(y,u))_{D,\Omega}$$

with $p \in H^1(D) \otimes L^2(\Omega)$ a Lagrange multiplier and $(.,.)_{D,\Omega}$ the standard inner product in $L^2(D) \otimes L^2(\Omega)$. The necessary first order conditions for optimality are then found by setting the partial derivatives to $p$, $y$, and $u$ to zero:

$$\begin{cases} 0 = \nabla_p \mathcal{L} = c(y,u), \\ 0 = \nabla_y \mathcal{L} = \nabla_y J + \left(\dfrac{\partial c}{\partial y}\right)^* [p], \\ 0 = \nabla_u \mathcal{L} = \nabla_u J + \left(\dfrac{\partial c}{\partial u}\right)^* [p]. \end{cases} \tag{7}$$

The superscript $*$ denotes the adjoint of a linear operator. The expression $(\frac{\partial c}{\partial y})^*[p]$, for example, follows through the Riesz representation theorem [34] from

$$\frac{\partial}{\partial y}(p,c)_{D,\Omega}[h] = \left(p, \frac{\partial c}{\partial y}[h]\right)_{D,\Omega} = \left(\left(\frac{\partial c}{\partial y}\right)^*[p], h\right)_{D,\Omega}.$$

**4.2. Reduced gradient for the model problem.** The robust optimization objective can be reformulated in a more compact manner in terms of the norm $\|.\|_{D,\Omega}$ induced by $(.,.)_{D,\Omega}$,

$$\begin{aligned} J(y,u) &= \mathbb{E}[\|y-y_D\|_D^2] + \gamma\|\mathbb{S}[y]\|_D^2 + \alpha\|u\|_D^2 \\ &= \int_\Omega \int_D (y-y_D)^2 \,\mathrm{d}x\,\mathrm{d}\mu(\omega) + \gamma \int_D \int_\Omega (y-\mathbb{E}[y])^2 \,\mathrm{d}\mu(\omega)\,\mathrm{d}x + \alpha \int_D u^2 \,\mathrm{d}x \\ &= \|y-y_D\|_{D,\Omega}^2 + \gamma\|y-\mathbb{E}[y]\|_{D,\Omega}^2 + \alpha\|u\|_D^2. \end{aligned}$$

The terms $\nabla_y J$ and $\nabla_u J$ in (7) can be evaluated by noting that

$$\begin{aligned} \frac{\mathrm{d}}{\mathrm{d}y}\|y-\mathbb{E}[y]\|_{D,\Omega}^2[h] &= \left(2(y-\mathbb{E}[y]), \frac{\mathrm{d}(y-\mathbb{E}[y])}{\mathrm{d}y}[h]\right)_{D,\Omega} \\ &= (2(y-\mathbb{E}[y]), h-\mathbb{E}[h])_{D,\Omega} \\ &= (2(y-\mathbb{E}[y]), h)_{D,\Omega}. \end{aligned} \tag{8}$$

In the last step we used that $(y-\mathbb{E}[y], \mathbb{E}[h])_{D,\Omega} = (\mathbb{E}[y]-\mathbb{E}[y], \mathbb{E}[h])_D = 0$. Hence we find $\nabla_y J = 2(y-y_D) + 2\gamma(y-\mathbb{E}[y])$. Similarly, $\nabla_u J = 2\alpha u$. Since the operator $\nabla\cdot(k\nabla.)$ is linear and self-adjoint, $\nabla_y(p,c) = \nabla\cdot(k\nabla p)$. Finally, for the third equation in (7) we find

$$\frac{\partial}{\partial u}(p, c(y,u))_{D,\Omega}[h] = (p, \beta h)_{D,\Omega} = (\beta p, h)_{D,\Omega}. \tag{9}$$

Since $u \in L^2(D)$, this equality must hold for all increments $h \in L^2(D)$ (as opposed to $L^2(D)\otimes L^2(\Omega)$). Since $(\beta p, h)_{D,\Omega} = (\beta\mathbb{E}[p], h)_D$, we find $\nabla_u(p,c)_{D,\Omega} = \beta\mathbb{E}[p]$. Combining the results, the system of equations (7) reduces to

$$\begin{cases} -\nabla\cdot(k\nabla y) = \beta u & \text{on } D, \\ -\nabla\cdot(k\nabla p) = 2(y-y_D) + 2\gamma(y-\mathbb{E}[y]) & \text{on } D, \\ \nabla\tilde{J}(u) = 2\alpha u + \beta\mathbb{E}[p] = 0. \end{cases} \tag{10}$$

The Dirichlet boundary conditions are omitted for brevity. The last equation provides the so-called reduced gradient, i.e., the gradient of the *reduced cost functional* $\tilde{J}(u) = J(Su, u)$ where $S$ solves the constraint $c$.

*Remark.* The $\mathbb{E}[y]$ term in the second equation of (10) causes a more intricate connection between the values of $y$ and $p$ for the different instances governed by $\omega$. This essentially bars one from deriving the conditions for each $\omega$ separately and joining them in the third equation through an expected value, as is often done in the case $\gamma = 0$.

**4.3. Reduced Hessian for the model problem.** Consider the second order derivative of a functional $f : H \to \mathbb{R}$ and apply some calculus to obtain

$$\frac{\mathrm{d}^2 f(u)}{\mathrm{d}u^2}[h_1, h_2] = \frac{\mathrm{d}}{\mathrm{d}u}\left(\frac{\mathrm{d}f(u)}{\mathrm{d}u}[h_2]\right)[h_1] = \frac{\mathrm{d}}{\mathrm{d}u}\Big((\nabla f(u), h_2)\Big)[h_1] = \left(\frac{\mathrm{d}\nabla f(u)}{\mathrm{d}u}[h_1], h_2\right). \tag{11}$$

The mapping $\frac{\mathrm{d}\nabla f(u)}{\mathrm{d}u}[.], H \to H$ is the Hessian of $f$ in $u$, denoted as Hess $f(u)[.]$ [40]. In the finite dimensional setting, the Hessian can be represented by an ordinary matrix $M$. For any vectors $h_1$ and $h_2$, we have $h_1^T M h_2 = (Mh_1, h_2)_{\mathbb{R}^n}$. The similarity with (11) should be clear. Due to the linearity of the equations in (10), working out $\frac{\mathrm{d}\nabla \tilde{J}(u)}{\mathrm{d}u}[\delta u]$ immediately leads to

$$\begin{cases} -\nabla \cdot (k \nabla\, \delta y) = \beta\, \delta u & \text{on } D, \\ -\nabla \cdot (k \nabla\, \delta p) = 2\, \delta y + 2\gamma(\delta y - \mathbb{E}[\delta y]) & \text{on } D, \\ \text{Hess } \tilde{J}(u)[\delta u] = 2\alpha\, \delta u + \beta \mathbb{E}[\delta p]. \end{cases} \tag{12}$$

The model problem is quadratic since the Hessian is independent of $u$.

**4.4. Discretization.** We assume that the discretization of the equations (10) leads to a system of the form

$$\begin{cases} A\boldsymbol{y} = \beta \boldsymbol{u}, \\ A'\boldsymbol{p} = 2(\boldsymbol{y} - \boldsymbol{y_D}) + 2\gamma(\boldsymbol{y} - \mathbb{E}[\boldsymbol{y}]), \\ \nabla \tilde{J}(\boldsymbol{u}) = 2\alpha \boldsymbol{u} + \beta \mathbb{E}[\boldsymbol{p}] \end{cases} \tag{13}$$

with $A, A' \in \mathbb{R}^{m^d \times m^d}$ dependent on $\omega$. This is the case if, e.g., the finite volume discretization with $m^d$ volumes is used. We use boldface to denote the finite dimensional approximations.

Consider the discretized cost functional

$$J(\boldsymbol{y}, \boldsymbol{u}) = \mathbb{E}[\|\boldsymbol{y} - \boldsymbol{y_D}\|^2] + \gamma \|\mathbb{S}[\boldsymbol{y}]\|^2 + \alpha \|\boldsymbol{u}\|^2 \approx J(y, u), \tag{14}$$

where the norms and inner products over $\mathbb{R}^{m^d}$ are defined as the approximation of their continuous counterparts, i.e.,

$$\|\boldsymbol{v}\|^2 \triangleq \frac{\boldsymbol{v}^T \boldsymbol{v}}{m^d} \approx \|v\|^2 \quad \text{and} \quad (\boldsymbol{u}, \boldsymbol{v}) \triangleq \frac{\boldsymbol{u}^T \boldsymbol{v}}{m^d} \approx (u, v). \tag{15}$$

This definition ensures that the discretized cost functional gives comparable results regardless of the number of discretization points. Using standard differentiation techniques, one can show that the discretized gradient (13) is also the exact gradient of the reduced cost functional $\tilde{J}(\boldsymbol{u}) = J(A^{-1}\beta \boldsymbol{u}, \boldsymbol{u})$ w.r.t. the inner product given in (15). Discretizing the Hessian (12) in the same way is identical to taking the derivative of the discretized gradient, i.e., Hess $\tilde{J}(\boldsymbol{u})[\boldsymbol{\delta u}] = \frac{\mathrm{d}\nabla \tilde{J}(\boldsymbol{u})}{\mathrm{d}\boldsymbol{u}}[\boldsymbol{\delta u}]$.

## 5. Multilevel Monte Carlo.

The evaluation of the reduced gradient (10) or Hessian (12) requires an approximation for $\mathbb{E}[p]$ and, if $\gamma \neq 0$, also for $\mathbb{E}[y]$. Because of the PDE setting, it makes sense to consider an MLMC estimator, which is briefly recalled following the exposition in [12]. Section 5.2 discusses in detail how we handle function valued quantities of interest. Section 6 analyzes the application of the method to the estimation of $\mathbb{E}[p]$ in particular.

**5.1. Scalar-valued quantities of interest.** Assume one wishes to estimate the expected value of some *quantity of interest* (QoI) $Q : \Omega \to \mathbb{R}$. Because of approximation or discretization errors, one often can not generate exact samples $Q(\omega)$ of $Q$. Instead, one can generate samples $Q_m(\omega)$ of an approximation $Q_m$ to $Q$, where $m$ is a measure for the accuracy of the approximation. The numerical scheme is assumed to have a weak order of convergence equal to $\rho$, i.e.,

$$|\mathbb{E}[Q_m - Q]| \lesssim m^{-\rho}. \tag{16}$$

We define $a \lesssim b \Leftrightarrow a \leq cb$ with $c$ independent of $m$ and $\boldsymbol{n}$ below. We write $a \eqsim b$ iff $a \lesssim b$ and $b \lesssim a$. The computational cost $\mathcal{C}(Q_m(\omega))$ for a single sample is assumed to satisfy

$$\mathcal{C}(Q_m(\omega)) \lesssim m^{\kappa} \tag{17}$$

for some constant $\kappa$. Both $\rho$ and $\kappa$ depend on the algorithm employed to solve the PDE.

In the MLMC method [12, 17] one considers multiple approximations $Q_{m_0}, \ldots, Q_{m_L}$ for $Q$. In our setting, $m_\ell = m_0 \cdot 2^\ell$ corresponds to the grid size in a single dimension of the PDE discretization. The coarsest grid size is $m_0$, the finest is $m_L$. The method recursively estimates an expected value on a finer grid as an expected value on a coarser grid (acting as a control variate) combined with a corrective term. This leads to a telescopic sum decomposition

$$\mathbb{E}[Q_{m_L}] = \mathbb{E}[Q_{m_0}] + \sum_{\ell=1}^{L} \mathbb{E}[Q_{m_\ell} - Q_{m_{\ell-1}}] = \sum_{\ell=0}^{L} \mathbb{E}[Y_\ell],$$

where $Y_l \triangleq Q_{m_\ell} - Q_{m_{\ell-1}}$ and $Q_{m_{-1}} \triangleq 0$. On level $\ell$, $\mathbb{E}[Y_\ell]$ is estimated using the ordinary Monte Carlo (MC) method with $n_\ell$ samples, yielding

$$\hat{Y}^{\mathrm{MC}}_{\ell,n_\ell} \triangleq \frac{1}{n_\ell} \sum_{i=1}^{n_\ell} Y_\ell(\omega_i) = \frac{1}{n_\ell} \sum_{i=1}^{n_\ell} \Big( Q_{m_\ell}(\omega_i) - Q_{m_{\ell-1}}(\omega_i) \Big). \tag{18}$$

It is important to use the same stochastic realization $\omega_i$ on both levels for each sample of $Y_\ell$ to ensure a high correlation. The MLMC estimator is then defined as

$$\hat{Q}^{\mathrm{MLMC}}_{\boldsymbol{m},\boldsymbol{n}} \triangleq \sum_{\ell=0}^{L} \hat{Y}^{\mathrm{MC}}_{\ell,n_\ell} \tag{19}$$

with the vector $\boldsymbol{m} = \{m_\ell\}_{\ell=0}^L$ and $\boldsymbol{n} = \{n_\ell\}_{\ell=0}^L$. The linearity of the expected value operator and the fact that all the expectations are estimated independently lead to

$$\mathbb{E}[\hat{Q}^{\mathrm{MLMC}}_{\boldsymbol{m},\boldsymbol{n}}] = \mathbb{E}[Q_{m_L}], \qquad \mathbb{V}[\hat{Q}^{\mathrm{MLMC}}_{\boldsymbol{m},\boldsymbol{n}}] = \sum_{\ell=0}^{L} n_\ell^{-1} \mathbb{V}[Y_\ell]. \tag{20}$$

Moreover, the MSE of $\hat{Q}^{\mathrm{MLMC}}_{\boldsymbol{m},\boldsymbol{n}}$ as an estimator for $\mathbb{E}[Q]$ can be characterized (see [12]) as follows:

$$\begin{aligned} \mathbb{E}\Big[ \big( \hat{Q}^{\mathrm{MLMC}}_{\boldsymbol{m},\boldsymbol{n}} - \mathbb{E}[Q] \big)^2 \Big] &= \mathbb{V}[\hat{Q}^{\mathrm{MLMC}}_{\boldsymbol{m},\boldsymbol{n}}] + \big( \mathbb{E}[\hat{Q}^{\mathrm{MLMC}}_{\boldsymbol{m},\boldsymbol{n}}] - \mathbb{E}[Q] \big)^2 \\ &= \sum_{\ell=0}^{L} n_\ell^{-1} \mathbb{V}[Y_\ell] \;\; + \;\; \mathbb{E}[Q_{m_L} - Q]^2. \end{aligned} \tag{21}$$

The first term is due to the stochastic error, which can be decreased by taking more samples. The second term is due to the discretization error, equal to the bias squared. It can be decreased by solving the PDE on a finer grid, i.e., by increasing $L$. In order to have an RMSE of at most $\epsilon$ it is sufficient if both[2] terms are smaller than $\epsilon^2/2$.

Many possibilities exist for $\boldsymbol{n}$ to achieve a stochastic error smaller than $\epsilon^2/2$. This freedom can be used to minimize the cost of the MLMC estimator. Denote the cost of taking a sample of $Y_\ell$ as $\mathcal{C}_\ell$. The cost of the MLMC estimator is then $\mathcal{C}(\hat{Q}^{\text{MLMC}}_{\boldsymbol{m},\boldsymbol{n}}) = \sum_{\ell=0}^{L} n_\ell \mathcal{C}_\ell$. Minimizing this cost subject to the constraint $\sum_{\ell=0}^{L} n_\ell^{-1}\mathbb{V}[Y_\ell] = \epsilon^2/2$ yields an optimization problem which is easily solved using Lagrange multipliers. The solution, rounded upward, yields the optimal number of samples

$$n_\ell = \left\lceil \frac{2}{\epsilon^2}\sqrt{\mathbb{V}[Y_\ell]\mathcal{C}_\ell^{-1}} \sum_{i=0}^{L} \sqrt{\mathbb{V}[Y_i]\mathcal{C}_i} \right\rceil. \tag{22}$$

Substituting (22), before rounding upward, into the expression for the cost yields

$$\mathcal{C}(\hat{Q}^{\text{MLMC}}_{\boldsymbol{m},\boldsymbol{n}}) = \frac{2}{\epsilon^2}\left( \sum_{\ell=0}^{L} \sqrt{\mathbb{V}[Y_\ell]\mathcal{C}_\ell} \right)^2.$$

If $\mathbb{V}[Y_\ell]$ decreases faster than $\mathcal{C}_\ell$ increases with increasing $\ell$, the dominant cost is on the coarsest level $\ell = 0$ and is proportional to $\mathbb{V}[Y_0]\mathcal{C}_0$. The cost savings compared to the standard MC method are then proportional to $\mathcal{C}_0/\mathcal{C}_L \approx (m_0/m_L)^\kappa \approx \epsilon^{\kappa/\rho}$. If the converse is true, then the dominant cost is on the finest level $L$ and proportional to $\mathbb{V}[Y_L]\mathcal{C}_L$. The cost savings are then approximately $\mathbb{V}[Y_L]/\mathbb{V}[Y_0]$.

*Remark.* Note that $m_0$ cannot be made arbitrarily small. If the discretization is too coarse, the relevant features of the PDE solution can no longer be resolved. The resemblance between the coarse and the fine level solution will then be lost, i.e., $\mathbb{V}[Y_1]$ will no longer be smaller than $\mathbb{V}[Q_{m_1}]$. It is then cheaper to estimate $\mathbb{E}[Q_{m_1}]$ directly, which is equivalent to increasing $m_0$.

Collecting all of the assumptions and quantifying the decay of $\mathbb{V}[Y_\ell]$ yields the MLMC cost theorem as presented and proven in [12]:

Theorem 5.1 (multilevel Monte Carlo cost). *Suppose that there are positive constants $\rho, \phi$, $\kappa > 0$ such that $\rho > \frac{1}{2}\min(\phi, \kappa)$ and*

$$|\mathbb{E}[Q_{m_\ell} - Q]| \lesssim m_\ell^{-\rho}, \qquad \mathbb{V}[Y_\ell] \lesssim m_\ell^{-\phi}, \qquad \mathcal{C}_\ell \lesssim m_\ell^{\kappa}.$$

*Then, for any $\epsilon < e^{-1}$, there exist a value $L$ and a sequence $\boldsymbol{n} = \{n_\ell\}_{\ell=0}^{L}$ such that the MSE*

$$\mathbb{E}\left[\left(\hat{Q}^{\text{MLMC}}_{\boldsymbol{m},\boldsymbol{n}} - \mathbb{E}[Q]\right)^2\right] < \epsilon^2$$

*and the cost*

$$\mathcal{C}(\hat{Q}^{\text{MLMC}}_{\boldsymbol{m},\boldsymbol{n}}) \lesssim \begin{cases} \epsilon^{-2} & \text{if } \phi > \kappa, \\ \epsilon^{-2}(\log\epsilon)^2 & \text{if } \phi = \kappa, \\ \epsilon^{-2-(\kappa-\phi)/\rho} & \text{if } \phi < \kappa. \end{cases} \tag{23}$$

[2] In practice, often a larger part of the RMSE is allocated to the stochastic error.

In practice, the problem dependent parameters $\rho, \phi$, and $\kappa$ are not always known in advance and may have to be estimated. Furthermore, $L$ has to be selected carefully in order to have a sufficiently small bias term.

**5.2. Function valued quantities of interest.** The main quantities of interest in this paper are the gradient and the Hessian vector product, which in our application are functions instead of scalar values. These functions are discretized on the different levels and have to be combined in the course of the estimation algorithm. Hence, it is necessary to define a mapping between those discretizations.

**5.2.1. Mapping between different levels.** Consider a linear transform $I_{\ell_1}^{\ell_2} : \mathbb{R}^{m_{\ell_1}^d} \to \mathbb{R}^{m_{\ell_2}^d} : \boldsymbol{v} \mapsto I_{\ell_1}^{\ell_2}\boldsymbol{v}$ that maps vectors from level $\ell_1$ to level $\ell_2$. The operator is a *prolongation* if $\ell_1 < \ell_2$, a *restriction* if $\ell_1 > \ell_2$, and the identity if $\ell_1 = \ell_2$. For any $\ell_1, \ell_2 \in \mathbb{N}$ with $\ell_1 < \ell_2$, we shall require that

$$I_{\ell_1}^{\ell_2} = I_{\ell_2-1}^{\ell_2} I_{\ell_2-2}^{\ell_2-1} \dots I_{\ell_1}^{\ell_1+1} \quad \text{and} \quad I_{\ell_1}^{\ell_2} = c^{\ell_2-\ell_1}(I_{\ell_2}^{\ell_1})^T \tag{24}$$

for some constant $c$. In our case, $c = 2^d$. An analogous expression should hold if $\ell_1 > \ell_2$. The precise definition of $I_{\ell_1}^{\ell_2}$ depends on the discretization method and the selection of the mesh points at the different levels. The ideas in this paper do not depend on a specific method used to solve PDE (4) for a single realization $\omega$. Here the *finite volume method* will be used to obtain function values at *control volume* centers. None of the nodes existing at a level $\ell$ are then present at the level $\ell + 1$; see Figure 2. The prolongation operator is often chosen to interpolate linearly. In our situation, the effect is a smoothing of the function when mapped from one level to the next. Note that MLMC works best if the results on consecutive levels match as closely as possible. Alternative definitions for $I_{\ell_1}^{\ell_2}$ are of course possible. However, choosing a poor definition results in a slower decay of $\mathbb{V}[\boldsymbol{Y}_\ell]$ since it can cause unnecessary dissimilarity between $\boldsymbol{Q}_{m_\ell}$ and $I_{\ell-1}^{\ell}\boldsymbol{Q}_{m_{\ell-1}}$.

**5.2.2. Revised algorithm and bias estimation.** We can now amend (18) and (19) as follows:

$$\hat{\boldsymbol{Y}}_{\ell,n_\ell}^{\mathrm{MC}} = \frac{1}{n_\ell}\sum_{i=1}^{n_\ell}\left(\boldsymbol{Q}_{m_\ell}(\omega_i) - I_{\ell-1}^{\ell}\boldsymbol{Q}_{m_{\ell-1}}(\omega_i)\right) \quad \text{and} \quad \hat{\boldsymbol{Q}}_{\boldsymbol{m},\boldsymbol{n}}^{\mathrm{MLMC}} \triangleq \sum_{\ell=0}^{L} I_{\ell}^{\bar{L}}\hat{\boldsymbol{Y}}_{\ell,n_\ell}^{\mathrm{MC}}. \tag{25}$$

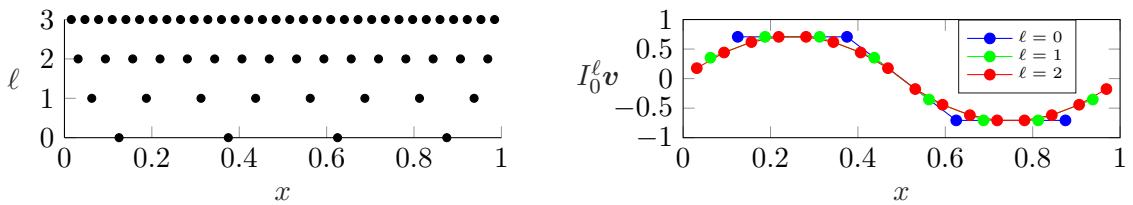



**Figure 2.** *Left: Discretization node pattern corresponding to the control volume centers for a 1D problem using the finite volume method. None of the nodes existing at a certain level $\ell$ are present at another level. Right: Interpolating some $\boldsymbol{v}$ to finer grids smooths it.*

In this paper, the estimated functions are always returned discretized at some fixed level $\bar{L}$. If a sufficiently small RMSE is reached for some $L < \bar{L}$, no samples are taken at levels $\ell > L$. Returning the result at a fixed predetermined level simplifies the implementation of the optimization algorithm. Optimization software usually requires the gradient and Hessian in a format that is not allowed to change from iteration to iteration.

Some methods to extend the MLMC theory to vector or function valued QoI can be found in [17, pp. 274–276]. Let $I^\ell$ be the discretization operator, which samples a function in the discretization nodes of level $\ell$. In this paper, we demand the MSE (21) to be smaller than $\epsilon^2$ for each point on the return level $\bar{L}$, i.e., we demand

$$\mathbb{E}[(\hat{\boldsymbol{Q}}^{\mathrm{MLMC}}_{\boldsymbol{m},\boldsymbol{n}} - I^{\bar{L}}\mathbb{E}[Q])^2] \leq \epsilon^2. \tag{26}$$

Much of the MLMC theory for scalar valued quantities can then be reused without much modification. Some technical details about how we attempted to satisfy (26) now follow.

Evaluating the variance of $\hat{\boldsymbol{Q}}^{\mathrm{MLMC}}_{\boldsymbol{m},\boldsymbol{n}}$ from its definition (25) shows that the relevant series of variances necessary to evaluate (22) for all discretization points is given by $\{\mathbb{V}[I^{\bar{L}}_\ell \boldsymbol{Y}_\ell]\}_{\ell=0}^L$. These can be approximated as $\{I^{\bar{L}}_\ell \mathbb{V}[\boldsymbol{Y}_\ell]\}_{\ell=0}^L$, where $\mathbb{V}[\boldsymbol{Y}_\ell]$ are estimated using some warm-up samples. The optimal number of samples is determined for each domain point separately. Then, the maximum over all the domain points is taken as $\boldsymbol{n}$. This ensures that $\sum_{\ell=0}^L n_\ell^{-1} I^{\bar{L}}_\ell \mathbb{V}[\boldsymbol{Y}_\ell] \leq \epsilon^2/2$ in all points of the domain.

For the bias we start with an estimation of $\rho$ in (16). Assume there exist $c, c' \in \mathbb{R}$ such that from a certain level $\ell$ onward

$$\|\mathbb{E}[\boldsymbol{Q}_{m_\ell} - I^\ell Q]\|_\infty \approx c m_\ell^{-\rho} = c 2^{-\rho\ell} \quad \text{and} \quad \|\mathbb{E}[\boldsymbol{Q}_{m_\ell} - I^\ell_{\ell-1}\boldsymbol{Q}_{m_{\ell-1}}]\|_\infty \approx c' 2^{-\rho\ell}.$$

These assumptions were found experimentally to hold best, especially for low $\ell$, when using the inf-norm. Fitting a line to $\log_2 \|\mathbb{E}[\boldsymbol{Q}_{m_\ell} - I^\ell_{\ell-1}\boldsymbol{Q}_{m_{\ell-1}}]\|_\infty \approx \log_2 c' - \rho\ell$ and getting the first degree coefficient then provides an estimation of $\rho$. The reverse triangle inequality yields

$$\begin{aligned}\|\mathbb{E}[\boldsymbol{Q}_{m_\ell} - I^\ell_{\ell-1}\boldsymbol{Q}_{m_{\ell-1}}]\|_\infty &\geq \|\mathbb{E}[I^\ell_{\ell-1}\boldsymbol{Q}_{m_{\ell-1}} - I^\ell Q]\|_\infty - \|\mathbb{E}[\boldsymbol{Q}_{m_\ell} - I^\ell Q]\|_\infty \\ &\approx (2^\rho - 1)\|\mathbb{E}[\boldsymbol{Q}_{m_\ell} - I^\ell Q]\|_\infty,\end{aligned}$$

leading to the following bound for the largest bias over the domain:

$$\|\mathbb{E}[\boldsymbol{Q}_{m_\ell} - I^\ell Q]\|_\infty \leq (2^\rho - 1)^{-1}\|\mathbb{E}[\boldsymbol{Q}_{m_\ell} - I^\ell_{\ell-1}\boldsymbol{Q}_{m_{\ell-1}}]\|_\infty. \tag{27}$$

Since the necessary number of levels $L$ is not a priori known, the algorithm starts out with only a few levels and checks

$$\left\| \sum_{\ell=0}^{L} n_\ell^{-1} I^{\bar{L}}_\ell \mathbb{V}[\boldsymbol{Y}_\ell] \right\|_\infty + \|\mathbb{E}[\boldsymbol{Q}_{m_L} - I^L Q]\|^2_\infty \leq \epsilon^2 \tag{28}$$

with the second term estimated through (27). This is a somewhat overly conservative test for (26). The above equation holds for all domain points if it holds for the worst case point, hence the inf-norm over the first term. It is also sufficient to simply replace the first term by $\epsilon^2/2$.

**Algorithm 1.** Multilevel Monte Carlo estimation of function valued quantities

$L \leftarrow 0$, *converged* $\leftarrow$ false
**while** not *converged* and $L \leq \bar{L}$ **do**
  take an amount $n_{\text{init}}$ of initial samples at level $L$
  estimate $\mathbb{V}[\boldsymbol{Y}_L]$ from these samples
  calculate the optimal number of samples $\boldsymbol{n} = \{n_\ell\}_{\ell=0}^{L}$ following subsection 5.2.2
  take more samples on levels $0, \dots, L$ until the total number taken is at least $\boldsymbol{n}$
  **if** $L \geq 1$ **then**
    estimate $\rho$ and the bias following subsection 5.2.2
    *converged* $\leftarrow$ check (28) at level $L$
  **end if**
  $L \leftarrow L + 1$
**end while**
return $\hat{\boldsymbol{Q}}^{\text{MLMC}}_{\boldsymbol{m},\boldsymbol{n}}$, following (25)

If the resulting requirement for the bias is not satisfied, an additional level is added. An overestimation of either term would cause the algorithm to consider an additional unnecessary level. Note that this is not too bad if the dominant cost is on the coarsest grid, as is the case for all experiments in this paper. This provides another justification for the use of the conservative inf-norm in the bias estimation. The full MLMC algorithm is given in Algorithm 1.

## 6. Estimator for the gradient.

We turn now to the specific problem of finding an estimate for $\mathbb{E}[\boldsymbol{p}]$ and thus for the gradient in the optimality conditions (13). Estimating $\mathbb{E}[\boldsymbol{p}]$ seems to require an estimation of $\mathbb{E}[\boldsymbol{y}]$ first. This leads to two problems. First, it is assumed that the available computer memory is too small to save all the samples used to estimate $\mathbb{E}[\boldsymbol{y}]$. Any such sample is thus lost unless it is recalculated later, thereby increasing calculation cost. Second, it is unclear which MSE would have to be requested for $\mathbb{E}[\boldsymbol{y}]$. So, we want to get rid of the need to estimate $\mathbb{E}[\boldsymbol{y}]$ in advance. Moreover, we want to retain the property that the calculated gradient is *exact*, meaning that it is the exact gradient of some cost function.

### 6.1. Generating samples of $p$ directly.

The $\mathbb{E}[\boldsymbol{y}]$ term in (13) stems from $\nabla_{\boldsymbol{y}} \gamma \|\mathbb{S}[\boldsymbol{y}]\|^2 = 2\gamma(\boldsymbol{y} - \mathbb{E}[\boldsymbol{y}])$, where the gradient is expressed w.r.t. the inner product

$$(\boldsymbol{u}, \boldsymbol{v})_{D,\Omega} = \int_\Omega \frac{\boldsymbol{u}^T \boldsymbol{v}}{m^d} \, \mathrm{d}\mu(\omega) \approx (u, v)_{D,\Omega}. \tag{29}$$

This holds for any stochastic space, in particular also for a finite subset of samples $\Omega_0 = \{\omega_1, \dots, \omega_n\} \subset \Omega$, each having equal probability. For any such set $\Omega_0$, (29) reduces to

$$(\boldsymbol{u}, \boldsymbol{v})_{D,\Omega_0} = \frac{1}{n} \sum_{\omega \in \Omega_0} \frac{\boldsymbol{u}^T \boldsymbol{v}}{m^d}. \tag{30}$$

Writing $\boldsymbol{y}(\omega_i) = \boldsymbol{y}_i$, the following gradient w.r.t. $(.,.)_{D,\Omega_0}$ is therefore equal to

$$\nabla_{\boldsymbol{y}} \left\| \sqrt{\frac{1}{n} \sum_{j=1}^{n} \left( \boldsymbol{y}_j - \frac{1}{n} \sum_{i=1}^{n} \boldsymbol{y}_i \right)^2} \right\|^2 = 2 \left( \boldsymbol{y} - \frac{1}{n} \sum_{i=1}^{n} \boldsymbol{y}_i \right). \tag{31}$$

Hence, if $\mathbb{E}[\boldsymbol{y}]$ in (13) is estimated by means of $n$ MC samples, the standard deviation term in $J(\boldsymbol{y}, \boldsymbol{u})$ is to be evaluated as suggested by the left-hand side in (31), i.e., by using the standard (biased) sample variance. Any alternative way to estimate $\mathbb{E}[\boldsymbol{y}]$ entails a corresponding change in the estimator for the variance in the cost functional and vice versa. Consider another estimator based on two sets of $n$ samples each,

$$\hat{V}[\boldsymbol{y}] \triangleq \frac{1}{2n}\sum_{j=1}^{n}(\boldsymbol{y}_j - \boldsymbol{y}'_j)^2, \tag{32}$$

with $\boldsymbol{y}'_j = \boldsymbol{y}(\omega'_j)$ and $\Omega'_0 = \{\omega'_1, \ldots, \omega'_n\} \subset \Omega$ a second set of samples. All of the $2n$ samples are independent. Due to the independence of $\omega_j$ and $\omega'_j$ in particular, it is an unbiased estimator for the variance since

$$\begin{aligned}\mathbb{E}[(\boldsymbol{y}_j - \boldsymbol{y}'_j)^2] &= \mathbb{E}[(\boldsymbol{y}_j - \mathbb{E}[\boldsymbol{y}] + \mathbb{E}[\boldsymbol{y}] - \boldsymbol{y}'_j)^2] \\ &= \mathbb{E}[(\boldsymbol{y}_j - \mathbb{E}[\boldsymbol{y}])^2] + \mathbb{E}[(\mathbb{E}[\boldsymbol{y}] - \boldsymbol{y}'_j)^2] \\ &= 2\mathbb{V}[\boldsymbol{y}].\end{aligned}$$

Note that the MSE $\mathbb{V}[\hat{V}[\boldsymbol{y}]]$ is somewhat less favorable than that of the standard biased estimator, i.e., more samples are needed for an accurate estimate of $\mathbb{V}[\boldsymbol{y}]$. This can be demonstrated as follows. Assuming that $\boldsymbol{y}_j$ and $\boldsymbol{y}'_j$ are Gaussian with variance $\sigma^2$, it can be shown that $\mathbb{V}[(\boldsymbol{y}_j - \boldsymbol{y}'_j)^2] = 8\sigma^4$ and thus $\mathbb{V}[\hat{V}[\boldsymbol{y}]] = \frac{2\sigma^4}{n}$. The standard biased variance estimator uses only $n$ realizations of $\boldsymbol{y}$ and has the same variance. However, since it leads to a gradient containing the term $\mathbb{E}[\boldsymbol{y}]$, which has to be estimated in advance with additional samples of $\boldsymbol{y}$, its implied computational cost is not lower in practice. In contrast, the gradient corresponding to $\hat{V}[\boldsymbol{y}]$ w.r.t. $(.,.)_{D,\Omega_0}$ is

$$\nabla_{\boldsymbol{y}}\left\|\sqrt{\hat{V}[\boldsymbol{y}]}\right\|^2 = \boldsymbol{y} - \boldsymbol{y}'. \tag{33}$$

Using this expression in (13) to replace $2(\boldsymbol{y} - \mathbb{E}[\boldsymbol{y}])$ yields the gradient of (14) where $\mathbb{S}[\boldsymbol{y}]$ is estimated by $(\hat{V}[\boldsymbol{y}])^{1/2}$ and $\mathbb{E}[.]$ by the average over $\Omega_0$. The $j$th sample of $\boldsymbol{p}$ at some given level then requires two solves of the state equation to obtain $\boldsymbol{y}_j$ and $\boldsymbol{y}'_j$ and a single solve of the adjoint equation. Because each sample $\boldsymbol{p}_j$ depends on distinct samples $\omega_j$ and $\omega'_j$, all of $\{\boldsymbol{p}_j\}_{j=1}^n$ are clearly independent, as is required in the MLMC method.

*Remark.* It is in principle possible to improve the resulting gradient estimator by using $\{\boldsymbol{y}'_j\}_{j=1}^n$ to generate additional samples $\{\boldsymbol{p}'_j\}_{j=1}^n$ of $\boldsymbol{p}$ that in effect correspond with $\{\omega'_j\}_{j=1}^n$, at the cost of $n$ additional adjoint equation solves. However, $\boldsymbol{p}_j$ and $\boldsymbol{p}'_j$ are not independent! If compromising on the independence of samples is allowed, many other methods can be constructed. The next section provides such a method that is very similar but easier to analyze and generalize.

**6.2. Generating cheaper samples of $p$ directly.** Consider yet another estimator for the variance based only on a single set of $n$ independent samples $\Omega_0$:

$$\hat{V}_1[\boldsymbol{y}] \triangleq \frac{1}{2n}\sum_{j=1}^{n}(\boldsymbol{y}_j - \boldsymbol{y}_{j-1})^2, \tag{34}$$

where $\boldsymbol{y}_0 = \boldsymbol{y}_n$. Since the samples $\Omega_0$ are independent, we have again $\mathbb{E}[(\boldsymbol{y}_j - \boldsymbol{y}_{j-1})^2] = 2\mathbb{V}[\boldsymbol{y}]$, making $\hat{V}_1[\boldsymbol{y}]$ an unbiased estimator for $\mathbb{V}[\boldsymbol{y}]$. The gradient w.r.t. $(.,.)_{D,\Omega_0}$ is worked out explicitly in Appendix A, yielding

$$\nabla_{\boldsymbol{y}} \left\| \sqrt{\hat{V}_1[\boldsymbol{y}]} \right\|^2 = 2\boldsymbol{y} - \boldsymbol{y}_{\oplus 1} - \boldsymbol{y}_{\ominus 1}, \tag{35}$$

where $\boldsymbol{y}_{\oplus i}$ denotes the stochastic variable $\boldsymbol{y}$ "shifted" by $i$ samples in the sampled stochastic space: $\boldsymbol{y}_{\oplus i}(\omega_j) \triangleq \boldsymbol{y}(\omega_{j+i})$ with $\omega_{n+i} = \omega_i$, and $\boldsymbol{y}_{\ominus i} \triangleq \boldsymbol{y}_{\oplus -i}$. This definition only makes sense for a given ordered finite subset of $\Omega$. A single sample of (35), e.g., the sample corresponding to $\omega_j$, is then $2\boldsymbol{y}_j - \boldsymbol{y}_{j+1} - \boldsymbol{y}_{j-1}$. Calculating multiple samples in succession then merely requires saving the previous sample, $\boldsymbol{y}_{j-1}$, and obtain the next sample, $\boldsymbol{y}_{j+1}$, early.

One of the disadvantages is the somewhat higher variance of the estimator:

$$\mathbb{V}[\hat{V}_1[\boldsymbol{y}]] = \frac{1}{4n^2}\sum_{j=1}^{n} \mathbb{V}[(\boldsymbol{y}_j - \boldsymbol{y}_{j-1})^2] + \frac{1}{2n^2}\sum_{j=1}^{n} \mathrm{Cov}[(\boldsymbol{y}_j - \boldsymbol{y}_{j-1})^2, (\boldsymbol{y}_{j+1} - \boldsymbol{y}_j)^2].$$

Under the assumption that $\boldsymbol{y}_j$ are Gaussian variables with variance $\sigma^2$, one has that

$$\mathbb{V}[(\boldsymbol{y}_j - \boldsymbol{y}_{j-1})^2] = 8\sigma^4 \quad \text{and} \quad \mathrm{Cov}[(\boldsymbol{y}_j - \boldsymbol{y}_{j-1})^2, (\boldsymbol{y}_{j+1} - \boldsymbol{y}_j)^2] = 2\sigma^4$$

and therefore

$$\mathbb{V}[\hat{V}_1[\boldsymbol{y}]] = \frac{2\sigma^4}{n} + \frac{\sigma^4}{n} = \frac{3\sigma^4}{n}.$$

This is to be compared to the Cramér–Rao lower bound for unbiased estimators of the variance, which is $\frac{2\sigma^4}{n}$ for Gaussian variables.

Taking dependent samples requires some changes in the classical MC and MLMC theory. For simplicity, the description is, as before, given for a scalar valued QoI. We consider only dependencies between samples of $Y_\ell$ taken on the same level $\ell$. From (19) we then have

$$\mathbb{V}[\hat{Q}^{\mathrm{MLMC}}_{\boldsymbol{m},\boldsymbol{n}}] = \sum_{\ell=0}^{L} \mathbb{V}[\hat{Y}^{\mathrm{MC}}_{\ell,n_\ell}] = \sum_{\ell=0}^{L} n_\ell^{-2} \sum_{i=1}^{n_\ell}\sum_{j=1}^{n_\ell} \mathrm{Cov}[Y_{\ell,i}, Y_{\ell,j}],$$

where $Y_{\ell,i}$, the $i$th sample of $Y_\ell$, is interpreted as a random variable. If the covariance matrix is circulant and if $\mathrm{Cov}[Y_{\ell,i}, Y_{\ell,j}] = 0$ for $b < |i-j| < n_\ell - b$, we get

$$\mathbb{V}[\hat{Q}^{\mathrm{MLMC}}_{\boldsymbol{m},\boldsymbol{n}}] = \sum_{\ell=0}^{L} n_\ell^{-1} \left( \mathbb{V}[Y_\ell] + 2\sum_{j=2}^{b+1} \mathrm{Cov}[Y_{\ell,1}, Y_{\ell,j}] \right). \tag{36}$$

For independent samples, $b = 0$ and this equation reduces to (20). For the sampling method associated with $\hat{V}_1$ above, we have $b = 2$. It is then necessary to estimate the $b = 2$ covariances in addition to the sample variance in line 5 of Algorithm 1. In (22) and in Theorem 5.1, $\mathbb{V}[Y_\ell]$ is then replaced by $\mathbb{V}[Y_\ell] + 2\sum_{j=2}^{b+1} \mathrm{Cov}[Y_{\ell,1}, Y_{\ell,j}]$. Note that since the covariances can also be negative, they can actually reduce the amount of samples required! Because the variances and covariances are estimated by a small number of samples, especially at the finer levels, the risk of underestimating the latter quantity is mitigated by replacing $\mathbb{V}[Y_\ell]$ with

$\max\{\frac{1}{2}\mathbb{V}[Y_\ell], \mathbb{V}[Y_\ell] + 2\sum_{j=2}^{b+1}\mathrm{Cov}[Y_{\ell,1}, Y_{\ell,j}]\}$ instead. The $\frac{1}{2}$ term was chosen rather arbitrarily and can probably be improved, depending on the precision of the estimators. $\hat{V}_1$ is used in the remainder of this text.

**6.3. Exactness of the MLMC generated gradient.** Assume one uses the MLMC method to calculate the gradient in a given point $\boldsymbol{u}$ defined on the finest level. This requires the method to take samples $\boldsymbol{Q}_{m_\ell}(\boldsymbol{v}, \omega, \Omega_k)$ which depend on $\boldsymbol{v} = I_{\bar{L}}^{\ell}\boldsymbol{u}$. Following the previous subsections, each sample may depend on multiple elements in the given ordered sample set $\Omega_k$. The variable $\omega \in \Omega_k$ is simply used to index the samples.

Theorem 6.1 (exactness of the MLMC gradient). *Assume that for any level $\ell$ and any sample set $\Omega_k \subset \Omega$ of size $n_k$, the mapping $\mathbb{R}^{m_\ell^d} \to \mathbb{R}^{m_\ell^d} : \boldsymbol{v} \mapsto n_k^{-1}\sum_{\omega\in\Omega_k}\boldsymbol{Q}_{m_\ell}(\boldsymbol{v}, \omega, \Omega_k)$ forms the exact gradient of some cost function. Then, the MLMC method that uses on each level a combination of those sample sets describes a mapping $\mathbb{R}^{m_{\bar{L}}^d} \to \mathbb{R}^{m_{\bar{L}}^d} : \boldsymbol{u} \mapsto \nabla\hat{J}(\boldsymbol{u})$ which is itself the exact gradient of some cost function $\hat{J}$.*

*Proof.* Let $\tilde{J}_\ell^k : \mathbb{R}^{m_\ell^d} \to \mathbb{R}$ denote the cost function corresponding to the exact gradient $n_k^{-1}\sum_{\omega\in\Omega_k}\boldsymbol{Q}_{m_\ell}(\boldsymbol{v}, \omega, \Omega_k) = \nabla\tilde{J}_\ell^k(\boldsymbol{v}) \in \mathbb{R}^{m_\ell^d}$. Consider $\boldsymbol{u} \in \mathbb{R}^{m_{\bar{L}}^d}$ given at the level $\bar{L}$. The chain rule and the second equation of (24) yield

$$\frac{\mathrm{d}}{\mathrm{d}\boldsymbol{u}}\tilde{J}_\ell^k(I_{\bar{L}}^{\ell}\boldsymbol{u})[\boldsymbol{h}] = (\nabla\tilde{J}_\ell^k(I_{\bar{L}}^{\ell}\boldsymbol{u}), I_{\bar{L}}^{\ell}\boldsymbol{h}) = \frac{1}{m_\ell^d}\nabla\tilde{J}_\ell^k(I_{\bar{L}}^{\ell}\boldsymbol{u})^T I_{\bar{L}}^{\ell}\boldsymbol{h} = \frac{c^\ell m_{\bar{L}}^d}{c^{\bar{L}} m_\ell^d}(I_\ell^{\bar{L}}\nabla\tilde{J}_\ell^k(I_{\bar{L}}^{\ell}\boldsymbol{u}), \boldsymbol{h}). \tag{37}$$

Note that the two instances of the inner product (15) are different since they use a different number of discretization points. Let $\nabla\hat{J}(\boldsymbol{u})$ denote the gradient approximation generated by the MLMC algorithm, assumed to have converged on a level $L$. Denote the set of $n_\ell$ samples taken by the algorithm at any level $\ell$ by $\Omega_\ell \subset \Omega$. From (25) we have

$$\begin{aligned}\nabla\hat{J}(\boldsymbol{u}) &= \sum_{\ell=0}^{L} I_\ell^{\bar{L}}\frac{1}{n_\ell}\sum_{\omega\in\Omega_\ell}\left(\boldsymbol{Q}_{m_\ell}(I_{\bar{L}}^{\ell}\boldsymbol{u}, \omega, \Omega_\ell) - I_{\ell-1}^{\ell}\boldsymbol{Q}_{m_{\ell-1}}(I_{\bar{L}}^{\ell-1}\boldsymbol{u}, \omega, \Omega_\ell)\right)\\ &= I_0^{\bar{L}}\nabla\tilde{J}_0^0(I_{\bar{L}}^{0}\boldsymbol{u}) + \sum_{\ell=1}^{L} I_\ell^{\bar{L}}\left(\nabla\tilde{J}_\ell^\ell(I_{\bar{L}}^{\ell}\boldsymbol{u}) - I_{\ell-1}^{\ell}\nabla\tilde{J}_{\ell-1}^\ell(I_{\bar{L}}^{\ell-1}\boldsymbol{u})\right).\end{aligned}$$

This calculated gradient is the exact gradient of the cost functional

$$\hat{J}(\boldsymbol{u}) = \frac{c^{\bar{L}}m_0^d}{c^0 m_{\bar{L}}^d}\tilde{J}_0^0(I_{\bar{L}}^{0}\boldsymbol{u}) + \sum_{\ell=1}^{L}\left(\frac{c^{\bar{L}}m_\ell^d}{c^\ell m_{\bar{L}}^d}\tilde{J}_\ell^\ell(I_{\bar{L}}^{\ell}\boldsymbol{u}) - \frac{c^{\bar{L}}m_{\ell-1}^d}{c^{\ell-1}m_{\bar{L}}^d}\tilde{J}_{\ell-1}^\ell(I_{\bar{L}}^{\ell-1}\boldsymbol{u})\right) \tag{38}$$

as can be checked using (37). ∎

In a similar fashion, it can be proven that the Hessian vector product calculated using MLMC is exact for some cost functional. Note that in this paper, the constant $c$ from (24) satisfies $c = 2^d$ and $m_\ell^d = 2^{d\ell}m_0^d$ such that $c^\ell m_{\bar{L}}^d(c^{\bar{L}}m_\ell^d)^{-1} = 1$ in (37). This is also true for many other common grid definitions and mapping operators. The cost functional (38) then simplifies to

$$\hat{J}(\boldsymbol{u}) = \tilde{J}_0^0(I_{\bar{L}}^{0}\boldsymbol{u}) + \sum_{\ell=1}^{L}\left(\tilde{J}_\ell^\ell(I_{\bar{L}}^{\ell}\boldsymbol{u}) - \tilde{J}_{\ell-1}^\ell(I_{\bar{L}}^{\ell-1}\boldsymbol{u})\right), \tag{39}$$

which corresponds to the cost functional calculated using MLMC. The reason to construct the MLMC estimator for the gradient directly is that the optimization requires the gradient with some known precision, as measured by the RMSE. For some given number of samples $\boldsymbol{n}$, the RMSE on the cost functional estimator is, in general, completely different from the RMSE on the gradient estimator. Finally, we remark that (38) implies that MLMC retains convexity properties. In Theorem 6.1, the term "cost function" may be replaced everywhere by "convex cost function," since the convexity of all $J_\ell^k$ implies convexity of $\hat{J}$. Strict convexity is retained only if $L = \bar{L}$.

## 7. Numerical optimization.

We follow the *reduced* optimization approach, in which the state $y$ is eliminated. This results in $u$ being the only unknown in the optimization problem. The alternative is the *simultaneous* approach in which the original constrained optimization problem is solved directly [6]. However, in the stochastic case, one has that $y \in H_0^1(D) \otimes L^2(\Omega)$. Assuming an MLMC approach, the full sample set of discretized state functions on all levels would have to be part of the variables that one optimizes for. This approach seems infeasible due to excessive memory demands.

In this section, we elaborate on two methods to solve our optimization problem up to a given gradient tolerance. First, the use of the NCG method is investigated. Specific attention is given to how many samples and which samples should be used. Next, the Newton method is studied, which requires Hessian information. Because of its unwieldy size in the problems we consider, the Hessians are never computed explicitly and the linear systems in the Newton iterations are solved using a matrix vector product based implementation of the conjugate gradient (CG) method . For quadratic problems, the NCG method and a single Newton step with CG are known to be equivalent [32]. The difference in this stochastic context will lie mainly in the times at which the samples are updated.

### 7.1. Gradient based optimization.

The use of MLMC in a gradient based optimization algorithm is tested using the NCG method. Variables at the $k$th iteration are indicated by a $(k)$ superscript. The gradient calculated at iteration $k$ is denoted by $\boldsymbol{g}^{(k)}$. In the NCG method, the search direction $\boldsymbol{d}^{(k)}$ is obtained recursively as $\boldsymbol{d}^{(k)} = -\boldsymbol{g}^{(k)} + \beta^{(k)}\boldsymbol{d}^{(k-1)}$ with $\boldsymbol{d}^{(0)} = -\boldsymbol{g}^{(0)}$. We use the Dai–Yuan (DY) formula

$$\beta^{(k)} = \frac{||\boldsymbol{g}^{(k)}||^2}{(\boldsymbol{d}^{(k-1)}, \boldsymbol{g}^{(k)} - \boldsymbol{g}^{(k-1)})},$$

which offers certain advantages in PDE constrained optimization [7, 13]. The system input is then updated as $\boldsymbol{u}^{(k+1)} = \boldsymbol{u}^{(k)} + s^{(k)}\boldsymbol{d}^{(k)}$. The step size $s^{(k)}$ is found by approximating the cost function along the search direction with an interpolating parabola using a second gradient evaluation (we use the point $\boldsymbol{u}^{(k)} + s^{(k-1)}\boldsymbol{d}^{(k)}$ with $s^{(-1)}$ some well-chosen initial value). Note that this approximate linesearch is exact for a quadratic problem.

#### 7.1.1. Choosing samples.

In theory, the gradient and the Hessian vector product are deterministic quantities due to the expected value operators in their equations. Computationally, however, the result depends on the specific samples drawn by the MLMC algorithm. These samples can be saved memory efficiently by storing the number of samples $\boldsymbol{n}$ taken at each level and the random number generator seeds that have ultimately determined the samples. Let the subscript $f$ in $\nabla \hat{J}_f(\boldsymbol{u})$ denote that the gradient in some $\boldsymbol{u}$ is calculated using some

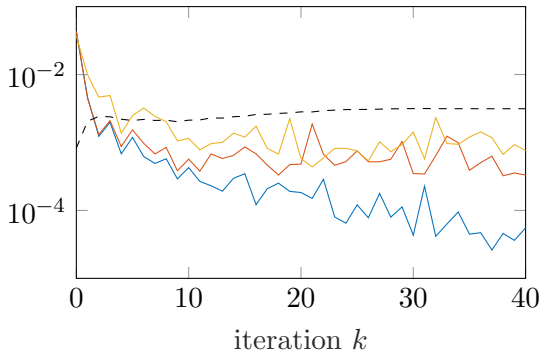



**Figure 3.** *Effect of using fixed or new samples on the evolution of the gradient; see section* 7.1.1. $\|\nabla \hat{J}_f(\boldsymbol{u}_f^{(k)})\|$ (——), $\|\nabla \hat{J}_\$(\boldsymbol{u}_f^{(k)})\|$ (——), $\|\nabla \hat{J}_\$(\boldsymbol{u}_\$^{(k)})\|$ (——). *The expected RMSE* $\epsilon$ *of* $\nabla \hat{J}_f(\boldsymbol{u}_f^{(k)})$ (- - -).

fixed set of samples $f$. The function $\nabla \hat{J}_f$ is then deterministic. Because $\boldsymbol{n}$ and $L$ are then given, the bias and many other things do not have to be estimated again. The case where new samples are taken is denoted with the subscript \$, e.g., $\nabla \hat{J}_\$(\boldsymbol{u})$.

To demonstrate the effect of any fixed or new samples, we perform 40 NCG-DY optimization steps using $\nabla \hat{J}_f$. Here, $f$ is determined during the first gradient evaluation call with the tolerance for the underlying MLMC algorithm set to $\epsilon = 1\text{e}{-3}$. The blue line in Figure 3 shows the decay of the norm of $\nabla \hat{J}_f$. The resulting sequence of control inputs $\{\boldsymbol{u}_f^{(k)}\}_{k=0}^{40}$ are then evaluated using $\nabla \hat{J}_\$$. Only this new sample gradient $\nabla \hat{J}_\$(\boldsymbol{u})$ is relevant to assess the quality of a control input $\boldsymbol{u}$. After all, the solution must perform well for the original problem, not just for the specific set of fixed samples $f$. The norm (15) of $\nabla \hat{J}_\$(\boldsymbol{u}_f^{(k)})$ is shown as the red line in Figure 3. Observe that the decay levels off at a certain point because $\nabla \hat{J}_f$ only resolves the gradient up to some RMSE $\epsilon$, which we have estimated using (27) and (28). It is thus important either to stop at that point or to decrease $\epsilon$, generating a new, larger set of samples. For comparison NCG-DY is also executed using $\nabla \hat{J}_\$$, i.e., while always using new random samples in each iteration. The resulting iterates, denoted by $\{\boldsymbol{u}_\$^{(k)}\}_{k=0}^{40}$, are outperformed by those produced using fixed samples, even when tested using new samples. This is the argument for holding onto the fixed samples as long as possible.

Consider again the sequence $\{\boldsymbol{u}_f^{(k)}\}_{k=0}^{40}$. As the iterates come closer to the minimizer $\bar{\boldsymbol{u}}$, the variances $\mathbb{V}[\boldsymbol{Y}_\ell]$ tend to converge to some constant and, in general, nonzero level. Indeed, if $k \to \infty$, $\boldsymbol{u}^{(k)} \to \bar{\boldsymbol{u}}$ and the variances in consideration tend to those for $\bar{\boldsymbol{u}}$. This is illustrated in Figure 4, where $\|\mathbb{V}[\boldsymbol{Y}_\ell]\|_\infty$ and $\|\max\{\frac{1}{2}\mathbb{V}[\boldsymbol{Y}_\ell], \mathbb{V}[\boldsymbol{Y}_\ell] + 2\sum_{j=2}^{b+1} \mathrm{Cov}[\boldsymbol{Y}_{\ell,1}, \boldsymbol{Y}_{\ell,j}]\}\|_\infty$ are shown for each level as a function of $k$. The use of correlated samples clearly reduces the variance on all levels. Observe also that during the first few iterations, the variances still change substantially. This causes the expected RMSE $\epsilon$ to fluctuate in Figure 3.

**7.1.2. Algorithm.** Let $\tau$ denote the tolerance on the gradient norm and consider Algorithm 2. Line 2 evaluates the initial gradient and collects the samples into an object $f$. Lines 4–8 implement the stopping condition. If $\|\boldsymbol{g}^{(k)}\| \leq \tau$, the current iterate is checked again using new samples (see line 5) before returning it. Lines 9–10 describe the optimization step. Other optimization algorithms can also be used here. Lines 11–16 govern the generation of the

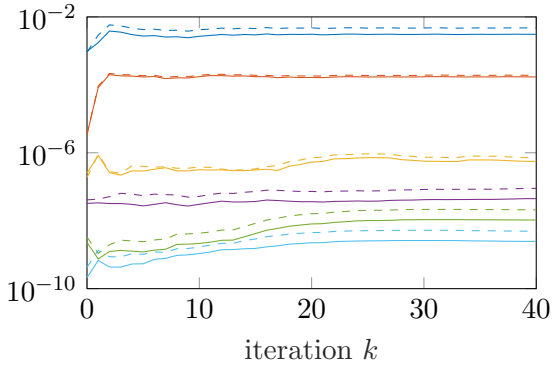



**Figure 4.** *Evolution of the variances.* $\|\mathbb{V}[\boldsymbol{Y}_\ell]\|_\infty$ *(- - -) and* $\|\max\{\frac{1}{2}\mathbb{V}[\boldsymbol{Y}_\ell], \mathbb{V}[\boldsymbol{Y}_\ell] + 2\sum_{j=2}^{b+1} Cov[\boldsymbol{Y}_{\ell,1}, \boldsymbol{Y}_{\ell,j}]\}\|_\infty$ *(——) for levels* $\ell = \{0, \ldots, 5\}$*. A higher line always corresponds to a coarser grid.*

gradient. In each iteration step, either the gradient is calculated using new samples, which are stored in $f$ (see line 13), or the gradient is calculated using the existing fixed sample set $f$ (see line 15). In that case the expected RMSE $\epsilon$ is also calculated. During the optimization procedure, a gradient is useful only if the gradient estimator $\boldsymbol{g}^{(k)}$ has an RMSE $\epsilon \leq q\|\boldsymbol{g}^{(k)}\|$. The constant $q$, which we set to 1 for all experiments, essentially determines how large the relative RMSE $\epsilon/\|\boldsymbol{g}^{(k)}\|$ is allowed to be. An error of a given size has little effect on a large gradient but may completely distort a small gradient. Algorithm 2 therefore keeps $\epsilon$ proportional to the currently attained gradient norm. Note that this is much more efficient than keeping $\epsilon$ proportional to the target norm[3] $\tau$ since, in the former approach, all but the last few iterations are then computed using a larger $\epsilon$. The samples are reused as long as possible until $\epsilon \leq q\|\boldsymbol{g}^{(k)}\|$ no longer holds; see line 11. At that point $\epsilon$ is reduced by a well-chosen factor $\eta$. A smaller $\eta$ causes a given set of samples to last for more iterations, but the requested tolerance $\epsilon$ might then be reduced more than necessary, which is inefficient. In this paper we use $\eta = 0.2$. Note that $\epsilon$ should not be reduced below $q\tau$ as the iteration will stop approximately when $\|\boldsymbol{g}^{(k)}\| \leq \tau$ anyway. This is the reason for the max in lines 11–12. Note that line 11 also checks if $\epsilon$ is unnecessarily small by testing $\epsilon^{(k)} < \eta^2 q\|\boldsymbol{g}^{(k)}\|$.

**7.1.3. Performance.** For the NCG-DY algorithm, the norm of the gradient is known to converge linearly, i.e., $||\boldsymbol{g}^{(k)}|| = \mathcal{O}(e^{-k})$. We assume that this linear convergence is retained even though the sample set that generates the gradient changes at some of the iterations. We also assume that a single triple $\rho, \phi, \kappa$ exists such that the three assumptions[4] in Theorem 5.1 hold uniformly, i.e., for the same constants implicit in $\lesssim$, for each point on the return level $\bar{L}$. This follows from simply considering the constants corresponding to the worst case point. Furthermore, it is reasonable to also assume uniformity from a certain optimization step onward. It was observed already that the variances converge to fixed values, meaning $\phi$ converges to a fixed value. The costs at each level, and therefore also $\kappa$, are constant by design.

[3]This behavior can still be achieved using Algorithm 2 by setting $\epsilon^{(0)} = q\tau$ and $\eta = 0$.
[4]For correlated samples, the second assumption is amended as suggested by (36); see subsection 6.2.

**Algorithm 2.** Gradient based optimization

input $\tau$, $q$, $\epsilon^{(0)}$, $\eta$, $k_{\max}$, $\boldsymbol{u}^{(0)}$, $\nabla \hat{J}_{\cdot}(.)$
$(\boldsymbol{g}^{(0)}, f) \leftarrow \nabla \hat{J}_{\$}(\boldsymbol{u}^{(0)})$ using RMSE $\epsilon^{(0)}$. ▷ Save new sample data in $f$
**for** $k = 0, \ldots, k_{\max} - 1$ **do**
  **if** $\|\boldsymbol{g}^{(k)}\| \leq \tau$ **then** ▷ Test convergence using current samples
    **if** $\|\nabla \hat{J}_{\$}(\boldsymbol{u}^{(k)})\| \leq \tau$ **then** ▷ Test convergence using new random samples
      **return** $\boldsymbol{u}^{(k)}$
    **end if**
  **end if**
  Get $\boldsymbol{d}^{(k)}$ and $s^{(k)}$, e.g., using NCG-DY and approximate linesearch
  $\boldsymbol{u}^{(k+1)} \leftarrow \boldsymbol{u}^{(k)} + s^{(k)}\boldsymbol{d}^{(k)}$
  **if** $\epsilon^{(k)} > \max\{q\tau, q\|\boldsymbol{g}^{(k)}\|\}$ or $\epsilon^{(k)} < \eta^2 q\|\boldsymbol{g}^{(k)}\|$ **then**
    $\epsilon^{(k+1)} \leftarrow \max\{q\tau, \eta q\|\boldsymbol{g}^{(k)}\|\}$
    $(\boldsymbol{g}^{(k+1)}, f) \leftarrow \nabla \hat{J}_{\$}(\boldsymbol{u}^{(k+1)})$ using RMSE $\epsilon^{(k+1)}$. ▷ Save new sample data in $f$
  **else**
    $(\boldsymbol{g}^{(k+1)}, \epsilon^{(k+1)}) \leftarrow \nabla \hat{J}_{f}(\boldsymbol{u}^{(k+1)})$ ▷ Calculate expected RMSE $\epsilon$
  **end if**
**end for**

Theorem 7.1 (MLMC optimization cost). *Let $\rho, \phi, \kappa$ exist such that the assumptions in Theorem 5.1 hold uniformly for each point on the return level $\bar{L}$ and from a certain optimization step onward and let $\epsilon^{(0)}$ be independent of $\tau$. If the norm of the gradient converges linearly, the cost $\mathcal{C}_{opt}(\tau)$ for Algorithm 2 to reach a gradient $\|\boldsymbol{g}^{(k)}\| \leq \tau$ is*

$$\mathcal{C}_{opt}(\tau) \lesssim \begin{cases} \tau^{-2} & \text{if } \phi > \kappa, \\ \tau^{-2}(\log \tau)^2 & \text{if } \phi = \kappa, \\ \tau^{-2-(\kappa-\phi)/\rho} & \text{if } \phi < \kappa, \end{cases} \qquad \tau \to 0. \tag{40}$$

*Proof.* Since $||\boldsymbol{g}^{(k)}|| = \mathcal{O}(e^{-k})$, the number of iterations $K$ needed to satisfy the tolerance $\tau$ is $K = \mathcal{O}(-\log \tau)$. Consider the three cases in (23). From a certain optimization step onward, the case in which one finds oneself remains the same. We now consider each case separately. Assume the cost of a gradient evaluation to be $\mathcal{C}(\boldsymbol{g}^{(k)}) \lesssim (\epsilon^{(k)})^{-2}$, i.e., assume the first case in Theorem 5.1. If $\epsilon^{(k)} \simeq \|\boldsymbol{g}^{(k)}\|$, the total cost is

$$\mathcal{C}_{\text{opt}}(\tau) = \sum_{k=0}^{K} \mathcal{C}(\boldsymbol{g}^{(k)}) \lesssim \sum_{k=0}^{K} \|\boldsymbol{g}^{(k)}\|^{-2} \simeq \sum_{k=0}^{K} e^{2k} = \frac{e^{2K+2} - 1}{e^2 - 1} = \mathcal{O}(\tau^{-2}), \quad \tau \to 0.$$

The case $\phi = \kappa$ in Theorem 5.1 has $\mathcal{C}(\boldsymbol{g}^{(k)}) \lesssim (\epsilon^{(k)})^{-2}(\log \epsilon^{(k)})^2$ and yields

$$\sum_{k=0}^{K} \|\boldsymbol{g}^{(k)}\|^{-2}(\log \|\boldsymbol{g}^{(k)}\|)^2 \simeq \sum_{k=0}^{K} k^2 e^{2k} = \mathcal{O}(K^2 e^{2K}) = \mathcal{O}(\tau^{-2}(\log \tau)^2), \quad \tau \to 0.$$

The third case is mathematically analogous to the first. ∎

This cost is proportional to the cost of a single gradient evaluation with RMSE $\tau$. Note that if instead the tolerance would be kept fixed, i.e., $\epsilon^{(0)} = \cdots = \epsilon^{(k)} \simeq \tau, \tau \to 0$, the total cost would amount to

$$\mathcal{C}_{\text{opt}}(\tau) = \mathcal{C}(\boldsymbol{g}^{(k)})\mathcal{O}(-\log\tau) \lesssim \begin{cases} -\tau^{-2}\log\tau & \text{if } \phi > \kappa, \\ -\tau^{-2}(\log\tau)^3 & \text{if } \phi = \kappa, \\ -\tau^{-2-(\kappa-\phi)/\rho}\log\tau & \text{if } \phi < \kappa, \end{cases} \qquad \tau \to 0.$$

**7.2. Hessian based optimization.** For a general problem, the cost functional can be approximated at a certain iteration as

$$\tilde{J}(\boldsymbol{u}^{(k)} + \Delta\boldsymbol{u}) \approx \tilde{J}(\boldsymbol{u}^{(k)}) + (\nabla\tilde{J}(\boldsymbol{u}^{(k)}))^T\Delta\boldsymbol{u}^{(k)} + \frac{1}{2}\Delta\boldsymbol{u}^T(\text{Hess }\tilde{J}(\boldsymbol{u}^{(k)}))\Delta\boldsymbol{u}.$$

For our quadratic model problem, this approximation is exact. Taking the derivative to $u$ and setting it to $\mathbf{0}$ yields

$$(\text{Hess }\tilde{J}(\boldsymbol{u}^{(k)}))\Delta\boldsymbol{u} = -\nabla\tilde{J}(\boldsymbol{u}^{(k)}). \tag{41}$$

Solving for $\Delta\boldsymbol{u}$ constitutes a Newton step and produces the next point $\boldsymbol{u}^{(k+1)} = \boldsymbol{u}^{(k)} + \Delta\boldsymbol{u}$. The model problem converges in one single Newton iteration. Of course, in general, multiple Newton steps are required.

As was already noted, it is infeasible to directly solve (41) because the Hessian is very large and dense. Therefore, a CG method is employed, as described in, e.g., [39]. This method requires only Hessian vector products and solves symmetric positive definite systems of equations. It can be shown that for a quadratic problem, the NCG method and a single solve of (41) using the CG method are equivalent, assuming the linesearches are exact. A concise overview of these relationships can be found in, e.g., [32]. This allows us to reuse some results from the previous section. Notably, Figure 3 can be reinterpreted as being about the residual $\boldsymbol{r}^{(i)} = (\text{Hess }\hat{J}(\boldsymbol{u}^{(k)}))\Delta\boldsymbol{u}^{(i)} + \nabla\hat{J}(\boldsymbol{u}^{(k)})$ for iteration $i$ of the CG method. Analogously, we conclude that the CG iterations can be meaningful only as long as $\epsilon \leq q\|\boldsymbol{r}^{(i)}\|$, with $q$ having the same meaning as before, leading to a residual tolerance of $q^{-1}\epsilon$. A possible algorithm is then given as Algorithm 3. It is constructed such that the CG method can be thought of as a black box solver (which allows one to swap it with other iterative methods). The sample set used in a single Newton iteration is determined by the RMSE $\epsilon$ requested for the gradient. The first iteration uses a large RMSE $\epsilon^{(0)}$ to cheaply get somewhat close to the optimizer $\bar{\boldsymbol{u}}$. Subsequent Newton steps lower the RMSE by a factor $\eta$ (we take $\eta = 0.2$) until $q\tau$ (we take $q = 1$) is reached.

**7.2.1. Performance.** Close to the solution, the Hessian does not change significantly. Consider the sequence of all CG iterations during all the Newton steps in the algorithm and index it with $i$. The total amount of steps until convergence is $I = \mathcal{O}(-\log\tau)$. From the algorithm it is clear that $\epsilon^{(i)} \lesssim \|\boldsymbol{r}^{(i)}\|$ (where $\epsilon^{(i)}$ now denotes the value of $\epsilon$ during CG iteration $i$). The cost of a single CG iteration, denoted $\mathcal{C}(\text{CG}^{(i)})$, is dominated by the Hessian vector product, which uses the samples $f$ in line 3. Hence, this cost is again given by Theorem 5.1 as a function of $\epsilon^{(i)}$. Everything thus being analogous to the gradient case, working out $\mathcal{C}_{\text{opt}}(\tau) = \sum_{i=0}^{I}\mathcal{C}(\text{CG}^{(i)})$ leads again to (40).

**Algorithm 3.** Hessian based optimization

input $\tau$, $q$, $\epsilon^{(0)}$, $\eta$, $k_{\max}$, $\boldsymbol{u}^{(0)}$, $\nabla \hat{J}_{\cdot}(.)$, Hess $\hat{J}_{\cdot}(.)[.]$
**for** $k = 0, \ldots, k_{\max} - 1$ **do**
  $(\boldsymbol{g}^{(k)}, f) \leftarrow \nabla \hat{J}_{\$}(\boldsymbol{u}^{(k)})$ using RMSE $\epsilon^{(k)}$.
  **if** $\|\boldsymbol{g}^{(k)}\| \leq \tau$ **then**
    **if** $\|\nabla \hat{J}_{\$}(\boldsymbol{u}^{(k)})\| \leq \tau$ **then**
      **return** $\boldsymbol{u}^{(k)}$
    **end if**
  **end if**
  $\Delta \boldsymbol{u} \leftarrow CG(\text{Hess } \hat{J}_f(\boldsymbol{u}^{(k)})[.], -\boldsymbol{g}^{(k)})$ with residual tolerance $q^{-1}\epsilon^{(k)}$.
  $\boldsymbol{u}^{(k+1)} \leftarrow \boldsymbol{u}^{(k)} + \Delta \boldsymbol{u}$
  $\epsilon^{(k+1)} \leftarrow \max\{q\tau, \eta\epsilon^{(k)}\}$
**end for**

**8. Numerical results.** This section contains results of a set of numerical experiments in which the gradient and Hessian based optimization algorithms are applied to the model problem. The algorithms are also tested on a nonlinear problem in section 8.3. The target function is set as

$$y_D(x) = \begin{cases} 1 & x \in [0.25, 0.75] \times [0.25, 0.75], \\ 0 & \text{otherwise.} \end{cases}$$

The following table contains all parameters that we fix for all experiments:

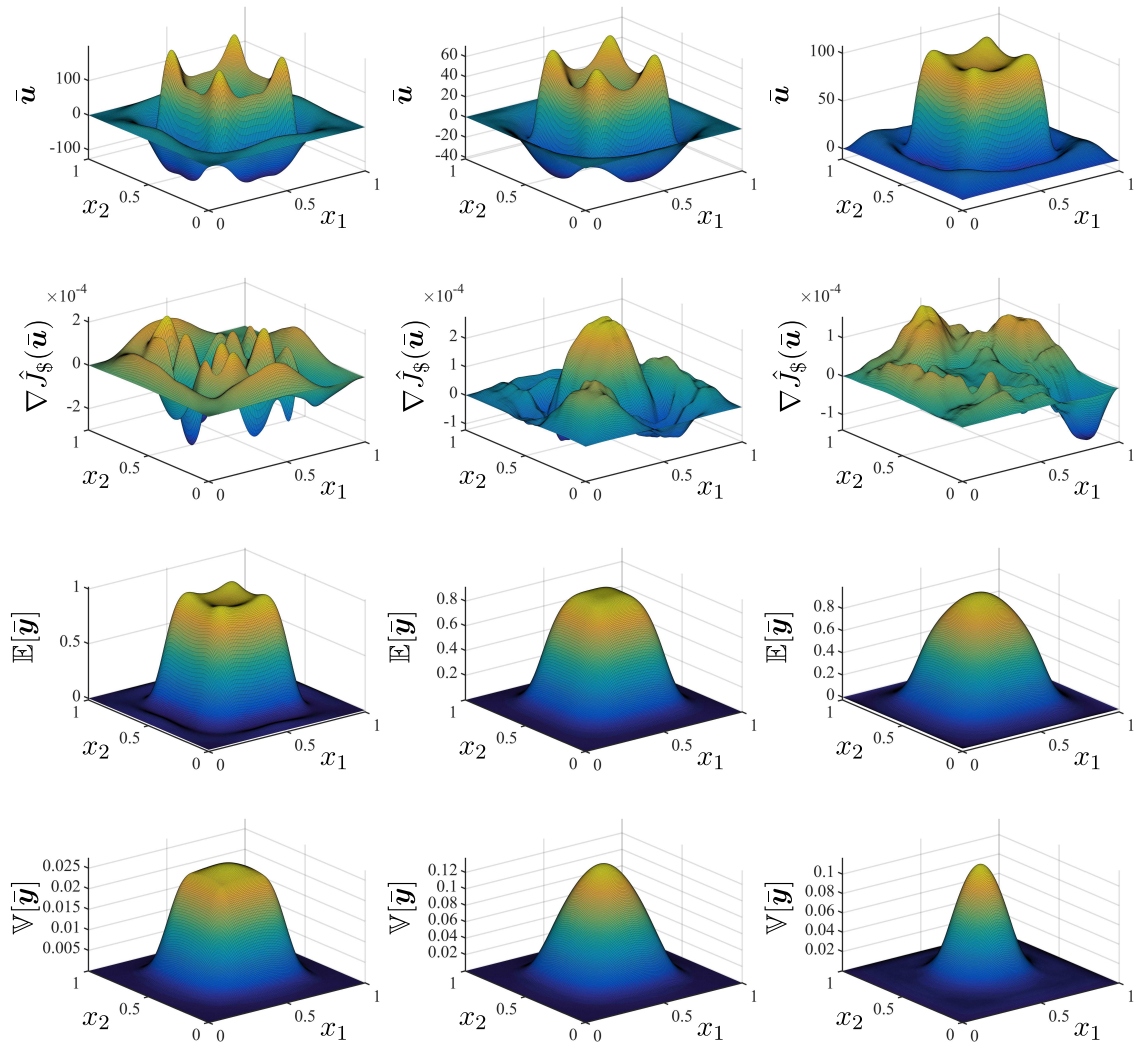
| uncertainty | solver parameters | | |
|---|---|---|---|
| $\lambda = 0.3$ | $m_0 = 8$ | $q = 1$ | $\epsilon^{(0)} = 1\text{e}{-2}$ |
| $n_{\text{KL}} = 500$ | $m_{\bar{L}} = 256$ | $\eta = 0.2$ | $\boldsymbol{u}^{(0)} = \boldsymbol{0}$ |

All calculations are performed in MATLAB on an Intel Core i5-5200U CPU @ 2.20GHz. The algebraic system of equations resulting from the finite volume discretization of the PDEs are solved using the MATLAB sparse matrix solver. This can be shown experimentally to yield $\kappa = 2.26$ in Theorems 5.1 and 7.1 for our 2D problem. In these experiments $\phi > \kappa$, and therefore the dominant cost is on the coarsest level. The optimization results for the next three problems can be found in Figure 5.

**8.1. Problem 1.** Problem 1 is further defined by

$$\alpha = 1\text{e}{-6}, \gamma = 1, \tau = 1\text{e}{-4}, \sigma^2 = 0.1; \tag{42}$$

see also Figure 1. The problem is solved using Algorithms 2 and 3. The convergence behavior of both methods is visualized in Figure 6. The total time (with all overhead included) was $1542s$ (gradient based) and $1989s$ (Hessian based). Tables 1 and 2 give sampling information each time new samples are generated for both methods. At the finer levels, the number of initial samples sometimes appears, meaning that no additional samples were required beyond those initial samples. The timings are calculated as the average wallclock time for a single




(a) Problem 1 (b) Problem 2 (c) Problem 3

**Figure 5.** *Optimization results.* $256 \times 256$.

NCG or CG iteration that uses the indicated number of samples. For comparison, taking $n_0$ samples on level 4, as would be approximately the case for the classical MC method, would take an estimated 19 hours for a single NCG iteration and 12 hours for a single CG iteration for $\epsilon = 1\text{e}{-4}$.

Figure 7 shows (a cross section of) the contributions to the gradient on each level. Note that the smoothness of these contributions follows from the discussion about Figure 2. Since the gradient must converge to zero, the contributions cancel each other out more effectively in the later iterations. Nevertheless, the variances remain much higher on the coarsest levels (the behavior is similar to the behavior observed in Figure 4). Therefore, removing the coarsest level (i.e., setting $m_0 = 16$ instead) would not improve performance.

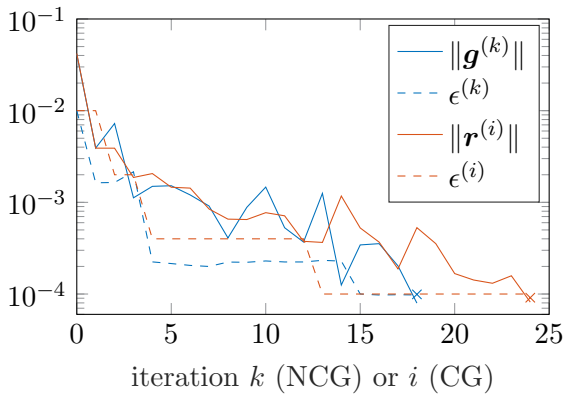



**Figure 6.** *Behavior of Algorithms 2 and 3 for Problem 1. The crosses (×) indicate the results of convergence tests performed using new samples.*

**Table 1**

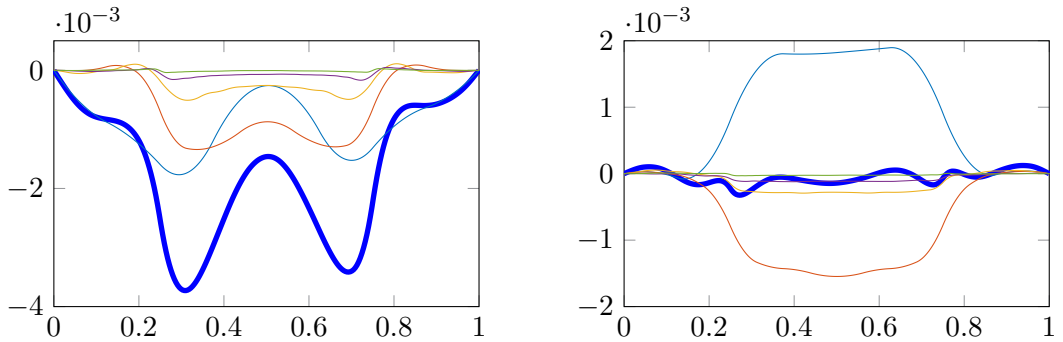
*Behavior of gradient based optimization (Algorithm 2).*

| $k$ | $\epsilon^{(k)}$ | $n_0$ | $n_1$ | $n_2$ | $n_3$ | $n_4$ | $n_5$ | Estimate of $\rho$ | $t^{(k)}[s]$ |
|---|---|---|---|---|---|---|---|---|---|
| 0 | 1e−2 | 140 | 76 | 44 | | | | 2.0237 | 2.05 |
| 4 | 2.24e−4 | 17150 | 1512 | 80 | 28 | 20 | | 1.5824 | 47.49 |
| 15 | 1e−4 | 98452 | 9156 | 940 | 118 | 20 | | 1.5825 | 248.84 |

**Table 2**

*Behavior of Hessian based optimization (Algorithm 3).*

| $i$ | $\epsilon^{(i)}$ | $n_0$ | $n_1$ | $n_2$ | $n_3$ | $n_4$ | $n_5$ | Estimate of $\rho$ | $t^{(i)}[s]$ |
|---|---|---|---|---|---|---|---|---|---|
| 0 | 1e−2 | 140 | 76 | 44 | | | | 1.8355 | 4.12 |
| 2 | 2e−3 | 140 | 76 | 44 | | | | 1.7030 | 4.36 |
| 4 | 4e−4 | 5964 | 521 | 44 | 28 | 20 | | 1.5905 | 14.45 |
| 13 | 1e−4 | 96159 | 9010 | 821 | 93 | 22 | | 1.6195 | 153.42 |




(a) Iteration $k = 5$ (b) Iteration $k = 18$ (last)

**Figure 7.** *Cross section of $\boldsymbol{g}^{(k)} = \sum_{\ell=0}^{L} I_\ell^{\bar{L}} \hat{\boldsymbol{Y}}_{\ell,n_\ell}^{MC}$ (——) and of contributions $I_\ell^{\bar{L}} \hat{\boldsymbol{Y}}_{\ell,n_\ell}^{MC}$ at levels $0, \ldots, L$ (——, ——, ——, ——, ——) for Problem 1.*

**8.2. Problem 2.** Problem 2 is further defined by

$$\alpha = 1\text{e}{-5}, \gamma = 0, \tau = 1\text{e}{-4}, \sigma^2 = 0.5. \tag{43}$$

Figure 8 and Tables 3 and 4 describe the behavior of the optimization algorithms in the same way as before. The total time (with all overhead included) was $6973s$ (gradient based) and $5114s$ (Hessian based). For $\epsilon = 1\text{e}{-4}$, taking $n_0$ samples at level 5 instead, would take approximately 427 hours for a single NCG iteration and 259 hours for a single CG iteration.

**8.3. Problem 3.** Consider as an example the following nonlinear extension to the model problem:

$$-\nabla \cdot (k\nabla y) + f(y) = \beta u \quad \text{on } D \quad \text{with} \quad y = 0 \quad \text{on } \partial D. \tag{44}$$

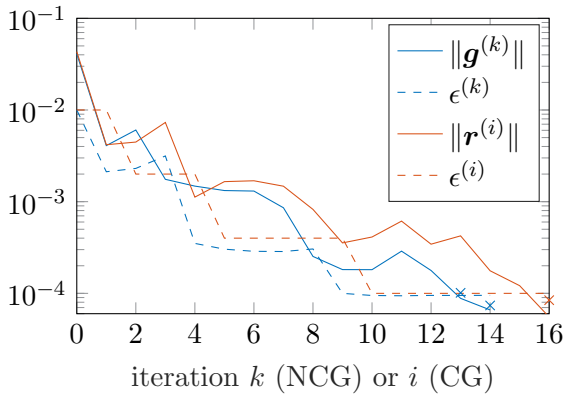



**Figure 8.** *Behavior of Algorithms 2 and 3 for Problem 2. The crosses indicate convergence tests performed using new samples.*

**Table 3**
*Behavior of gradient based optimization (Algorithm 2).*

| $k$ | $\epsilon^{(k)}$ | $n_0$ | $n_1$ | $n_2$ | $n_3$ | $n_4$ | $n_5$ | Estimate of $\rho$ | $t^{(k)}[s]$ |
|---|---|---|---|---|---|---|---|---|---|
| 0 | 1e−2 | 140 | 76 | 44 | | | | 2.0237 | 2.06 |
| 4 | 3.51e−4 | 35563 | 3220 | 136 | 28 | 20 | | 1.5824 | 93.44 |
| 9 | 1e−4 | 375256 | 38259 | 2082 | 135 | 21 | 16 | 1.5825 | 1092.71 |

**Table 4**
*Behavior of Hessian based optimization (Algorithm 3).*

| $i$ | $\epsilon^{(i)}$ | $n_0$ | $n_1$ | $n_2$ | $n_3$ | $n_4$ | $n_5$ | Estimate of $\rho$ | $t^{(i)}[s]$ |
|---|---|---|---|---|---|---|---|---|---|
| 0 | 1e−2 | 140 | 76 | 44 | | | | 1.9102 | 8.89 |
| 2 | 2e−3 | 393 | 76 | 44 | | | | 2.0975 | 11.77 |
| 5 | 4e−4 | 33063 | 6980 | 193 | 28 | 20 | | 1.7029 | 76.20 |
| 10 | 1e−4 | 388834 | 37023 | 1747 | 255 | 56 | 16 | 1.7818 | 668.58 |

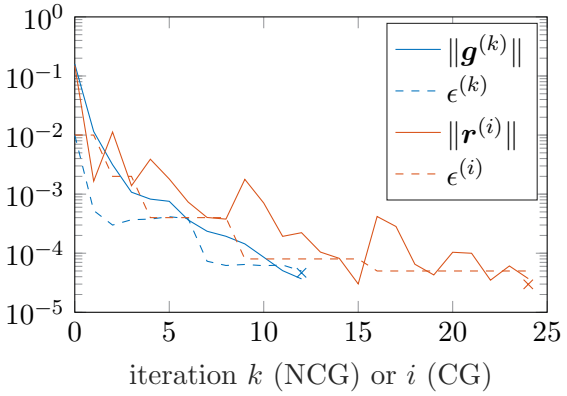



**Figure 9.** *Behavior of Algorithms 2 and 3 for Problem 3. The crosses indicate convergence tests performed using new samples.*

The function $f$ is some nonlinear reaction term. From (7) it is clear that only the term $(\frac{\partial c}{\partial y})^* p$ needs updating, leading to

$$\left\{\begin{array}{rcll} -\nabla\cdot(k\nabla y)+f(y) &=& \beta u & \text{on } D,\\ -\nabla\cdot(k\nabla p)+f'(y)p &=& 2(y-y_D)+2\gamma(y-\mathbb{E}[y]) & \text{on } D,\\ \nabla\tilde{J}(u) &=& 2\alpha u+\beta\mathbb{E}[p]. & \end{array}\right. \tag{45}$$

The expression $f'$ denotes the derivative of $f$, which is again localized ($f'(y)(x) = g'(y(x))$). The derivation of the Hessian equations is slightly more involved. We only state the result:

$$\left\{\begin{array}{rcll} -\nabla\cdot(k\nabla\,\delta y)+f'(y)\,\delta y &=& \beta\,\delta u. & \text{on } D,\\ -\nabla\cdot(k\nabla\,\delta p)+f'(y)\,\delta p+f''(y)p\,\delta y &=& 2\,\delta y+2\gamma(y-\mathbb{E}[\delta y]) & \text{on } D,\\ \text{Hess }\tilde{J}(u)[\delta u] &=& 2\alpha\,\delta u+\beta\mathbb{E}[\delta p]. & \end{array}\right. \tag{46}$$

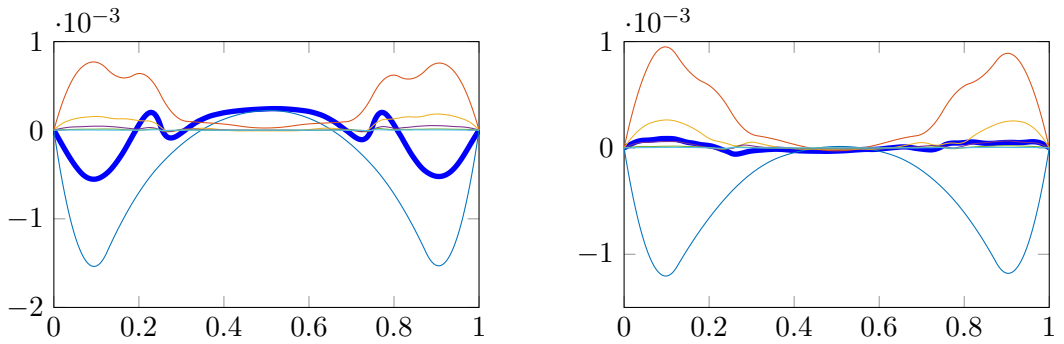
It is important to note that the adjoint equation is always linear in $p$ for the gradient and $\delta p$ for the Hessian. Therefore, the sampling methods from section 6 can always be used.

Consider as an example the nonlinear term $f(y) = 20 + e^{5y}$ and the parameters

$$\alpha = 1\text{e}{-5}, \gamma = 1, \tau = 5\text{e}{-5}, \sigma^2 = 0.5. \tag{47}$$

Figure 9 shows the convergence plot. Figure 10 shows a cross section of the contributions to the gradient on each level. The details obtained during the optimization are described in Tables 5 and 6 for both algorithms. The total time (with all overhead included) was $2607s$ (gradient based) and $8307s$ (Hessian based). For $\epsilon = 5\text{e}{-5}$, a single NCG and CG iteration would take approximately 73 hours and 69 hours, respectively.

**9. Conclusions and further work.** We presented an MLMC method for solving the robust optimization problem for a tracking type cost functional. Including an additional penalty on the variance of the state allows us to also solve the average control problem in the same framework. It has been shown that correlations between a limited number of samples of the gradient are hard to avoid if efficiency and correct error estimation are desired. The classical MLMC theory was extended to be able to deal with these samples. Since the correlations are




(a) Iteration $k = 7$ (b) Iteration $k = 12$ (last)

**Figure 10.** *Cross section of* $\boldsymbol{g}^{(k)} = \sum_{\ell=0}^{L} I_\ell^{\bar{L}} \hat{\boldsymbol{Y}}_{\ell,n_\ell}^{MC}$ (——) *and of contributions* $I_\ell^{\bar{L}} \hat{\boldsymbol{Y}}_{\ell,n_\ell}^{MC}$ *at levels* $0, \ldots, L$ (——, ——, ——, ——, ——, ——) *for Problem* 3.

**Table 5**

*Behavior of gradient based optimization (Algorithm 2).*

| $k$ | $\epsilon^{(k)}$ | $n_0$ | $n_1$ | $n_2$ | $n_3$ | $n_4$ | $n_5$ | Estimate of $\rho$ | $t^{(k)}[s]$ |
|---|---|---|---|---|---|---|---|---|---|
| 0 | 1e−2 | 923 | 83 | 44 | | | | 1.6309 | 10.19 |
| 7 | 7.32e−5 | 13369 | 3093 | 391 | 43 | 20 | 16 | 1.6069 | 341.36 |
| 12 | 5e−5 | 33347 | 11435 | 711 | 77 | 20 | 16 | 1.5983 | 508.27 |

**Table 6**

*Behavior of Hessian based optimization (Algorithm 3).*

| $i$ | $\epsilon^{(i)}$ | $n_0$ | $n_1$ | $n_2$ | $n_3$ | $n_4$ | $n_5$ | Estimate of $\rho$ | $t^{(i)}[s]$ |
|---|---|---|---|---|---|---|---|---|---|
| 0 | 1e−2 | 826 | 76 | 44 | | | | 1.9102 | 9.95 |
| 2 | 2e−3 | 140 | 76 | 44 | | | | 2.0975 | 7.55 |
| 4 | 4e−4 | 305 | 108 | 44 | 28 | | | 1.5521 | 14.54 |
| 9 | 8e−5 | 14357 | 3211 | 205 | 28 | 20 | | 1.5931 | 153.21 |
| 16 | 5e−5 | 34181 | 7623 | 850 | 345 | 44 | 16 | 1.5964 | 593.68 |
| 20 | 5e−5 | 24287 | 7394 | 2647 | 122 | 20 | 16 | 1.6211 | 727.11 |
| 23 | 5e−5 | 26008 | 7058 | 1597 | 1057 | 20 | 16 | 1.5856 | 996.88 |

usually negative, the usage of correlated samples turns out to reduce the number of samples required.

The MLMC method proves to be orders of magnitude more efficient compared to the regular MC method, which takes all samples on the finest level. In our experiments, the NCG method is usually more performant than the Hessian based method, but not always. The performance depends on the specific problem considered. The Hessian based method may be better suited for a small number of optimization variables. The Newton equation could then be solved directly.The method was tested in this paper on simple elliptic academic model problems. Its performance for realistic applications has not yet been investigated. In theory, other types of PDE constraints, such as parabolic or hyperbolic PDEs, do not formally require any major adaptations to the proposed strategy. In practice, however, the challenge will be to ensure that the realizations for the same stochastic parameters on two consecutive discretization levels

remain highly correlated. This will depend on how the levels are defined precisely. Note also that any difficulties appearing in the deterministic PDE constrained optimization problem will of course remain in the stochastic case.

Methods for optimization under uncertainties are often born out of previous research in simulation and uncertainty quantification of stochastic problems. In that context, the quasi–Monte Carlo (QMC) method [18, 14] and its multilevel [25, 24] and multi-index [35] variants have been shown to reduce the cost for a given RMSE $\epsilon$ from $\mathcal{O}(\epsilon^{-2})$ to, in ideal circumstances, $\mathcal{O}(\epsilon^{-1})$. Hence, the use of QMC is expected to reduce the complexity of (40) even further. Some other variance reduction techniques, such as importance sampling [1, 20], might also be worth considering. The algorithmic details and the numerical evidence remain to be investigated.

**Appendix A. Gradient of $\hat{V}_1$.** This section provides the details of the derivation of the gradient of the variance estimator $\hat{V}_1$, i.e., we show how to arrive at (35). Note that $\|\sqrt{\boldsymbol{v}}\|^2 = (\boldsymbol{v}, \mathbf{1})$, such that

$$\frac{\mathrm{d}}{\mathrm{d}\boldsymbol{y}}\left\|\sqrt{\frac{1}{2n}\sum_{j=1}^{n}(\boldsymbol{y}_j-\boldsymbol{y}_{j-1})^2}\right\|^2[\boldsymbol{h}] = \frac{1}{2n}\frac{\mathrm{d}}{\mathrm{d}\boldsymbol{y}}\left(\sum_{j=1}^{n}(\boldsymbol{y}_j-\boldsymbol{y}_{j-1})^2,\mathbf{1}\right)[\boldsymbol{h}].$$

For simplicity, let $\boldsymbol{y}_i = \boldsymbol{y}_{n+i}$ and $\boldsymbol{h}_i = \boldsymbol{h}_{n+i}$. Using, e.g., the limit definition of the derivative yields

$$\begin{aligned}
\frac{1}{2n}\frac{\mathrm{d}}{\mathrm{d}\boldsymbol{y}}\left(\sum_{j=1}^{n}(\boldsymbol{y}_j-\boldsymbol{y}_{j-1})^2,\mathbf{1}\right)[\boldsymbol{h}] &= \frac{1}{n}\left(\sum_{j=1}^{n}(\boldsymbol{y}_j-\boldsymbol{y}_{j-1})(\boldsymbol{h}_j-\boldsymbol{h}_{j-1}),\mathbf{1}\right)\\
&= \frac{1}{n}\left(\sum_{j=1}^{n}\boldsymbol{y}_j\boldsymbol{h}_j+\sum_{j=1}^{n}\boldsymbol{y}_{j-1}\boldsymbol{h}_{j-1}-\sum_{j=1}^{n}\boldsymbol{y}_j\boldsymbol{h}_{j-1}-\sum_{j=1}^{n}\boldsymbol{y}_{j-1}\boldsymbol{h}_j,\mathbf{1}\right)\\
&= \frac{1}{n}\left(\sum_{j=1}^{n}\boldsymbol{y}_j\boldsymbol{h}_j+\sum_{j=1}^{n}\boldsymbol{y}_j\boldsymbol{h}_j-\sum_{j=1}^{n}\boldsymbol{y}_{j+1}\boldsymbol{h}_j-\sum_{j=1}^{n}\boldsymbol{y}_{j-1}\boldsymbol{h}_j,\mathbf{1}\right)\\
&= \frac{1}{n}\left(\sum_{j=1}^{n}(2\boldsymbol{y}_j-\boldsymbol{y}_{j-1}-\boldsymbol{y}_{j+1})\boldsymbol{h}_j,\mathbf{1}\right)\\
&= (2\boldsymbol{y}-\boldsymbol{y}_{\ominus 1}-\boldsymbol{y}_{\oplus 1},\boldsymbol{h})_{D,\Omega_0}.
\end{aligned}$$

Therefore, the gradient of $\hat{V}_1$ w.r.t. $(.,.)_{D,\Omega_0}$ is equal to $2\boldsymbol{y}-\boldsymbol{y}_{\ominus 1}-\boldsymbol{y}_{\oplus 1}$.

## REFERENCES

[1] M. B. Alaya, K. Hajji, and A. Kebaier, *Adaptive Importance Sampling for Multilevel Monte Carlo Euler Method*, preprint, arXiv:1603.02959, 2017.

[2] A. A. Ali, E. Ullmann, and M. Hinze, *Multilevel Monte Carlo analysis for optimal control of elliptic PDEs with random coefficients*, SIAM/ASA J. Uncertain. Quantif., 5 (2017), pp. 466–492.

[3] I. Babuška, F. Nobile, and R. Tempone, *A stochastic collocation method for elliptic partial differential equations with random input data*, SIAM Rev., 52 (2010), pp. 317–355.
[4] I. Babuška, R. Tempone, and G. E. Zouraris, *Galerkin finite element approximations of stochastic elliptic partial differential equations*, SIAM J. Numer. Anal., 42 (2004), pp. 800–825.
[5] A. Barth, A. Lang, and C. Schwab, *Multilevel Monte Carlo method for parabolic stochastic partial differential equations*, BIT, 53 (2013), pp. 3–27.
[6] A. Borzì and V. Schulz, *Multigrid methods for PDE optimization*, SIAM Rev., 51 (2009), pp. 361–395.
[7] A. Borzì and V. Schulz, *Computational Optimization of Systems Governed by Partial Differential Equations*, Comput. Sci. Eng. 8, SIAM, Philadelphia, 2012.
[8] A. Borzì, V. Schulz, C. Schillings, and G. Von Winckel, *On the treatment of distributed uncertainties in PDE-constrained optimization*, GAMM-Mitt., 33 (2010), pp. 230–246.
[9] A. Borzì and G. von Winckel, *Multigrid methods and sparse-grid collocation techniques for parabolic optimal control problems with random coefficients*, SIAM J. Sci. Comput., 31 (2009), pp. 2172–2192.
[10] A. Borzì and G. von Winckel, *A POD framework to determine robust controls in PDE optimization*, Comput. Vis. Sci., 14 (2011), pp. 91–103.
[11] P. Chen and A. Quarteroni, *Weighted reduced basis method for stochastic optimal control problems with elliptic PDE constraint*, SIAM/ASA J. Uncertain. Quantif., 2 (2014), pp. 364–396.
[12] K. A. Cliffe, M. B. Giles, R. Scheichl, and A. L. Teckentrup, *Multilevel Monte Carlo methods and applications to elliptic PDEs with random coefficients*, Comput. Vis. Sci., 14 (2011).
[13] Y.-H. Dai and Y. Yuan, *A nonlinear conjugate gradient method with a strong global convergence property*, SIAM J. Optim., 10 (1999), pp. 177–182.
[14] J. Dick, F. Y. Kuo, and I. H. Sloan, *High-dimensional integration: The quasi-Monte Carlo way*, Acta Numer., 22 (2013), pp. 133–288.
[15] C. R. Dietrich and G. N. Newsam, *Fast and exact simulation of stationary Gaussian processes through circulant embedding of the covariance matrix*, SIAM J. Sci. Comput., 18 (1997), pp. 1088–1107.
[16] R. G. Ghanem and P. D. Spanos, *Stochastic Finite Elements: A Spectral Approach*, Courier Corporation, Mineola, NY, 2003.
[17] M. B. Giles, *Multilevel Monte Carlo methods*, Acta Numer., 24 (2015), pp. 259–328.
[18] I. G. Graham, F. Y. Kuo, D. Nuyens, R. Scheichl, and I. H. Sloan, *Quasi-Monte Carlo methods for elliptic PDEs with random coefficients and applications*, J. Comput. Phys., 230 (2011), pp. 3668–3694.
[19] K. Karhunen, *Über lineare methoden in der wahrscheinlichkeitsrechnung*, Ann. Acad. Sci. Fenn. Math. Physi., 37 (1947), pp. 1–79.
[20] A. Kebaier and J. Lelong, *Coupling importance sampling and multilevel Monte Carlo using sample average approximation*, Methodol. Comput. Appl. Probab., 20 (2018), pp. 611–641.
[21] D. P. Kouri, *A multilevel stochastic collocation algorithm for optimization of PDEs with uncertain coefficients*, SIAM/ASA J. Uncertain. Quantif., 2 (2014), pp. 55–81.
[22] D. P. Kouri, M. Heinkenschloss, D. Ridzal, and B. G. van Bloemen Waanders, *A trust-region algorithm with adaptive stochastic collocation for PDE optimization under uncertainty*, SIAM J. Sci. Comput., 35 (2013), pp. A1847–A1879.
[23] D. P. Kouri and T. M. Surowiec, *Risk-averse PDE-constrained optimization using the conditional value-at-risk*, SIAM J. Optim., 26 (2016), pp. 365–396.
[24] F. Y. Kuo, R. Scheichl, C. Schwab, I. H. Sloan, and E. Ullmann, *Multilevel quasi-Monte Carlo methods for lognormal diffusion problems*, Math. Comp., 86 (2017), pp. 2827–2860.
[25] F. Y. Kuo, C. Schwab, and I. H. Sloan, *Multi-level quasi-Monte Carlo finite element methods for a class of elliptic PDEs with random coefficients*, Found. Comput. Math., 15 (2015), pp. 411–449.
[26] H.-C. Lee and M. D. Gunzburger, *Comparison of approaches for random PDE optimization problems based on different matching functionals*, Comput. Math. Appl., 73 (2017), pp. 1657–1672.
[27] H.-C. Lee and J. Lee, *A stochastic Galerkin method for stochastic control problems*, Commun. Comput. Phys., 14 (2013), pp. 77–106.
[28] R. M. Lewis and S. G. Nash, *Model problems for the multigrid optimization of systems governed by differential equations*, SIAM J. Sci., Comput., 26 (2005), pp. 1811–1837.
[29] M. Loève, *Fonctions aléatoires de second ordre*, Rev. Sci., (1946), pp. 195–206.

[30] J. Martin, L. C. Wilcox, C. Burstedde, and O. Ghattas, *A stochastic Newton MCMC method for large-scale statistical inverse problems with application to seismic inversion*, SIAM J. Sci. Comput., 34 (2012), pp. A1460–A1487.
[31] S. G. Nash, *A multigrid approach to discretized optimization problems*, Optim. Methods Softw., 14 (2000), pp. 99–116.
[32] J. Nazareth, *Conjugate gradient method*, Wiley Interdiscip. Rev. Comput. Stat., 1 (2009), pp. 348–353.
[33] F. Nobile, R. Tempone, and C. G. Webster, *A sparse grid stochastic collocation method for partial differential equations with random input data*, SIAM J. Numer. Anal., 46 (2008), pp. 2309–2345.
[34] J. Peypouquet, *Convex Optimization in Normed Spaces: Theory, Methods and Examples*, Springer, Cham, Switzerland, 2015.
[35] P. Robbe, D. Nuyens, and S. Vandewalle, *A multi-index quasi–Monte Carlo algorithm for lognormal diffusion problems*, SIAM J. Sci. Comput., 39 (2017), pp. S851–S872.
[36] E. Rosseel and G. N. Wells, *Optimal control with stochastic PDE constraints and uncertain controls*, Comput. Methods Appl. Mech. Engrg., 213 (2012), pp. 152–167.
[37] A. L. Teckentrup, R. Scheichl, M. B. Giles, and E. Ullmann, *Further analysis of multilevel monte carlo methods for elliptic pdes with random coefficients*, Numer. Math., 125 (2013), pp. 569–600.
[38] H. Tiesler, R. M. Kirby, D. Xiu, and T. Preusser, *Stochastic collocation for optimal control problems with stochastic PDE constraints*, SIAM J. Control Optim., 50 (2012), pp. 2659–2682.
[39] L. N. Trefethen and D. Bau III, *Numerical Linear Algebra*, SIAM, Philadelphia, 1997.
[40] F. Tröltzsch, *Optimal Control of Partial Differential Equations*, AMS, Providence, RI, 2010.
[41] D. Xiu and J. S. Hesthaven, *High-order collocation methods for differential equations with random inputs*, SIAM J. Sci. Comput., 27 (2005), pp. 1118–1139.
[42] D. Xiu and G. E. Karniadakis, *Modeling uncertainty in flow simulations via generalized polynomial chaos*, J. Comput. Phys., 187 (2003), pp. 137–167.