SIAM J. SCI. COMPUT. © 2021 Society for Industrial and Applied Mathematics
Vol. 43, No. 2, pp. A1412–A1440

MULTILEVEL DESIGNED QUADRATURE FOR PARTIAL DIFFERENTIAL EQUATIONS WITH RANDOM INPUTS[*]

VAHID KESHAVARZZADEH[†], ROBERT M. KIRBY[†‡], AND AKIL NARAYAN[†§]

Abstract. We introduce a numerical method, multilevel designed quadrature for computing the statistical solution of partial differential equations with random input data. Similar to multilevel Monte Carlo methods, our method relies on hierarchical spatial approximations in addition to a parametric/stochastic sampling strategy. A key ingredient in multilevel methods is the relationship between the spatial accuracy at each level and the number of stochastic samples required to achieve that accuracy. Our sampling is based on flexible quadrature points that are designed for a prescribed accuracy, which can yield less overall computational cost compared to alternative multilevel methods. We propose a constrained optimization problem that determines the number of samples to balance the approximation error with the computational budget. We further show that the optimization problem is convex and derive analytic formulas for the optimal number of points at each level. We validate the theoretical estimates and the performance of our multilevel method via numerical examples on a linear elasticity and a steady state heat diffusion problem.

Key words. multilevel Monte Carlo, stochastic partial differential equation, designed quadrature, hierarchical spatial approximation

AMS subject classifications. 65D32, 65C20, 65C30

DOI. 10.1137/20M1333407

1. Introduction.

The computer simulation of science and engineering problems is subject to various uncertainties, ranging from modeling inaccuracies to measurement errors. The ever-growing field of uncertainty quantification develops efficient mathematical tools to address such challenges in numerical simulations. In this paper we adopt the main ideas in multilevel Monte Carlo methods [15] to estimate statistical moments for solution of random partial differential equations (PDEs). Let h_i denote the resolution size (e.g., mesh size) for level $i \in \{0, \dots, L\}$ associated with spatial or temporal approximation/discretization with $h_L < \cdots < h_1 < h_0$, $\mathbb{E}$ the mathematical expectation operator, and Q_i the quantity of interest (e.g., a linear functional of the PDE solution) for that approximation, where Q_i is random due to the randomness in the PDE. Multilevel Monte Carlo exploits the linearity of expectations and writes the "true" expectation on the finest level as

$$\mathbb{E}[Q_L] = \mathbb{E}[Q_0] + \sum_{i=1}^{L} \mathbb{E}[Q_i - Q_{i-1}].$$

[*]Submitted to the journal's Methods and Algorithms for Scientific Computing section April 22, 2020; accepted for publication (in revised form) February 1, 2021; published electronically April 26, 2021.
https://doi.org/10.1137/20M1333407

Funding: This work was sponsored by ARL under cooperative agreement number W911NF-12-2-0023. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of ARL or the U.S. Government. The U.S. Government is authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation herein. The work of the first and third authors was partially supported by AFOSR FA9550-20-1-0338. The work of the third author was partially supported by DARPA EQUiPS N660011524053.

[†]Scientific Computing and Imaging Institute, University of Utah, Salt Lake City, UT 84112 USA (vkeshava@sci.utah.edu).
[‡]School of Computing, University of Utah, Salt Lake City, UT 84112 USA (kirby@cs.utah.edu).
[§]Department of Mathematics, University of Utah, Salt Lake City, UT 84112 USA (akil@sci.utah.edu).

Assuming the spatial/temporal discretizations are convergent, then Q_i is expected to be an increasingly accurate estimate for Q as i increases, so the variance of $Q_i - Q_{i-1}$ becomes smaller as i is increased. This smaller variance implies that Monte Carlo estimates for these differences require fewer samples for larger i but require a large number of samples for small i. This idea constitutes the main strategy of multilevel Monte Carlo, which saves significant computational cost by requiring fewer functional evaluations for terms associated with finer resolutions (which are more expensive to generate). Each term in the summation above can be estimated by M_i samples

$$\mathbb{E}[Q_i - Q_{i-1}] = \frac{1}{M_i}\sum_{j=1}^{M_i} Q_i^{(j)} - Q_{i-1}^{(j)},$$

where $M_L < \cdots < M_1 < M_0$ and $Q_i^{(j)}$ is a stochastic sample of Q_i. Our main goal in this paper is to take $Q = Q(u)$, where u is the solution to a parametric PDE,

$$-\nabla \cdot (a(\boldsymbol{y}, \boldsymbol{x})\nabla u(\boldsymbol{y}, \boldsymbol{x})) = f(x), \tag{1}$$

where the differential operators act on the spatial ($\boldsymbol{x}$) variable, $\boldsymbol{y}$ is a Euclidean N-dimensional parameter that we model as a random variable, and a is some prescribed diffusion coefficient depending on $\boldsymbol{y}$. The randomness in $\boldsymbol{y}$ makes $Q(u)$ a stochastic quantity.

The original multilevel approaches for stochastic approximations are based on Monte Carlo sampling [15]. As examples, readers are referred to [8, 21, 28, 29] for works on Monte Carlo-based approximation of random PDEs. To further enhance the performance of multilevel methods, it is possible to use deterministic sampling techniques which exhibit higher convergence rates for smooth dependence on the random variables [18, 19, 31]. Additional sample-based approaches for approximating random PDEs include collocation and quadrature approaches [3, 5, 14, 23, 30] and multilevel methods such as multi-index Monte Carlo [20, 26] and multi-index quasi-Monte Carlo [13, 17, 28]. These methods use sample-based quadrature or collocation-based polynomial approximations with respect to $\boldsymbol{y}$ to estimate the expectation. The main challenge for the existing quadrature-based work is the onerous dependence on the number of quadrature points to achieve a particular polynomial integration accuracy for large N.

In particular, these approaches utilize sparse grid methods and so are restricted to the number of samples that are dictated by the sparse-grid construction; however, there are alternative methods that can achieve similar accuracy with fewer sample points.

Our contributions in this paper are as follows. In this work we exploit the flexibility of "designed quadrature" [27], a framework for constructing unstructured multivariate quadrature rules, to generate quadrature rules for general quadrature grid sizes and polynomial integration accuracy. This increased flexibility allows us to formulate convex optimization problems using some a priori error and cost analysis for parameterized elliptic PDE. Taking the analysis in [31] as a template for multilevel estimates in tandem with designed quadrature allows us to formulate optimization problems over the number of samples allocated at each level to minimize the cost/error of the procedure, subject to accuracy/cost constraints. The optimization solution, which we derive explicitly and analytically, prescribes a particular number of samples that should be allocated at each level. This information would only be partially helpful for inflexible constructions like sparse grids. The flexibility offered by designed quadrature, however, allows one to generate quadrature grids whose cardinality exactly matches the output of the optimization. This allows us to use designed quadrature in a multilevel framework. We compare the performance of our method with other sampling techniques, namely, sparse grids for moderate dimensions and quasi-Monte Carlo for higher dimensions, in our numerical examples, and observe considerable computational acceleration, sometimes achieving two orders of magnitude speed-up.

The paper is organized as follows. In section 2 we introduce the problem and briefly discuss polynomial approximation for PDEs with random inputs, which motivates the use of polynomial-based quadrature rules. Section 3 discusses the abstract multilevel procedure, and section 4 describes designed quadrature and augmentations for use in this paper. Section 5 introduces the optimization problem to determine the number of function evaluations for each approximation level, provides an explicit solution to this problem, and finally summarizes the entire multilevel designed quadrature algorithm.

Section 6 presents numerical results for random PDEs, including a heat diffusion equation and a linear elasticity problem, both of the form (1), where the logarithm of a has an affine dependence on $\boldsymbol{y}$.

2. Polynomial approximation of parameterized elliptic PDEs.

This section discusses the approximation of parameterized elliptic PDEs using global polynomial approximation in the parametric variable. We review results from [10, 11] that establish high-order convergence of best n-term polynomial approximations. We will subsequently use these results to motivate a multilevel algorithm.

2.1. Notation and problem formulation.

On an open, bounded spatial domain $D \subset \mathbb{R}^d$, we consider the following parameterized elliptic PDE for the parameterized solution u:

$$\begin{aligned} -\nabla \cdot (a(\boldsymbol{y},\boldsymbol{x})\nabla u(\boldsymbol{y},\boldsymbol{x})) &= f(x) && \forall(\boldsymbol{y},\boldsymbol{x}) \in \Gamma \times D, \\ u(\boldsymbol{y},\boldsymbol{x}) &= 0 && \forall(\boldsymbol{y},\boldsymbol{x}) \in \Gamma \times \partial D, \end{aligned} \tag{2}$$

where $f \in H^{-1}(D)$ is a given inhomogenous term and the parameter set Γ is determined by the state space $\boldsymbol{y}$. We assume that $\boldsymbol{y}$ consists of N mutually independent random variables, i.e., $\boldsymbol{y} := (y_1(\omega), \dots, y_N(\omega)) \in \mathbb{R}^N$, where ω is the event variable on a complete probability space $(\Omega, \mathcal{F}, P)$. Each random variable y_j takes values in $\Gamma_j \subset \mathbb{R}$ with probability density $\rho_k(y_k)$. The independence assumption then implies that $\boldsymbol{y} \in \Gamma = \otimes_{j=1}^N \Gamma_j$ and has joint density $\rho(\boldsymbol{y}) := \prod_{i=1}^N \rho_i(y_i)$. We will assume throughout that, since $\boldsymbol{y}$ is a dimension-N vector, $a(\boldsymbol{y},\boldsymbol{x})$ has a finite-dimensional stochastic representation.

The main purpose of this section is to justify that polynomial approximations in the $\boldsymbol{y}$ variable can be effective. To make this argument, we assume, as is common (e.g., [9]) an *affine* dependence of the diffusion coefficient a on the parameters $\boldsymbol{y}$:

$$a(\boldsymbol{y},\boldsymbol{x}) = a_0(\boldsymbol{x}) + \sum_{i=1}^N a_i(\boldsymbol{x}) y_i(\omega), \tag{3}$$

where $\{a_j\}_{j=0}^N$ are the given functions. In the examples presented in section 6, we will actually assume that $\log(a - a_0)$ has affine dependence on the parameter. However, in this section we make the affine assumption to motivate the polynomial approximation strategy.

Although the system (2) involves a finite-dimensional coefficient $a(\boldsymbol{y},\boldsymbol{x})$, such a coefficient typically results from truncation of an infinite-dimensional random field $a(\omega,\boldsymbol{x})$ where $\omega \in \Omega$. If, for example, the coefficient is in $L^2(\Omega, L^2(D))$ then a Karhunen–Loéve expansion can be used to represent it, i.e.,

$$a(\boldsymbol{y},\boldsymbol{x}) = a_0(\boldsymbol{x}) + \sum_{i=1}^{\infty} a_i(\boldsymbol{x}) y_i(\omega),$$

where $\{y_i\}_{i\geq 1}$ are mean-zero, uncorrelated random variables. A truncation of this infinite sum along with an additional assumption of independence then results in model (2). A study of errors committed on the solution u by truncating an infinite-dimensional diffusion coefficient to a finite-dimensional series has been considered both in the affine case as presented above [10, 11] and also in the case when $\log(a - a_0)$ has an affine dependence on $\boldsymbol{y}$ [6].

In order to ensure well-posedness of (2), we make the assumption that there are positive deterministic constants $\underline{\mathrm{A}}$ and $\bar{A}$ such that the inequalities

$$0 < \underline{\mathrm{A}} \leq \inf_{x\in D} a(\boldsymbol{y},\boldsymbol{x}), \qquad \sup_{x\in D} a(\boldsymbol{y},\boldsymbol{x}) \leq \bar{A} < \infty \tag{4}$$

hold almost surely.

This assumption guarantees that (2) results in a well-posed $H_0^1(D)$-valued solution map $\boldsymbol{y} \mapsto u(\boldsymbol{y},\cdot)$ almost surely.

2.2. Parametric regularity and approximation by polynomials. We will propose an algorithm to solve (2) that utilizes polynomial approximations in the parametric ($\boldsymbol{y}$) variable. We motivate this strategy in this section by recalling established results indicating that best polynomial approximations can achieve high-order convergence rates. The discussion in this section contains specializations of the results in [9, 10, 11].

The essential idea to constructing polynomial approximations is to consider an n-term polynomial approximation of the form

$$u(\boldsymbol{y},\boldsymbol{x}) \approx u_{\Lambda_n}(\boldsymbol{y},\boldsymbol{x}) = \sum_{\lambda\in\Lambda_n} \boldsymbol{y}^{\lambda} c_{\lambda}(\boldsymbol{x}),$$

where c_λ are $H_0^1(D)$-valued coefficients, $\Lambda_n \subset \mathbb{N}_0^N$ is a size-n multi-index set, and we have used multi-index notation:

$$\boldsymbol{y}^{\lambda} \coloneqq \prod_{i=1}^{N} y_i^{\lambda_i}, \qquad \lambda = (\lambda_1,\ldots,\lambda_d) \subset \mathbb{N}_0^N.$$

We will restrict our attention to downward-closed index sets Λ_n, i.e., those such that

$$\lambda \in \Lambda_n \quad \Longrightarrow \quad \nu \in \Lambda_n \;\forall\; \nu \in \mathbb{N}_0^N \text{ satisfying } \nu \leq \lambda,$$

where $\nu \leq \lambda$ is true if the inequality is componentwise true. We also define a polynomial space defined by Λ_n:

$$P_{\Lambda_n} \coloneqq \operatorname{span}\left\{\boldsymbol{y}^{\lambda} \;\middle|\; \lambda \in \Lambda_n\right\}.$$

Under our uniform well-posedness assumption (4), the solution map

$$\Gamma \ni \boldsymbol{y} \mapsto u(\boldsymbol{y},\cdot) \in H_0^1(D)$$

of the parametric PDE (2) is holomorphic in a certain open region around Γ in $\mathbb{C}^N$. This holomorphy property allows one to construct a $\boldsymbol{y}$-Taylor series approximation to truncate this series to a finite number of terms and to estimate the truncation. The result is that one can conclude that there exists a set Λ_n such that

$$\sup_{\boldsymbol{y}\in\Gamma} \|u(\boldsymbol{y},\cdot) - u_{\Lambda_n}(\boldsymbol{y},\cdot)\|_{H_0^1(D)} \leq C\exp(-cn^{1/N}), \tag{5}$$

where C, c are constants that depend on N; see (3.186) in [9]. Thus, we obtain exponential convergence tempered by the curse of dimensionality. An alternative analysis in [11] considers errors via a Stechkin estimate of the truncated tail and allows one to overcome the curse of dimensionality, yielding the estimate

$$\sup_{\boldsymbol{y}\in\Gamma} \|u(\boldsymbol{y}) - u_{\Lambda_n}(\boldsymbol{y})\|_{H_0^1(D))} \leq C(p)n^{-q}, \qquad q := \frac{1}{p} - 1, \tag{6}$$

where $C(p)$ with $p < 1$ depends on the ℓ^p summability of the diffusion coefficient functions:

$$\sum_{j=1}^{N} \|a_j\|^p_{L^\infty(D)}.$$

For more details, see Theorem 1.2 in [11]. The main message is that both results (5) and (6) suggest that polynomial approximations in the parametric variable can be very accurate under reasonable assumptions on the model elliptic problem (2) as n increases. In particular, we note that one can obtain a (possibly high) algebraic rate of decay in the error with respect to the dimension of the polynomial space.

Our main goal in this article is to consider sequential or hierarchical approximation using polynomial approximation in the $\boldsymbol{y}$ variable. To this end, for $i = 1, 2, \ldots$ we will identify a sequence $\{\Lambda_j\}_{j\geq 1}$ of nested index sets, having cardinality $n_j = |\Lambda_j|$, on which to construct a sequence of polynomial approximations. The previous discussion suggests that error decaying like n_j^{-q} might be expected in this case. Our main innovation is the application of novel collocation-based methods for forming exact polynomial quadrature rules on P_{Λ_n}, which results in orders of magnitude savings compared to alternative approaches. The construction of these nodal sets and the identification of the associated multi-index sets will be discussed later.

3. Hierarchical, multilevel approximation.

Section 2.2 discussed the accuracy of best polynomial approximations constructed from polynomial spaces associated with certain multi-index sets Λ_n. Our ultimate goal is to compute expectations (integrals), but accurate approximation with quadrature rules with respect to general sets Λ_n is challenging. Existing methods frequently address this problem with sparse grids [31]. In this paper we use a different strategy: we present a particular method for constructing quadrature schemes that is more flexible than more standard schemes such as low-discrepancy sets and sparse grid constructions. This flexibility comes at the cost of solving a nonconvex optimization problem in the quadrature construction.

Many subsections below discuss ingredients from a multilevel algorithm proposed in [31]. We will blend this algorithm with our distinct quadrature scheme.

3.1. Spatial approximation.

Our spatial discretization is via finite element methods. We consider a sequence of hierarchical finite element spaces U_i (where i is the level index) corresponding to a mesh parameter h_i; we assume $h_i > h_{i+1}$ for all $i \geq 0$. Being hierarchical, the finite element spaces satisfy

$$U_0 \subset U_1 \subset \cdots \subset U_L \subset H_0^1(D)$$

for some fixed maximum level index L. Each space U_i is composed of standard piecewise polynomial shape functions with maximum mesh spacing h_i. Accordingly, $u_i(\boldsymbol{y}) \in U_{h_i}$ denotes the finite element solution computed as the approximation of the true solution $u(\boldsymbol{y})$. For simplicity, in all our examples we construct fine-scale meshes by iterative uniform division of coarse-level meshes. In other words, we consider $h_i = \eta^{-i} h_0$, where h_0 is the size of the coarsest mesh with a fixed refinement integer $\eta > 1$. We also assume that there exist positive constants $C_{\mathcal{H}}$ and α independent of the mesh size h_i, such that for all $i \in \mathbb{N}_0$

$$\mathbb{E}\|u - u_i\|_{H_0^1(D)} \leq C_{\mathcal{H}} h_i^{\alpha}, \tag{7}$$

where, e.g., $\mathbb{E}u \equiv \mathbb{E}[u]$ [7]. The constant α is dependent on the regularity of u, the diffusion coefficient a, and right-hand side data f. These constants $C_{\mathcal{H}}$ and α will play a role in our numerical framework; by assuming the asymptotic behavior (7), we will compute approximations to these constants from finite element solutions on coarser meshes. More detailed analyses of finite element errors and related types of estimation are provided in [32].

3.2. Stochastic approximation. We use polynomial approximation over Γ for approximation in the stochastic $(\boldsymbol{y})$ variable. Specifically, we consider collocation-based methods, and hence we assume continuity in $\boldsymbol{y}$, i.e., $u \in C^0(\Gamma; H_0^1(D))$. This approach is philosophically identical to the approach in [31], but here our approach differs in that we index directly by the number of collocation points M, which later will yield a direct connection to cost of the procedure.

We define an operator $\mathcal{S}_M$, indexed by the number of points M, $\mathcal{S}_M u : C^0(\Gamma) \to L^2_\rho(\Gamma)$, which is an approximation to the mathematical expectation. We leave these interpolation operators unspecified for the moment and will return to them in section 4.

However, we here articulate assumptions that we make on S_M: First, we assume that there is an operator $\mathbb{H}(u) : L^2_\rho(\Gamma; H_0^1) \to \mathbb{R}$ that satisfies

$$\mathbb{H}(u_i) \leq C_{\mathbb{H}} h_0^{\beta}, \qquad \mathbb{H}(u_i - u_{i-1}) \leq C_{\mathbb{H}} h_i^{\beta} \tag{8}$$

for some positive constant $C_{\mathbb{H}}$ and β. This first assumption concerns errors between consecutive spatial solutions as the mesh is refined. Our second assumption is that there exists positive constants $C_{\mathcal{S}}$ and r such that

$$\|\mathbb{E}u - \mathcal{S}_{M_i} u\|_{H_0^1(D)} \leq C_{\mathcal{S}} M_i^{-r} \mathbb{H}(u) \quad \forall u \in L^2_\rho(\Gamma; H_0^1), \tag{9}$$

where similarly to spatial approximation, $\mathcal{S}_{M_L}$ denotes the most accurate stochastic operator and $\mathcal{S}_{M_0}$ is the least accurate one, i.e., $M_0 \leq M_1 \leq \cdots \leq M_L$. We provide analysis in section 4.2.3 motivating that quadrature rules adhering to (9) can be constructed. Note that we do not disallow situations in which the number of samples can be identical in consecutive levels. The key factor in these estimates is the rate r which is associated with the properties of the quadrature rule that will be shown to be higher in the case of our quadrature rule and the hierarchical spatial error. It is noted that the norm $\|u_i\|_{H_0^1(D)}$ is bounded by a constant, whereas $\|u_i - u_{i-1}\|_{H_0^1(D)}$ decays with h_i^{β}, which depends on the mesh size at level i. This feature is the main feature of the multilevel approximation, making it more efficient than a single-level approximation.

The assumptions we have made above are based heavily on the assumptions in [31], but ours are slightly different since we assume explicit dependence on the size of the collocation mesh M.

3.3. Multilevel approximation. The formulation for the multilevel method uses a simple telescoping sum; e.g., for the spatial approximation, the finest level is approximated as

$$u_L = \sum_{i=0}^{L}(u_i - u_{i-1}) \coloneqq \sum_{i=0}^{L} \Delta u_i,$$

where $u_{-1} \coloneqq 0$. The idea for the multilevel approximation is to use a more (less) accurate stochastic approximation, i.e., S_{M_i} with large (small) i, to measure the integral of telescopic discrepancy Δu_i for small (large) i [4, 7, 16]. This strategy leads the authors in [31] to propose the computational estimator:

$$\mathbb{E}u_L \approx \tilde{u}_L \coloneqq \sum_{i=0}^{L} \mathcal{S}_{M_{L-i}}[u_i - u_{i-1}]. \tag{10}$$

This formula uses different levels of the stochastic operator on hierarchical finite element approximations and relates these approximations via the index i. As seen, as the level index increases and the discrepancy between u_i and u_{i-1} decreases, a less accurate stochastic operator $\mathcal{S}_{M_{L-i}}$ is used to compute the expectation of that discrepancy.

4. Multilevel approximation with designed quadrature.

Designed quadrature (DQ) is a recently developed quadrature rule for integration in multiple dimensions that computes moment-matching quadrature rules via optimization [27]. Due to its generic formulation, quadrature rules with various combinations of dimension, order, and weight function on general geometries can be generated.

4.1. DQ. Given an index set Λ with $n = |\Lambda|$, DQ seeks to compute an M-point quadrature rule with nodes $\{y_m\}_{m=1}^{M} \subset \Gamma$ and weights $\{w_m\}_{m=1}^{M}$ satisfying

$$\int_{\Gamma} p(\boldsymbol{y})\rho(\boldsymbol{y})\mathrm{d}\boldsymbol{y} = \sum_{m=1}^{M} w_m p(y_m), \qquad\qquad p \in P_\Lambda.$$

It accomplishes this by directly solving moment-matching conditions subject to geometric constraints on M nodes $\boldsymbol{X} \in \mathbb{R}^{dM}$ and positive constraints on the weights $\boldsymbol{w} \in \mathbb{R}^{M}$,

$$\begin{aligned} \boldsymbol{R}(\boldsymbol{d}) = \boldsymbol{V}(\boldsymbol{Y})\boldsymbol{w} - \boldsymbol{e}_1/\pi_{\boldsymbol{0}} = \boldsymbol{0}&, \\ \boldsymbol{y}_j \in \Gamma, &\quad j = 1, \dots, M, \\ w_j > \boldsymbol{0}, &\quad j = 1, \dots M, \end{aligned} \tag{11}$$

where $\boldsymbol{R}$ is the vector of P_Λ moment-matching residuals, $\boldsymbol{d} = (\boldsymbol{Y}, \boldsymbol{w})$ is the vector of decision variables consisting of nodes $\boldsymbol{Y}$ and $\boldsymbol{w}$, $V(\boldsymbol{Y})$ is the Vandermonde matrix (constructed with $L^2_\rho(\Gamma)$ orthonormal polynomials spanning P_Λ), $\pi_{\boldsymbol{0}}$ is the L^2 norm of the zeroth-order polynomial in P_Λ, and $\boldsymbol{e}_1$ is the cardinal unit vector in the first direction in $\mathbb{R}^n$. In its original form, an index set, geometry, weight function, residual tolerance, and a tentative number of points M are specified in the first step. The node positions are also initialized uniformly at random. The method then uses penalty function methods to transform constrained root finding into an unconstrained minimization problem and subsequently solves a quadratic minimization problem via a Gauss–Newton algorithm. An additional Tikhonov regularization is used to remedy ill-conditioned matrices during Newton's iterations. To find the appropriate number of points in the original DQ approach, we typically start with a larger number of points than needed, which results in $||R|| < \delta$ initially; we then decrease the number of points successively (as much as possible) without making $||R|| > \delta$. If we start with a smaller number of points than required, i.e., $||R|| > \delta$ in the initialization, then we increase the number of points successively until the residual satisfies $||R|| < \delta$. In other words, if the tentative quadrature size at the beginning is small, the number of points is increased successively, and if it is large, it is decreased successively. We refer the readers to [27] for more complete details of this algorithm that we omit here. Specifically, an empirical recommendation on the initial size of the quadrature rule is found based on a similar sparse grid rule. In the following sections, we describe in more detail some novel augmentations of this algorithm that are required to solve our multilevel approximation problem.

4.2. Fixed-M DQ. As described in section 4.1, in the standard use of DQ results, the size of the quadrature rule M is not determined a priori, and only the index set is provided to the algorithm. However, in this paper we use the DQ framework in a slightly different setting: our goal is to develop quadrature rules for a *fixed* M. Hence, we instead start the DQ optimization with a fixed number of points M and an initial index set of initial cardinality $|\Lambda|$. We then gradually decrease the cardinality of the index set until the residual tolerance δ is met (or if the tolerance is already met, we attempt to increase the size of Λ.). In other words, instead of increasing M to achieve quadrature accuracy, we instead choose as large a Λ as possible based on the accuracy afforded by M points. Note that we utilize the same termination criterion: the optimization successfully terminates when Λ is small enough to guarantee $\|\boldsymbol{R}\| \leq \delta$.

Empirically, we observe that the DQ algorithm always successfully terminates, although we cannot prove this. In particular, there is a strong theoretical underpinning for why the DQ should always successfully terminate once the size of Λ is decreased to $|\Lambda| = M$: due to Tchakaloff's theorem, given any multi-index set Λ, there exists a nonnegative quadrature rule of size at most $M = |\Lambda|$ that is exact on P_Λ [12]. Thus, during the DQ procedure, if the size of Λ is decreased to the point where $|\Lambda| = M$, then Tchakaloff's theorem guarantees that there is some nodal set that satisfies the residual termination criterion in DQ. However, this does not guarantee that the algorithm is able to find such nodes (since the optimization is nonconvex). However, in all of our experiments, DQ successfully terminates with $|\Lambda| > M$.

The next sections describe precisely how we determine an initial index set, how we increase/decrease the index set Λ to achieve termination of the algorithm, and what accuracy guarantees are possible from this procedure.

We emphasize that the version of DQ described below can generate quadrature rules for any prescribed nodal count M in any dimension. It does so by allowing an unstructured nodal distribution. In alternative sampling approaches, such as sparse grids that have structured nodal distributions, the number of nodes M is not directly controllable; we will see that the flexibility afforded by DQ allows us to achieve substantial savings in computational cost.

4.2.1. Generating an initial Λ. The DQ algorithm is an iterative optimization algorithm wherein Λ is shrunk or enlarged as iterations progress. We must prescribe how initialization of the index set Λ is accomplished. We make two assumptions that facilitate our initial guess: First we assume some a priori knowledge of a sequence of nested downward-closed multi-index sets $\{\Lambda_p^T\}_{p\geq 0}$, $\Lambda_p^T \subset \Lambda_{p+1}$. We do not perform an approximation with these index sets; they are used in a training procedure to identify index sets for approximation. Two examples of such index sets are the total degree and hyperbolic cross index sets,

$$\Lambda_p^T = \Lambda_p^{\mathrm{TD}} := \left\{\lambda \in \mathbb{N}_0^N \;\middle|\; |\lambda| \leq p\right\}, \quad \Lambda_p^T = \Lambda_p^{\mathrm{HC}} := \left\{\lambda \in \mathbb{N}_0^N \;\middle|\; |\log(\lambda+1)| \leq \log(1+p)\right\}, \tag{12}$$

where both $\lambda+1$ and $\log \lambda$ are elementwise operations. The specification of what kinds of index sets Λ_p^T should be, is problem-dependent; for example, one could choose dimensionally anisotropic versions of the sets above if one has knowledge that some parameters are unimportant. In all our examples, we will choose Λ_p^T as either (isotropic) total degree or hyperbolic cross sets.

The second assumption we make is that the Euclidean parameters $\boldsymbol{y} = (y_1, \ldots, y_N)$ are sorted based on importance with respect to influence on the output u, i.e., that y_j has a stronger impact on u than y_{j+1}.

With these two assumptions, our initial guess proceeds as follows: In an "offline" stage, we use the standard DQ algorithm outlined in section 4.1 with input training index set Λ_p^T to output a set of M_p^T nodes. We repeat this for $p = 0, 1, \ldots, P$, where P is some chosen maximum level. The result is a set of data points $(M_p^T, |\Lambda_p^T|)_{p=0}^P$ that relate the DQ nodal count to the cardinality of the associated index sets. Then, in the "online" stage with M the fixed number of nodes required from the DQ procedure of this section, we can use an interpolation scheme to approximate an associated index set size $|\Lambda| = |\Lambda|(M)$. See Figure 1 for a visual depiction of this procedure.

We can now generate an index set cardinality; to generate the actual index set, we compute the smallest index set Λ_{p^*} whose size is at least $|\Lambda|$, and identify our output index set Λ the first $|\Lambda|$ indices from Λ_{p^*} sorted by total-degree-graded lexicographic ordering. In other words, we identify the number $p^* = p^*(M)$ defined as

$$p^*(M) = \min\left\{p \;\middle|\; M_p^T \geq M\right\}. \tag{13}$$

We next take the indices in the index set $\Lambda_{p^*}^T$, order them via lexicographic ordering graded by total degree, and define Λ as the first $|\Lambda|$ indices in this ordered set. Note that our choice of sorting by lexicographic ordering is motivated by our second assumption above that orders parameters in the vector $\boldsymbol{y}$ by importance.

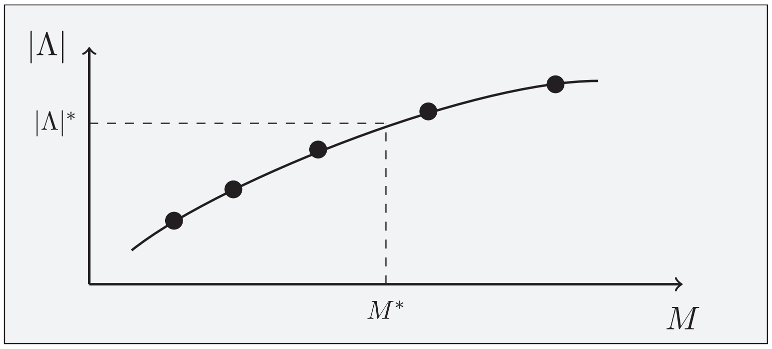

FIG. 1. *Determining the index set cardinality $|\Lambda|^*$ corresponding to a given M^*. Black circles indicate the previously generated data. Unlike sparse grids, DQ can be generated for any $|\Lambda|^*$ due to its flexibility.*

4.2.2. Adding or removing indices from Λ during optimization. At any iteration in the DQ procedure, if the desired moment-matching residual achieves the specified tolerance, we add a multi-index to Λ. If the residual violates the tolerance criterion, we remove a multi-index from the set. The setup described in section 4.2.1 provides the methodology for how to add or remove multi-indices.

The removal of multi-indices proceeds by ordering the indices in Λ via graded lexicographic ordering and removing the last index from this ordered set. To add a multi-index, we recall that $\Lambda^T_{p^*}$, with p^* as determined in (13), identifies a multi-index set (sorted with graded lexicographic ordering) where Λ is the first $|\Lambda|$ entries in $\Lambda^T_{p^*}$. To add a multi-index to Λ, we simply append the next multi-index from the sorted list $\Lambda^T_{p^*}$.

4.2.3. DQ accuracy. Upon successful termination of the DQ algorithm, the tolerance δ provides an error estimate for the quadrature rule.

LEMMA 4.1 (Proposition 2.5 of [27]). *Let Λ be the multi-index set resulting from successful termination of the DQ procedure with M quadrature points. If the stopping criterion uses the tolerance δ, then for any $f \in L^2_\rho(\Gamma) \cap C(\Gamma)$,*

$$\left|\int f(\boldsymbol{y})\rho(\boldsymbol{y})d\boldsymbol{y} - \sum_{m=1}^{M} w_m f(y_m)\right| \le \delta \|f\|_{L^2_\rho(\Gamma)} + (1+\delta) \max_{m=1,\dots,M} |f(y_m) - f^*(y_m)|\,,$$

where f^* is the best $L^2_\rho(\Gamma)$ approximation to f from P_Λ:

$$f^* = \operatorname*{argmin}_{p \in P_\Lambda} \|f - p\|_{L^2_\rho(\Gamma)}\,.$$

Since δ is frequently chosen as a small tolerance, the dominating term in the estimate above is the pointwise discrepancy between f and f^* on the quadrature nodes.

To end this section, we extend Lemma 4.1 to apply to $H^1_0(D)$-valued functions.

LEMMA 4.2. *Let Λ be the multi-index set resulting from successful termination of the DQ procedure with M quadrature points. If the stopping criterion uses the tolerance δ, then for any $f \in L^2_\rho(\Gamma; H^1_0(D)) \cap C(\Gamma; H^1_0(D))$,*

(14)
$$\left\|\int f(\boldsymbol{y})\rho(\boldsymbol{y})d\boldsymbol{y} - \sum_{m=1}^{M} w_m f(y_m)\right\|_{H^1_0(D)} \le \delta \|f\|_{L^2_\rho(\Gamma;H^1_0(D))} + (1+\delta)\|f - f^*\|_{C(\Gamma;H^1_0(D))},$$

where f^* is the $L^2_\rho(\Gamma; H^1_0(D))$-best approximation of f from $H^1_0(D) \otimes P_\Lambda$:

$$f^* = \operatorname*{argmin}_{p \in H^1_0(D)\otimes P_\Lambda} \|f - p\|_{L^2_\rho(\Gamma;H^1_0(D))}.$$

Proof. The inequality

$$\left\|\int f(\boldsymbol{y})\rho(\boldsymbol{y})\boldsymbol{y} - \sum_{m=1}^{M} w_m f(y_m)\right\|_{H^1_0(D)} \le \delta \|f\|_{L^2_\rho(\Gamma;H^1_0(D))} + (1+\delta) \max_{m=1,\dots,M} \|f(y_m) - f^*(y_m)\|_{H^1_0(D)} \tag{15}$$

can be derived in precisely the same way that Proposition 2.5 of [27] is proven for scalar-valued functions, so we present the above inequality without proof. Replacing the maximum over the M quadrature points with the supremum over all Γ results in (14). ∎

Note that if Γ is unbounded, then the bound on the right-hand side of (14) can be infinity. This can be rectified by instead using the result (15), which yields a finite bound, although it can still be large if the quadrature nodes from DQ are very far away from the origin.

The error estimate above provides evidence that one can expect DQ quadrature to achieve an error on the order of

$$\inf_{p \in H_0^1(D) \otimes P_\Lambda} \|f - p\|_{C(\Gamma; H_0^1(D))},$$

which is the best approximation error over the polynomial space P_Λ. This provides a link to the theory of best polynomial approximation discussed in section 2.2. For the root-exponential error estimate in (5), if the Λ in DQ is chosen as the size-n set Λ_n in (5), then application of this estimate in (14) and utilizing the fact that ρ is a probability density yield the DQ accuracy estimate,

$$\left\| \int f(\boldsymbol{y})\rho(\boldsymbol{y})d\boldsymbol{y} - \sum_{m=1}^{M} w_m f(y_m) \right\|_{H_0^1(D)} \le Cn^{-q} + \mathcal{O}(\delta), \tag{16}$$

with C and q as in (6). Thus, for small DQ tolerances, we should observe an algebraic rate of convergence. This in particular can be used for estimating the multilevel stochastic error discussed in section 3.2. Recall from section 4.2 that we observe empirically that DQ always terminates with $M < |\Lambda|$. Taking $\Lambda = \Lambda_n$, (16) becomes

$$\left\| \int f(\boldsymbol{y})\rho(\boldsymbol{y})d\boldsymbol{y} - \sum_{m=1}^{M} w_m f(y_m) \right\|_{H_0^1(D)} = CM^{-q} + \mathcal{O}(\delta), \tag{17}$$

which is a concrete estimate that can be used to certify the assumption (9). Therefore, for small DQ tolerances we expect the convergence rate r to be equal to q.

4.3. Multilevel DQ.

We use DQ to define the stochastic approximation operators $\mathcal{S}_i$ defined in section 3.2. With this definition, the multilevel DQ estimator for $\mathbb{E}u$ is given by (10).

The values $\{M_i\}_{i=0}^{L}$ are assumed known for now, and we will specify them precisely in the next section via a budget-oriented approach. Then given some value M_i, we used the (modified) DQ procedure in section 4.2 to compute a multi-index set Λ_i and DQ nodes and weights $(y_{i,m}, w_{i,m})_{m=1}^{M_i}$, which define our parametric approximation operators,

$$\mathcal{S}_i(f) \coloneqq \sum_{m=1}^{M} w_{i,m} f\left(y_{i,m}\right). \tag{18}$$

Under successful termination of the DQ algorithm with tolerance parameter δ, for each $i \in \{0, \ldots L\}$, these rules satisfy

$$\left| \mathcal{S}_i(p) - \int_\Gamma p(\boldsymbol{y})\rho(\boldsymbol{y})\mathrm{d}\boldsymbol{y} \right| \le \delta \|p\|_{L^2_\rho(\Gamma)}, \qquad p \in P_{\Lambda_i}.$$

Note that the index sets $\{\Lambda_i\}_{i \ge 0}$ are implicitly defined by the quadrature rule. Therefore, we do not know these index sets until the DQ computation is completed.

5. Multilevel budget optimization.

The main purpose of this section is to describe how the number of samples at each level $\{M_i\}_{i=0}^L$, is chosen. Using estimates for the computational cost associated with each level along with the accuracy of the quadrature rules allows us to formulate a convex optimization problem where the M_i appear as the design variables. Furthermore, we provide an analytic solution for this optimization problem.

5.1. Cost and accuracy estimate.

We first provide an estimate for the total multilevel error $\epsilon = \|\mathbb{E}u - \tilde{u}_L\|_{H_0^1(D)}$. We mimic the strategy from [31], which uses the triangle inequality along with estimates for the spatial and stochastic errors,

$$\epsilon := \|\mathbb{E}u - \tilde{u}_L\|_{H_0^1(D)} \leq \|\mathbb{E}u - \mathbb{E}u_L\|_{H_0^1(D)} + \|\mathbb{E}u_L - \tilde{u}_L\|_{H_0^1(D)}, \tag{19}$$

where in what follows we will denote $\|\mathbb{E}u - \mathbb{E}u_L\|_{H_0^1(D)} = \epsilon_{\mathcal{H}}$ and $\|\mathbb{E}u_L - \tilde{u}_L\|_{H_0^1(D)} = \epsilon_{\mathcal{S}}$ as the spatial and stochastic error.

The assumption (7) along with Jensen's inequality can be used to bound the spatial error:

$$\epsilon_{\mathcal{H}} = \|\mathbb{E}u - \mathbb{E}u_L\|_{H_0^1(D)} \leq \mathbb{E}\|u - u_L\|_{H_0^1(D)} \leq C_{\mathcal{H}} h_L^{\alpha}. \tag{20}$$

The stochastic error can also be bounded using the definition of multilevel approximation and another application of the triangle inequality,

$$\begin{aligned}\epsilon_{\mathcal{S}} = \|\mathbb{E}u_L - \tilde{u}_L\|_{H_0^1(D)} &= \left\| \sum_{i=0}^{L} \mathbb{E}(u_i - u_{i-1}) - \mathcal{S}_{M_{L-i}}(u_i - u_{i-1}) \right\|_{H_0^1(D)} \\ &\leq \sum_{i=0}^{L} \|\mathbb{E}(u_i - u_{i-1}) - \mathcal{S}_{M_{L-i}}(u_i - u_{i-1})\|_{H_0^1(D)} \\ &\leq \sum_{i=0}^{L} C_{\mathcal{S}\mathbb{H}} M_{L-i}^{-r} h_i^{\beta},\end{aligned} \tag{21}$$

where we define $C_{\mathcal{S}\mathbb{H}} = C_{\mathcal{S}} C_{\mathbb{H}}$. We will primarily be interested in this latter bound $\epsilon_{\mathcal{S}}$ for the stochastic error, and it will serve as either an objective or constraint in our optimization.

In multilevel approaches at each level, two deterministic PDE solves per each sample are needed. We assume a bound for computing $u_i - u_{i-1}$, i.e., the difference between deterministic solutions associated with h_i and h_{i-1}, on the cost denoted by C_i in the form of

$$C_i \leq C_{\mathcal{C}} h_i^{-\gamma}$$

for some constant $C_{\mathcal{C}}$ that depends on the refinement ratio η; cf. section 3.1. In this work we assume uniform refinement across levels, and we set $C_{\mathcal{C}} = 1$ in our analysis. The rate γ reflects how the computational cost of the spatial solver depends on the mesh parameter h_i. For finite element computations, this generally depends on the spatial dimension and the order of the finite element shape functions. For efficient linear solvers with near-linear complexity, the rate γ is close to the spatial dimension d, and we therefore use $\gamma = 2$ for our two-dimensional examples. The total multilevel cost estimate $\tilde{C}$ is then given by

$$\tilde{C} = \sum_{i=0}^{L} C_{\mathcal{C}} M_{L-i} h_i^{-\gamma}. \tag{22}$$

The total cost $\tilde{C}$ and the stochastic error $\epsilon_{\mathcal{S}}$ will enter into our optimization problem. The number of required levels L cannot be determined from the optimization problem alone unless a ratio between the spatial and stochastic error is assumed [31]. In this paper we first fix the number of levels L and compute the spatial error from the estimate (7) to find the portion of spatial error that contributes to the total error. That is, we define the convex weight w so that $\epsilon_{\mathcal{H}} = w\epsilon$ and $\epsilon_{\mathcal{S}} = (1-w)\epsilon$. The bound (20) can be used to compute a formula for the convex weight w:

$$w = \frac{C_{\mathcal{H}} h_0 \eta^{-L\alpha}}{\epsilon}. \tag{23}$$

This formula for w will be useful for the next section where we formulate the optimization problem that determines the nodal counts $\{M_i\}_{i=0}^{L}$ and show that, for a fixed maximum number of levels L, it is convex; furthermore, we provide an analytical formula for the solution.

5.2. Budget optimization. From the discussion and the formulas in the previous section, we can define a total cost g_c that is a function of $\boldsymbol{M} = [M_0, \ \dots \ M_L]^T$,

$$g_c(\boldsymbol{M}) = \sum_{i=0}^{L} a_i M_i, \qquad a_i = C_{\mathcal{C}} h_0 \eta^{(L-i)\gamma}. \tag{24a}$$

In a similar way, the estimated error g_e of the stochastic component of the multilevel procedure is

$$g_e(\boldsymbol{M}) = \sum_{i=0}^{L} b_i M_i^{-r}, \qquad b_i = C_{\mathcal{S}\mathbb{H}} h_0 \eta^{(L-i)\beta}. \tag{24b}$$

Then, given some desired total error tolerance level ϵ, one optimization problem minimizes the cost g_c subject to achieving the desired error:

$$\min_{\boldsymbol{M}} g_c(\boldsymbol{M}) \quad \text{subject to} \quad g_e(\boldsymbol{M}) = (1-w)\epsilon. \tag{25}$$

Alternatively, given a desired computational cost $\bar{C}$, we could minimize the error:

$$\min_{\boldsymbol{M}} g_e(\boldsymbol{M}) \quad \text{subject to} \quad g_c(\boldsymbol{M}) = \bar{C}. \tag{26}$$

The feasible set for the design variables $\boldsymbol{M}$ in these optimization problems is $\mathbb{R}_+^{L+1}$, where $\mathbb{R}_+ = [0, \infty)$. Thus, the feasible set is convex. The function g_c is convex since it is linear, and since $x \mapsto x^p$ is convex for $p < 0$, then g_e is also a convex function since $r > 0$.

We conclude that both optimization problems (25) and (26) are convex problems and hence both have unique solutions. We next present an analytical formula for the solution to this problem, which specifies the number of quadrature points at each level.

PROPOSITION 5.1. *Let the coefficients a_i, b_i be given by* (24). *With λ^* the dual variable (Lagrange multiplier), the optimal number of points M_i^* given by the solution to the accuracy-constrained optimization problem* (25) *is*

$$M_i^* = \left[\frac{a_i}{r\lambda^* b_i}\right]^{\left(\frac{-1}{r+1}\right)}, \qquad \lambda^* = \left[\frac{(1-w)\epsilon}{\sum_{i=0}^{L} b_i \left(\frac{a_i}{rb_i}\right)^{\left(\frac{r}{r+1}\right)}}\right]^{\left(\frac{r+1}{-r}\right)}. \tag{27}$$

Similarly, the optimal number of points and the dual variable for the cost-constrained optimization problem (26) *are*

$$M_i^* = \left[\frac{\lambda^* a_i}{rb_i}\right]^{\left(\frac{-1}{r+1}\right)}, \qquad \lambda^* = \left[\frac{\bar{C}}{\sum_{i=0}^{L} a_i \left(\frac{a_i}{rb_i}\right)^{\left(\frac{-1}{r+1}\right)}}\right]^{(-r-1)}. \tag{28}$$

Proof. We show the proof for (25). A solution $\boldsymbol{M}^*$ is optimal for (25) if and only if there is a $\lambda^* \in \mathbb{R}$ such that

$$g_e(\boldsymbol{M}^*) = (1-w)\epsilon \qquad \text{(primal feasibility)},$$

$$\nabla g_c(\boldsymbol{M}^*) + \lambda^* \nabla g_e(\boldsymbol{M}^*) = 0 \quad \text{(dual feasibility)}.$$

Solving the equality constrain problem is equivalent to finding a solution to these Karush–Kuhn–Tucker equations in $L+2$ variables $\boldsymbol{M}^*, \lambda^*$. From dual feasibility equations, we find

$$a_i + \lambda^* \left(-rb_i M_i^{-r-1}\right) = 0,$$

$$M_i^{-r-1} = \frac{a_i}{\lambda^* r b_i}, \qquad i = 0, \ldots, L,$$

which yields the formula for M_i^* in (27). Using the above equations in the primal feasibility equations, we find the scalar λ^* from

$$\lambda^{*\left(\frac{-r}{r+1}\right)} \sum_{i=0}^{L} b_i \left(\frac{a_i}{rb_i}\right)^{\left(\frac{r}{r+1}\right)} = (1-w)\epsilon$$

which readily yields the formula for λ^* in (27). The solution for error minimization subject to cost constraints is obtained in a similar manner. ∎

We remark that either of these optimization problems can be solved for a fixed L and w. For application to our multilevel problem, in an outer loop we solve one of these problems, say (25), for many choices of number of levels L, which results in several optimal costs. We then choose the value of L that results in the smallest cost.[1]

Algorithm 1 summarizes our numerical method for multilevel DQ, including the steps in sections 5.1–5.2.

The formulas described above allow us, in the case of the accuracy-constrained optimization (25), to determine the total cost required to achieve a given accuracy ϵ. Analogously, we can compute the achievable accuracy for the cost-constrained optimization (26) given a fixed cost budget $\bar{C}$.

[1]For the problem of optimization over L and w, the interested readers are referred to [22], in which the authors perform a general optimization of the parameters in the multilevel Monte Carlo setting.

Algorithm 1. Multilevel Designed Quadrature (MLDQ)

```
1: Specify parameters $\alpha, \beta, r$ and $C_{\mathcal{H}}, C_{\mathcal{S}\mathbb{H}}$ from an offline analysis
2: Compute $w$ for different choices of level $L$
3: Compute the optimization parameters $a_i$ and $b_i$; cf. (24a) and (24b)
4: Solve the optimization problem for different choices of $L$
5: Select the optimal number of levels based on the smallest cost (or error) and use
   the corresponding rounded half-up $\lceil \frac{\lfloor 2M_i \rfloor}{2} \rceil$ where $\lceil . \rceil$ and $\lfloor . \rfloor$ are ceiling and floor
   functions for number of nodes at each level
6: Design quadrature for $\{M_i\}_{i=0}^{L}$ points in $N$ dimensions using the discussion for
   fixed-$M$ DQ; cf. section 4.2
7: Estimate the actual cost and error (with respect to the most accurate parametric
   solution, i.e., finest mesh and largest size quadrature rule)
```

Corollary 5.1. *Consider the setup of Proposition* 5.1*. Then given a computational budget (cost), $\bar{C}$, the MLDQ procedure, Algorithm* 1*, utilizing the cost-constrained optimization* (26) *and its associated solution* (28)*, commits an error bounded as*

$$\|\mathbb{E}u - \tilde{u}_L\|_{H_0^1(D)} \le \bar{C}^{-r} \left(h_0 C_{\mathcal{S}\mathbb{H}C}\right)^{r+1} \left[\frac{\nu^{L+1}-1}{\nu-1}\right]^{r+1}, \tag{29}$$

where

$$C_{\mathcal{S}\mathbb{H}C} := C_{\mathcal{S}\mathbb{H}}^{\frac{1}{r+1}} C_C^{\frac{r}{r+1}}, \qquad \nu := \eta^{\left[\frac{\gamma r+\beta}{r+1}\right]} > 1.$$

In addition, given a desired accuracy ϵ in the MLDQ procedure, then Algorithm 1*, utilizing the accuracy-constrained optimization* (25) *and its associated solution* (27)*, incurs the total cost,*

$$g_c(\boldsymbol{M}^*) = \epsilon_{\mathcal{S}}^{-1/r} \left(h_0 C_{\mathcal{S}\mathbb{H}C}\right)^{(r+1)/r} \left[\frac{\nu^{L+1}-1}{\nu-1}\right]^{(r+1)/r}. \tag{30}$$

Proof. The result (29) is deducible from a direct manipulation of $g_e(\boldsymbol{M}^*)$ with $\boldsymbol{M}^*$ as in (28). To prove (30), a similar manipulation of $g_c(\boldsymbol{M}^*)$ with $\boldsymbol{M}^*$ as in (27) results in the equality

$$g_c(\boldsymbol{M}^*) = [(1-w)\epsilon]^{-1/r} \left(h_0 C_{\mathcal{S}\mathbb{H}C}\right)^{(r+1)/r} \left[\frac{\nu^{L+1}-1}{\nu-1}\right]^{(r+1)/r}.$$

Replacing $(1-w)\epsilon$ with $\epsilon_{\mathcal{S}}$ results in (30). Numerical verification of these estimates is provided in section 6. ∎

According to (30), the cost scales like $\epsilon^{-1/r}$ and hence large r, increases the efficiency of the method. The error bound (29) scales like $\bar{C}^{-r}$, so that increased computational investment (increased $\bar{C}$) results in rate-r algebraic error decay, showing that this method is efficient if r is large. In both cases, large r is beneficial, which corresponds to designing accurate quadrature rules.

Remark 5.1. We emphasize that there is a *training* (or offline) cost to the DQ approach in this paper. This cost is pertinent to estimating the initial size of the index set, which involves designing quadrature for some known index sets as described in section 4.2.1. We also note that using the MLDQ approach involves another offline cost. In particular, the key optimization parameters in this approach, i.e., $\alpha, \beta, r, C_{\mathcal{H}}, C_{\mathcal{S}\mathbb{H}}$, are obtained from an offline analysis, which considers multiple spatial meshes as well as quadrature rules to evaluate the convergence rate for spatial and stochastic approximations. Details of computation for these parameters are provided in section 6.1.1.

6. Numerical examples.

6.1. Linear elasticity. We consider a linear elastic structure shown in Figure 2 to compute the statistics of the displacement within the domain. To find the displacement, we solve the equations governing linear elasticity $\nabla \cdot \boldsymbol{\sigma} + f = 0$, where $\boldsymbol{\sigma}$ is the stress tensor and $\nabla\cdot$ is the divergence operator. This results in a linear elliptic PDE for the displacement u that has the form (2) and, with the notation in this section, reads $-\nabla \cdot (E(\boldsymbol{y}, \boldsymbol{x})\nabla u) = f$, where $E(\boldsymbol{y}, \boldsymbol{x})$ is the elastic modulus that is considered as a random field whose centered logarithm is parameterized by a Karhunen–Loéve expansion,

$$E(\boldsymbol{y}, \boldsymbol{x}) = E_0(\boldsymbol{x}) + \exp\left[\sum_{i=1}^{N} \sqrt{\lambda_i} E_i(\boldsymbol{x}) y_i(\omega)\right]. \tag{31}$$

Note that this example does not fall into the framework where the theory of section 2.1 applies. Nevertheless, the multilevel DQ algorithm can be applied to this kind of parameterized problem.

In this example the random variables are uniform $y_i \sim U[-1, 1]$, and the eigenvalues λ_i and basis functions E_i are taken as those for an exponential kernel $C(\boldsymbol{x}, \boldsymbol{x}')$ on $D = [0, 1]^2$. For the 1D case with $D = [0, 1]$ we have [31]

$$C(\boldsymbol{x}, \boldsymbol{x}') = \exp(-\|\boldsymbol{x} - \boldsymbol{x}'\|_1), \quad \lambda_i^{1D} = \frac{2}{\kappa_i^2 + 1}, \quad b_i^{1D} = A_i(\sin(\kappa_i \boldsymbol{x}) + w_i \cos(\kappa_i \boldsymbol{x})),$$

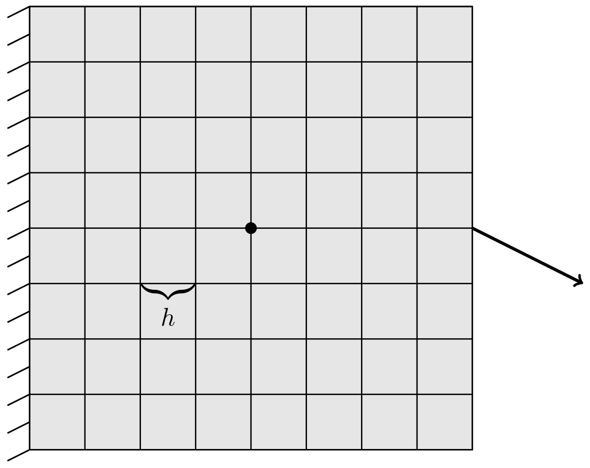

FIG. 2. *Linear elastic structure with varying size mesh. The quantity of interest is the displacement of the point indicated with the black dot.*

where κ_i are the positive ordered solutions to

$$\tan(\kappa) = \frac{2\kappa}{\kappa^2 - 1}.$$

We make the approximation $\kappa_i \approx i\pi$, which is the asymptotic behavior of these solutions. Using this approximation for κ_i, we then consider a two-dimensional domain $D = [0,1]^2$ with eigenpairs

$$\lambda_i = \lambda^{1D}_{i_j}\lambda^{1D}_{i_k}, \quad E_i = b^{1D}_{i_j} \otimes b^{1D}_{i_k} \quad \text{for some} \quad i_j, i_k \in \mathbb{N}, \tag{32}$$

where $\bigotimes$ is the tensor product. We assume a constant centered elastic modulus $E_0 = 3$ and consider the norm of two-dimensional (2D) displacement ($u = \sqrt{u_x^2 + u_y^2}$) at a point in the middle of the domain, $q = u(\boldsymbol{x}_0)$, with $\boldsymbol{x}_0$ indicated by the position of the black dot in the figure, as the quantity of interest. We also consider total order polynomial index sets $\Lambda^T_j = \Lambda^{\text{TD}}_j$ for performing our DQ training as described in section 4.2, resulting in M^T_j nodes for each quadrature rule.

In this example we will consider $N = 2$ and $N = 10$ dimensional problems. The selection of bases is based on the magnitude of the eigenvalues in two dimensions; cf. (32). We compute a large number of 2D eigenvalues and sort them based on their magnitude; we then choose first $N = 2$ and $N = 10$ eigenvalues and their corresponding eigenvectors to include in the Karhunen–Loéve expansion; cf. (31).

The finite element analysis of the linear elastic structure shown in Figure 2 is performed using part of the standard code for topology optimization [2]. Assuming a unit length for the dimensions of the square structure and providing the mesh size h, the analysis is performed with $1/h \times 1/h$ bilinear square isotropic finite elements.

6.1.1. N=2 dimensions. In an offline step, we approximate the constants α, β, and r. To that end we consider a hierarchy of meshes $h_0 = \frac{1}{4}$ and $h_i = h_0 2^{-i}$ for $i = 1, \dots, 7$. In order to estimate the desired constants, we utilize a very accurate parametric DQ rule $\mathcal{S}_{M_\uparrow}$, which integrates on the polynomial space $\Lambda = \Lambda^{\text{TD}}_{17}$, resulting in $M_\uparrow = 93$ nodes. We will also need a relatively inaccurate quadrature DQ rule $\mathcal{S}_{M_\downarrow}$, which uses $M_\downarrow = 3$ quadrature points associated with the index set Λ^{TD}_2.

We estimate α by varying mesh size (the index i) and using a fixed high-order quadrature rule, denoted by $\mathcal{S}_{M_\uparrow}$. The parameter β is obtained by varying the mesh size in the difference of $u_{h_i} - u_{h_{i+1}}$ using the fixed accurate rule $\mathcal{S}_{M_\uparrow}$ and relatively inaccurate rule $\mathcal{S}_{M_\downarrow}$ order quadrature, and parameter r is obtained by varying the quadrature order. The following steps summarize the estimation of the parameters $(\alpha, \beta, r, C_{\mathcal{H}}, C_{\mathcal{S}\mathbb{H}})$:

- $\alpha, C_{\mathcal{H}}$: These constants are defined by (7), so we inspect the slope of the graph of $\sqrt{S_{M_\uparrow}\left(u_7(\boldsymbol{x}_0) - u_i(\boldsymbol{x}_0)\right)^2}$ versus h_i for $i = 0, \dots, 6$ on a log scale for both axes to obtain r. The constant $C_{\mathcal{H}}$ is also obtained as the multiplicative coefficient associated with the fitted line. See Figure 3, left.
- β: Combining (9) and (8) and knowing $C_{\mathcal{S}\mathbb{H}} = C_{\mathcal{S}}C_{\mathbb{H}}$, we have

$$\left|\mathbb{E}(u_i(\boldsymbol{x}_0) - u_{i-1}(\boldsymbol{x}_0)) - \mathcal{S}_{M^T_j}(u_i(\boldsymbol{x}_0) - u_{i-1}(\boldsymbol{x}_0))\right| \leq C_{\mathcal{S}\mathbb{H}}\left(M^T_j\right)^{-r} h_i^{\beta}. \tag{33}$$

We use our refined quadrature rule $\mathcal{S}_{M_\uparrow}$ to emulate the expectation $\mathbb{E}$, and for identification of β replace $\mathcal{S}_{M^T_j} = \mathcal{S}_{M_\downarrow}$. Then, by varying the index i, we expect behavior of the form

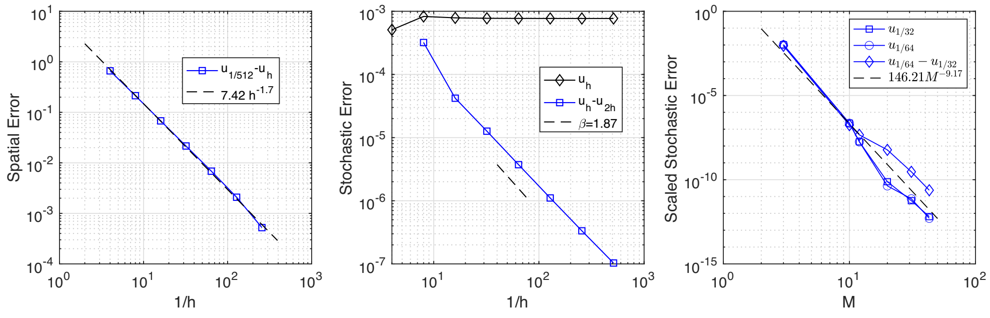

FIG. 3. *Identification of constants* $\alpha = 1.7$, $C_{\mathcal{H}} = 7.42$, $\beta = 1.87$, $C_{\mathcal{S}\mathbb{H}} = 146.21$, *and* $r = 9.17$ *for* $N = 2$ *stochastic variables.*

$$\left|\mathcal{S}_{M^{\uparrow}}(u_i(\boldsymbol{x}_0) - u_{i-1}(\boldsymbol{x}_0)) - \mathcal{S}_{M_{\downarrow}}(u_i(\boldsymbol{x}_0) - u_{i-1}(\boldsymbol{x}_0))\right| \lesssim h_i^{\beta},$$

from which we can compute an estimate of β by plotting the left-hand side versus h_i on a logarithmic plot. This is done in Figure 3, center (blue curve).

- $r, C_{\mathcal{S}\mathbb{H}}$: Another experiment from the estimate (33) is to fix i, resulting in the estimate

$$\frac{1}{h_i^{\beta}}\left|\mathbb{E}(u_i(\boldsymbol{x}_0) - u_{i-1}(\boldsymbol{x}_0)) - \mathcal{S}_{M_j^T}(u_i(\boldsymbol{x}_0) - u_{i-1}(\boldsymbol{x}_0))\right| \leq C_{\mathcal{S}\mathbb{H}}(M_j^T)^{-r}.$$

Similarly, using the first assumption in (8) (associated with h_0) yields

$$\frac{1}{h_0^{\beta}}\left|\mathbb{E}u_i(\boldsymbol{x}_0) - \mathcal{S}_{M_j^T}u_i(\boldsymbol{x}_0)\right| \leq C_{\mathcal{S}\mathbb{H}}(M_j^T)^{-r}.$$

Since β is already computed, the above estimates result in a strategy to approximate r and $C_{\mathcal{S}\mathbb{H}}$ by varying j. These experiments are run for two values of i in Figure 3, right. In addition, we can verify the assumption in (8) that $\mathbb{H}(u_i)$ is bounded independent of i, which is verified in Figure 3, center (black curve).

Figure 3 is an experimental summary of the above parameter estimation and also lists the values of the computed constants for this example. To find the dashed lines in the plots (regression lines), we perform a least square regression on the data points in each plot. The following spatial approximations are used in the first two plots (left and middle panes):

- Left pane: $u_h \in \{u_{\frac{1}{4}}, u_{\frac{1}{8}}, \ldots, u_{\frac{1}{256}}\}$.
- Middle pane (black curve): $u_h \in \{u_{\frac{1}{4}}, u_{\frac{1}{8}}, \ldots, u_{\frac{1}{512}}\}$.
- Middle pane (blue curve): $u_h - u_{2h} \in \{u_{\frac{1}{8}} - u_{\frac{1}{4}}, u_{\frac{1}{16}} - u_{\frac{1}{8}}, \ldots, u_{\frac{1}{512}} - u_{\frac{1}{256}}\}$.

We now investigate the solution of the budget optimization problem introduced in section 5.2. We solve the optimization problem numerically with built-in MATLAB functions with 100 randomized initial guesses. We fix $L = 4$ and $\epsilon = 0.01$, which yields $w = 0.63$. The objective and constraint functions for one of the numerical solutions (with random initial guess) are $\tilde{C} = 672.7702$ and $g_e - (1-w)\epsilon = 1.82 \times 10^{-09}$, respectively, whereas the exact analytical values are $\tilde{C} = 672.7701$ and $g_e - (1-w)\epsilon = 4.33 \times 10^{-19}$. It is apparent that the analytical values are in satisfactory agreement with the numerical optimization values. It is also worthwhile to scrutinize the estimate values of the computational cost in this example. This particular value is obtained by using (22) and knowing the number of nodes M_{L-i} at each level as well as the mesh sizes h_i. Since $L = 4$ levels are used in this example, the computational cost involves the mesh sizes $\{h_0 = \frac{1}{4}, h_1 = \frac{1}{8}, \ldots, h_4 = \frac{1}{64}\}$. The mesh sizes are fixed throughout the examples, but the number of points for different levels in each example is different which gives rise to different estimates for computational cost.

With the solution to the optimization problem, we can investigate the multilevel cost associated with different levels L for each fixed error tolerance ϵ, which is shown in Figure 4, left. We choose 20 equispaced values of ϵ on the interval $[10^{-3}, 10^{-1}]$, which yields 20 curves that are plotted. It is evident that for each ϵ, the smallest L yields the minimum cost, but for some values of ϵ, one cannot achieve the required tolerance without $L > 1$. On the right pane of Figure 4, we show the the smallest total cost that can be achieved for the prescribed tolerance. It is noted that in all cases (of prescribed ϵ), the smallest cost is associated with the smallest level. This log-log figure also shows that as the prescribed error increases, the theoretical cost decreases almost linearly. In the case of actual cost versus error, however, the growth of cost as a function of error is not always linear (in the log-log scale); e.g., see Figure 6.

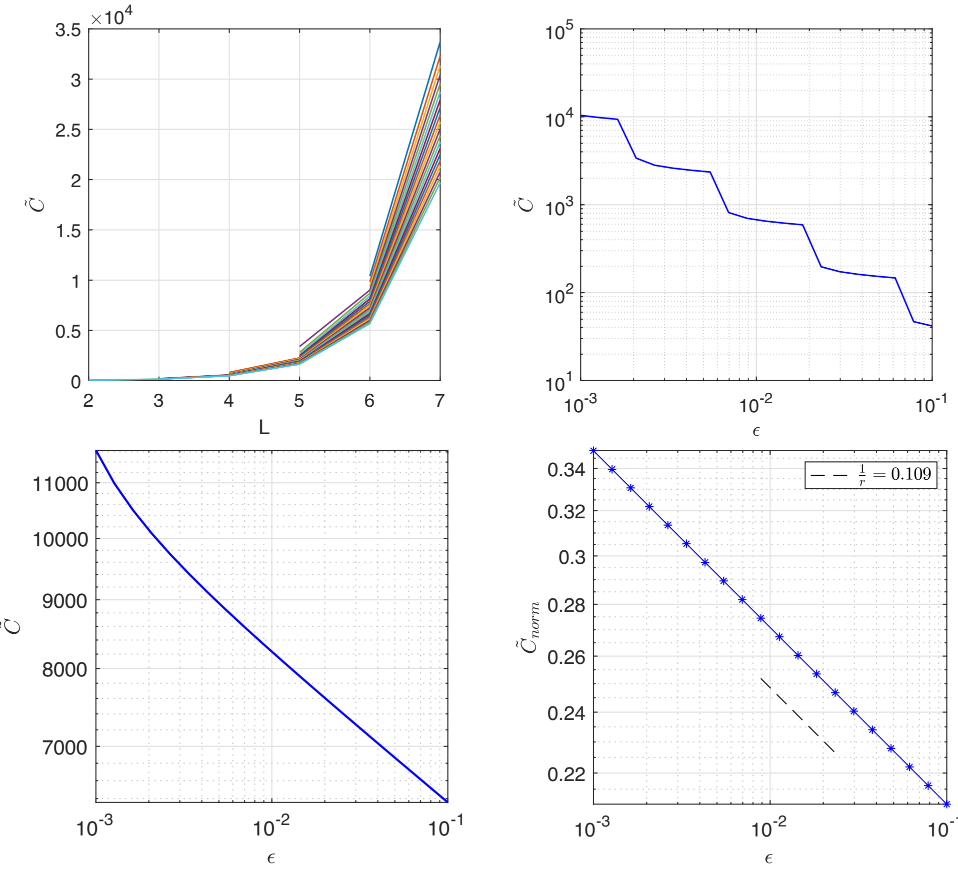

FIG. 4. *Optimal cost with respect to the number of levels (top-left), optimal cost versus total error for varying L (top-right), cost versus error for a fixed $L = 6$ (bottom-left), and verification of Corollary 5.1 with the estimated rate of convergence $1/r = 0.109$ matching the rate $r = 9.17$ in Figure 3 (bottom-right).*

To verify Corollary 5.1 we perform another experiment: we fix L to a large enough value ($L = 6$), which yields $w \in [0, 1]$ for all choices of ϵ (cf. (23)) and compute the theoretical estimates for cost from (24a) and (27). We then normalize this theoretical cost by dividing by the second and third multiplicative terms in the right-hand side of (30) to find the normalized cost $\tilde{C}_{norm}$, i.e.,

$$\tilde{C}_{norm} = \frac{g_c(\boldsymbol{M}^*)}{(h_0 C_{\mathbb{SH}C})^{(r+1)/r} \left[\frac{\nu^{L+1}-1}{\nu-1}\right]^{(r+1)/r}}. \tag{34}$$

The bottom left pane in Figure 4 shows the theoretical cost with fixed L against the error, and the bottom right pane shows the normalized cost $\tilde{C}_{norm}$ versus error, which is a straight line with the negative slope $1/r$ in the log-log scale verifying the result in Corollary 5.1.

For the case $L = 6$ and $\epsilon = 0.001$, which corresponds to $\tilde{C} = 1.038 \times 10^4$, the optimal number of points is $\boldsymbol{M} = \{1.36, 1.78, 2.32, 3.02, 3.94, 5.14, 6.85\}$. We round these numbers to $\boldsymbol{M} = \{1, 2, 2, 3, 4, 5, 7\}$, which requires the generation of $(1, 2, 3, 4, 5, 7)$-point rules in $N = 2$ dimensions. To generate the DQ training data, we generate quadrature rules associated with total degree sets, $\Lambda = \Lambda_p^{\mathrm{TD}}$, and fit a $|\Lambda|$ versus M curve, which is shown in Figure 5, left (black curve). From this, we can generate initial guesses for the index set Λ and perform fixed-M DQ, which results in actual M-point quadrature rules for any value of M. The cardinality of these actual DQ quadrature rules is shown in Figure 5, left (blue curve). An example of an $M = 5$-point rule is shown in the right pane of Figure 5, having a DQ index set size of $|\Lambda| = 14$ and a DQ residual of $\delta = \|\boldsymbol{R}\| = 4.72 \times 10^{-12}$.

The actual error can now be computed for different prescribed values of error tolerances ϵ using the multilevel DQ strategy. Figure 6 shows the actual error using the MLDQ strategy and compares it with a stochastic approximation on just the finest mesh. The true solution is a refined estimator computed with $\mathcal{S}_{M_\uparrow}$ and $h = h_7$. It is apparent that the MLDQ achieves the same magnitude of error at a much lower cost compared to the single level of fine mesh approximation.

We note that MLDQ always outperforms DQ and achieves higher accuracies with less computational effort. However, in situations in which the maximum value of L is restricted, the performance of MLDQ approaches that of DQ, as we observe in Figure 6

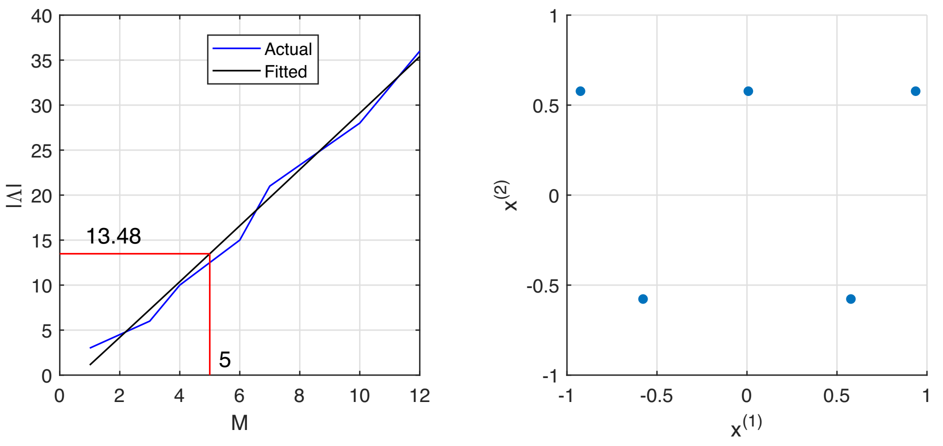

FIG. 5. *Finding the size of index set from the number of points (left); DQ for $M = 5$, $N = 2$.*

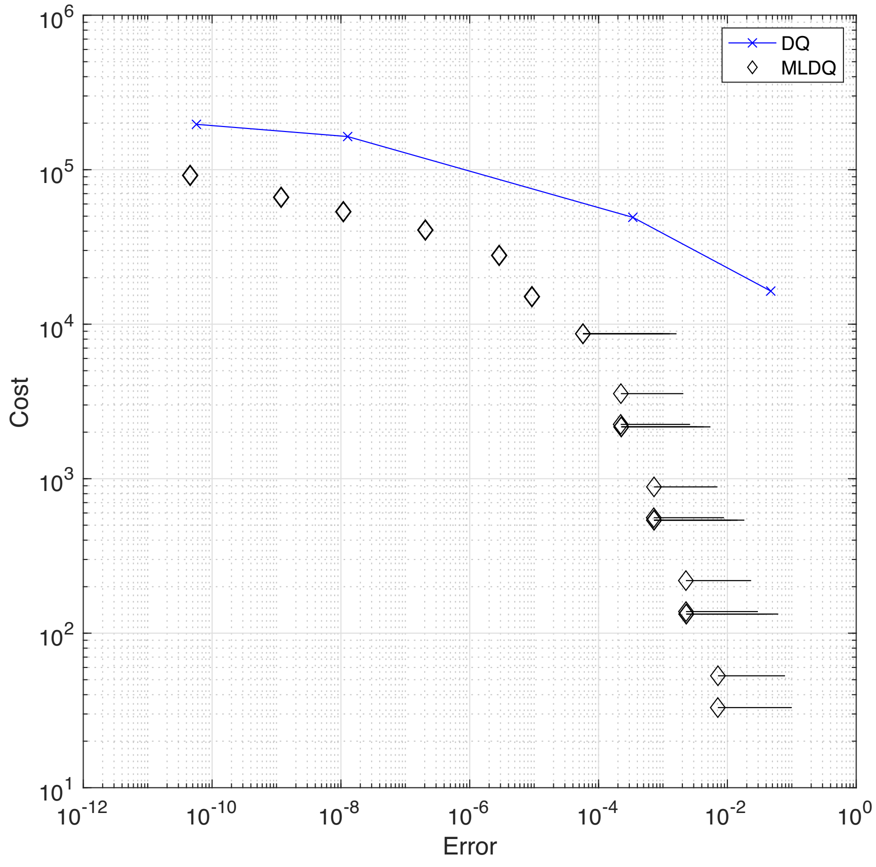

FIG. 6. *Actual cost versus error for MLDQ and DQ with the finest mesh $h = 1/512$. The black horizontal lines connects the actual error to the prescribed ϵ in the optimization problem.*

when the error tolerance is decreased below 10^{-5} (in this experiment the maximum L allowed is 6). The same behavior occurs when comparing standard Monte Carlo with a multilevel Monte Carlo method with the same level restriction and is expected: in this case the weight w changes so as to address all computational efforts towards reducing the statistical error (since the discretization error is fixed for fixed L), and many more samples/points are needed on all levels. The consequence is that in this scenario multilevel methods effectively start acting as single-level methods.

Remark 6.1. As mentioned previously, the error achieved by the MLDQ procedure with a prescribed accuracy (as described in section 5) is influenced by the spatial approximation error. In theory the method can achieve any small errors, as shown in Table 1. In practice, however, depending on the convergence rate of the spatial approximation, i.e., α (cf. (7)) and the number of levels L, different estimates are obtained for w which indicates the weight (or contribution) of spatial error in the total error. The weight w must be within a range $[0, 1]$ in our analysis. For smaller ϵ's and a fixed spatial convergence rate, one should increase L to achieve $w \in [0, 1]$, but this is not always possible in practice. To investigate what happens in this case, we fixed $L = 6$ in Figure 6.

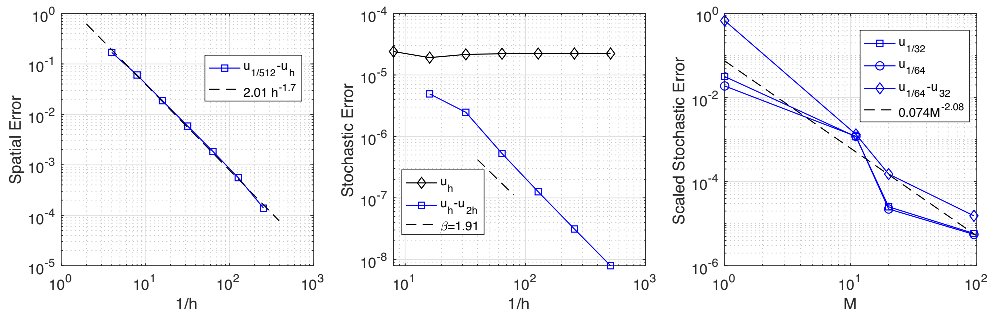

FIG. 7. *Identification of constants $\alpha = 1.7$, $C_{\mathcal{H}} = 2.01$, $\beta = 1.91$, $C_{\mathcal{S}\mathbb{H}} = 0.074$, and $r = 2.08$ for $N = 10$ stochastic variables.*

6.1.2. N=10 dimensions. We use the same problem setting and setup but with a higher number of parameters, $N = 10$. Similarly to Figure 3, we find the constants $(\alpha, \beta, r, C_{\mathcal{H}}, C_{\mathcal{S}\mathbb{H}})$ that we use in the optimization problem; cf. Figure 7. As in the previous example, we use the following spatial approximations in the first two panes in this figure:

- Left pane: $u_h \in \{u_{\frac{1}{4}}, u_{\frac{1}{8}}, \ldots, u_{\frac{1}{256}}\}$.
- Middle pane (black curve): $u_h \in \{u_{\frac{1}{8}}, \ldots, u_{\frac{1}{512}}\}$.
- Middle pane (blue curve): $u_h - u_{2h} \in \{u_{\frac{1}{16}} - u_{\frac{1}{8}}, \ldots, u_{\frac{1}{512}} - u_{\frac{1}{256}}\}$.

We observe that the rate r of error decay with respect to the number of samples is significantly smaller compared to the previous example. We investigate the difference between the two optimization results in section 5.2: we can minimize the cost $\tilde{C}$ subject to an accuracy tolerance certification ϵ (as done in the previous example), or we can minimize the accuracy ϵ subject to a given cost budget $\tilde{C}$. The result of these optimizations is shown in Figure 8. Figure 8 also shows the results associated with the verification of Corollary 5.1 for both optimization scenarios. The slopes of regression lines are obtained as $1/r = 0.48$ and $r = 2.08$, as expected from Corollary 5.1. Note that the normalized error ϵ_{norm} is obtained in a similar manner as $\tilde{C}_{norm}$ (cf. (34)) by normalizing $g_e(\boldsymbol{M}^*)$ with the second and third terms in the right-hand side of (29), i.e.,

$$e_{norm} = \frac{g_e(\boldsymbol{M}^*)}{(h_0 C_{S\mathbb{H}C})^{r+1} \left[\frac{\nu^{L+1}-1}{\nu-1}\right]^{r+1}}. \tag{35}$$

We compute the actual error after finding the optimal number of points for both cases of optimization in Figure 8. We computed the error against the most accurate solution, which is obtained with the finest mesh and sparse grid nodes for a high order. In particular, we use an $M = 5281$ sparse grid rule to find the true solution and use a smaller number of sparse grid nodes to obtain the curves associated with the sparse grid; cf. Figure 9. The sparse grid nodes are taken from [24, 25]. In particular, we use the 10-dimensional rule which considers the Kronrod–Patterson nested rule as the underlying univariate quadrature. This particular type of sparse grid is already computed and tabulated in [24] and is denoted by "KPU" (i.e., Kronrod–Patterson rule in conjunction with uniform weight). Results associated with sparse grids in Figure 9 (red curves) are obtained with $(1, 2, 201, 1201)$-point rules, which correspond to total order polynomials $(1, 3, 5, 7)$. The sparse grid nodes yield higher costs for

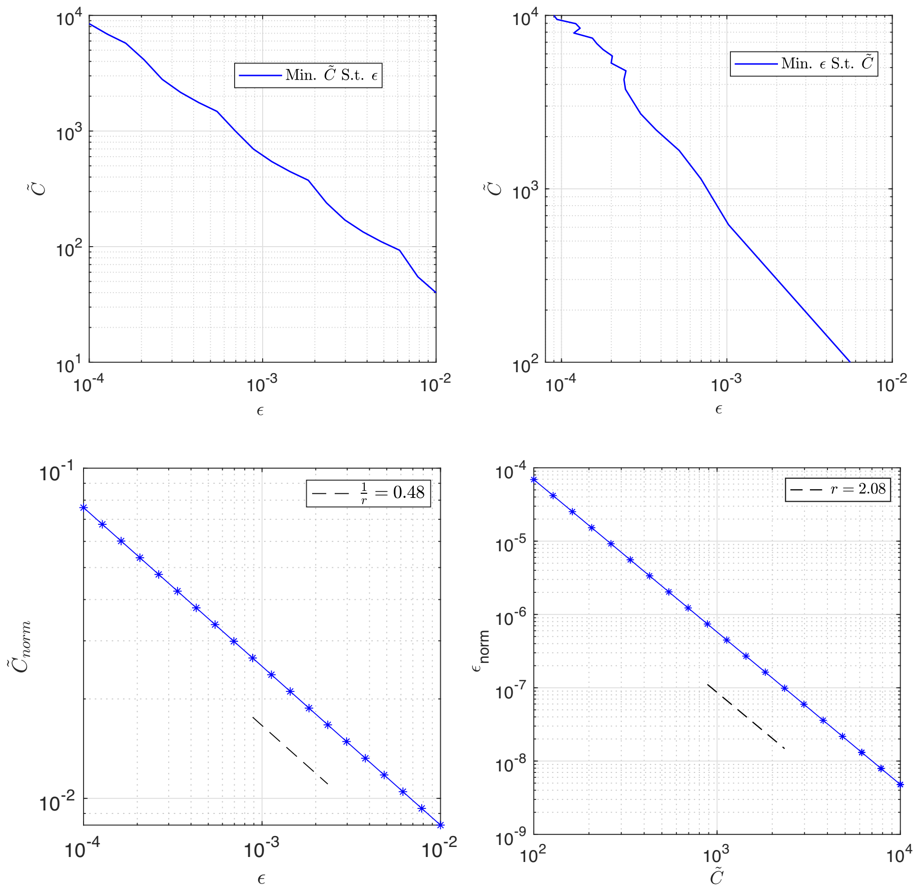

FIG. 8. *Cost versus error for two optimization problems: minimize $\tilde{C}$ subject to ϵ (top-left); minimize ϵ subject to $\tilde{C}$ (top-right). In the left figure,* 20 *values for error constraint are considered in the range $\epsilon \in [10^{-4}, 10^{-2}]$. Similarly, in the right figure,* 20 *values for cost constraint are considered in the range $\tilde{C} \in [10^2, 10^4]$. Verification of Corollary* 5.1 *with the estimated rate of convergence $1/r = 0.48$ and $r = 2.08$ for both scenarios of cost and error minimization subject to error and cost constraints (bottom row). The estimated rate of convergence matches the rate $r = 2.08$ in Figure* 7.

the same error or larger error for the same cost. It is also again observed that the MLDQ strategy is more efficient than the single-level DQ (with the finest mesh) and the sparse grid. Another observation in these results associated with the higher dimension example is the sublinear growth of actual cost as a function of error in the log-log scale. The same behavior is also observed in the 100-dimension example in the next section; see Figure 13.

Finally in this example, to better demonstrate the effectiveness of the MLDQ approach over other multilevel approaches that consider other quadrature rules, e.g., sparse grids, we compute the number of samples needed for MLDQ at each level and compare them with the number of points that can be selected from sparse grids. Results for different prescribed accuracies are listed in Table 1. As the number of levels increases, more points are needed, and that adversely affects the multilevel stochastic collocation (MLSC) approach, which uses sparse grids for stochastic computations.

To find the number of points associated with MLSC in Table 1, we perform the cost minimization subject to error constraint to find the number of points for the

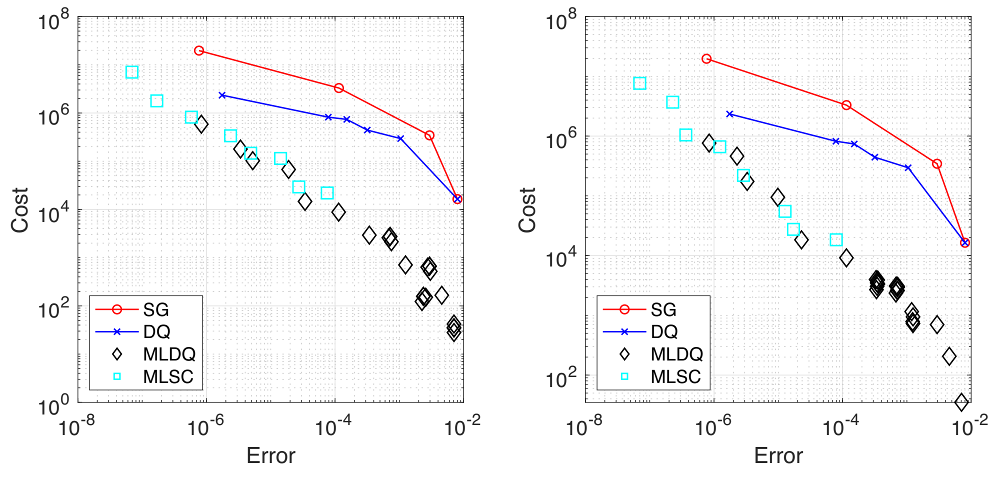

FIG. 9. *Actual cost versus error for two optimization problems: minimize $\tilde{C}$ subject to ϵ (left); minimize ϵ subject to $\tilde{C}$ (right). Results are compared with the finest mesh and different levels of quadrature accuracy for both DQ and sparse grids (SG).*

TABLE 1
Number of samples needed at each level for various prescribed accuracies for MLDQ and MLSC.

MLDQ, i	0	1	2	3	4	5	6	7	8	9	10	11
$\epsilon = 10^{-2}$	3	1	1	1								
$\epsilon = 10^{-4}$	85	33	14	6	2	1	1	1				
$\epsilon = 10^{-6}$	1988	767	318	132	55	23	9	4	2	1	1	1
MLSC, i	0	1	2	3	4	5	6	7	8	9	10	11
$\epsilon = 10^{-2}$	21	1	1	1								
$\epsilon = 10^{-4}$	201	201	21	21	21	1	1	1				
$\epsilon = 10^{-6}$	5281	1201	1201	201	201	201	21	21	21	1	1	1

MLDQ approach; cf. (27). We then round up this number to the next available sparse grids rule size from [24]. In this way the number of points is expectedly much larger than the DQ rule; however, the larger number of points can potentially yield more accuracy. To investigate cost versus accuracy for the MLSC approach, we compute the actual cost and error in the same manner we have done for MLDQ. To this end, we consider the number of points associated with the scenario with least error (and highest cost) and gradually add to the number of points associated with the finest mesh similarly to what we explained in the first numerical example and the results in Figure 6. There is a major difference in the way we compute the cost for MLSC. MLSC uses sparse grid points that are nested across levels. Therefore, the actual cost of computing $u_i - u_{i-1}$ on an effectively nested set of points; e.g., the 21-point rule does not consider the cost of computing u_{i-1} since those computations are already performed for the coarser level with a larger set of nodes (parent nodes), e.g., 201-point rule. The actual cost versus error data points for MLSC are also included in Figure 9. We observe that the rate of growth in cost with respect to error decrease is similar for both MLDQ and MLSC. However, the data points in these plots show smaller error and higher cost for the MLSC approach, which uses a larger number of points at the highest level.

6.2. Heat diffusion equation. In this example we consider the problem of heat distribution that is described with a linear elliptic PDE in the form of (2); however, the diffusion coefficient a does *not* have the form (3), and instead its logarithm is affine

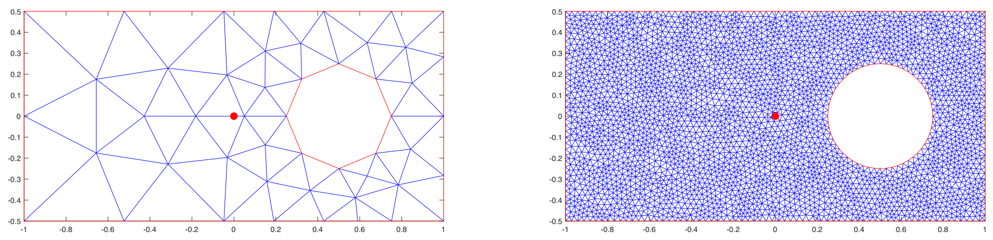

FIG. 10. *The finite element discretization for the heat diffusion equation: coarsest mesh (left), finest mesh (right). The quantity of interest is the temperature at the point indicated with the red circle.*

in the parameters $\boldsymbol{y}$, which can be unbounded, e.g., modeled as Gaussian random variables. Problems with such types of diffusion coefficients are outside the scope of the theory we have presented in this manuscript (since, e.g., the upper inequality in (4) is violated), but convergence for similar kinds of schemes on such classes of PDEs has been studied, e.g., [7]. The spatial domain and its finite element discretization is shown in Figure 10. We note in particular that due to refinement around the enclosed circle, these meshes are not nested, and so the finite element spaces U_i introduced in section 3.1 are not nested. In this case, we cannot directly use the telescoping setup outlined in that section. Instead of defining interpolation operators between spaces (as one remedy might be), we define the quantity of interest for this problem as the solution computed at the location of the red dot in Figure 10, e.g., for some fixed $\boldsymbol{x}^* \in D$, $q_i \coloneqq u_i(\boldsymbol{x}^*)$ for $i = 0, \dots, L$, is a scalar-valued quantity of interest. In this case, all the formalism of section 3.1 can be applied by replacing $u_i \in U_i$ with $q_i \in \mathbb{R}$. The spatial *error* still manifests in the error committed by q_i relative to the exact scalar value. Thus, we must still compute the spatial constants β and $C_{\mathbb{H}}$ in section 3.2. Finally, we note that assigning $q_i = u_i(\boldsymbol{x}^*)$ requires that we assume enough regularity for u_i so that point evaluation is a bounded functional; in two dimensions utilizing Sobolev embedding, we therefore require that $u \in H^s(D)$ for some $s > \frac{2}{2} = 1$. This in turn requires f be a bit smoother than H^{-1} in order to ensure that $u \in H^{1+\epsilon}(D)$.

We assume a constant forcing term $f = 1$ (which is smooth enough so that $u_i(\boldsymbol{x}^*)$ is well defined) and constant Dirichlet boundary conditions on the edges of rectangle, $u = 400$, and on the circle, $u = 273$. The finite element analysis in this example is performed using the *Partial Differential Equation Toolbox* in MATLAB. We analyze the domain with triangular meshes that are automatically generated by the program once the mesh size parameter is specified [1]. We assume a Karhunen–Loéve expansion for the logarithm of the heat diffusion coefficient, having the same covariance kernel as the previous problem. In this example we use $N = 100$ standard normal Gaussian variables. We note that since our Karhunen–Loéve expansion is on the logarithm, strict positivity and boundedness of the elliptic coefficients are ensured with probability 1, and hence the solution exists and is unique almost surely [6].

For this high-dimensional problem, the training DQ rules are associated with hyperbolic cross index sets $\Lambda_p^T = \Lambda_p^{\mathrm{HC}}$ defined in (12); we train on the index sets associated with $p = 2, 3, 5$. For example, an $M = 109$-point DQ rule generated using the strategy in section 4.2 accurately integrates an index set Λ with $|\Lambda| = 5880$.

Similarly to previous examples, we obtain the constants from an offline analysis that we use in the optimization problem. The identification results are shown in

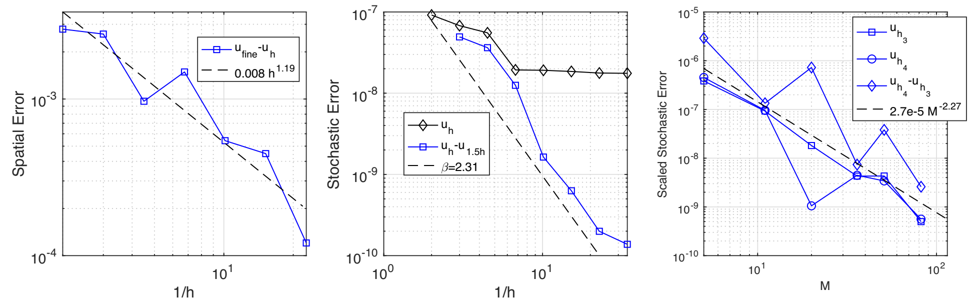

Fig. 11. *Identification of constants* $\alpha = 1.19$, $C_{\mathcal{H}} = 0.008$, $\beta = 2.31$, $C_{\mathcal{S}\mathbb{H}} = 2.7 \times 10^{-5}$, *and* $r = 2.27$ *for the heat diffusion problem with* $N = 100$ *stochastic variables.*

Figure 11. The formula of successive mesh sizes in this example is $h_i = \{0.5/1.5^i\}_{i=0}^{7}$. Note that the mesh sizes are reduced by a factor of 1.5. We use the following spatial approximations in the first two panes in Figure 11:

- Left pane: $u_h \in \{u_{h_0}, \ldots, u_{h_6}\}$.
- Middle pane (black curve): $u_h \in \{u_{h_0}, u_{h_1}, \ldots, u_{h_7}\}$.
- Middle pane (blue curve): $u_h - u_{1.5h} \in \{u_{h_1} - u_{h_0}, u_{h_2} - u_{h_1}, \ldots, u_{h_7} - u_{h_6}\}$.

The relationship between the optimal cost and prescribed error is shown in Figure 12. In contrast to the earlier example, in some cases a larger L yields a lower cost; see top left pane. Given a fixed ϵ, strategies with lower costs are selected as the optimal strategy. To verify Corollary 5.1, we fix $L = 7$ and compute the theoretical cost estimate similarly to previous examples. The bottom row of Figure 12 shows the theoretical cost with fixed L versus error (left pane) and the regression line for the normalized cost $\tilde{C}_{norm}$ versus error (right pane) with $1/r = 0.44$, which matches the rate of convergence $r = 2.27$ in Figure 11.

After finding the optimal number of points for each level, we compute the actual error, which is shown in Figure 13. In this example we compute the actual error against quasi-Monte Carlo rules since sparse grid rules yield a much larger number of points. In particular, we use an $M = 8192$-point quasi-Monte Carlo rule generated via a Sobol' sequence to compute the true solution. To this end we use the Sobol' point generator in MATLAB to generate a 100-dimensional Sobol' point sequence. We use $M = \{2^i\}_{i=0}^{10}$ points to obtain the quasi-Monte Carlo curve in Figure 13. Similarly to previous results, the MLDQ is more efficient than the single-level DQ. The single-level DQ also yields lower cost compared to quasi-Monte Carlo points.

Remark 6.2. We emphasize that there is a limit to the number of stochastic dimensions for which the quadrature approaches, including DQ, sparse grids, MLDQ, and MLSC, are plausible. As the number of dimensions grows, it is expected that the quadrature rules cease to be effective as their computation becomes intractable. Contrarily, for large stochastic dimensions or highly nonlinear problems in stochastic space that involve dense index sets, using stochastic sampling such as quasi-Monte Carlo sampling is proven to be more effective. The results in this section are pertinent only to the settings of the considered problem, and we expect that the conclusion drawn from the performance comparison (between quadrature rules and stochastic sampling approaches) would change for problems with different settings, e.g., problems with larger stochastic dimensions or those involving much larger M.

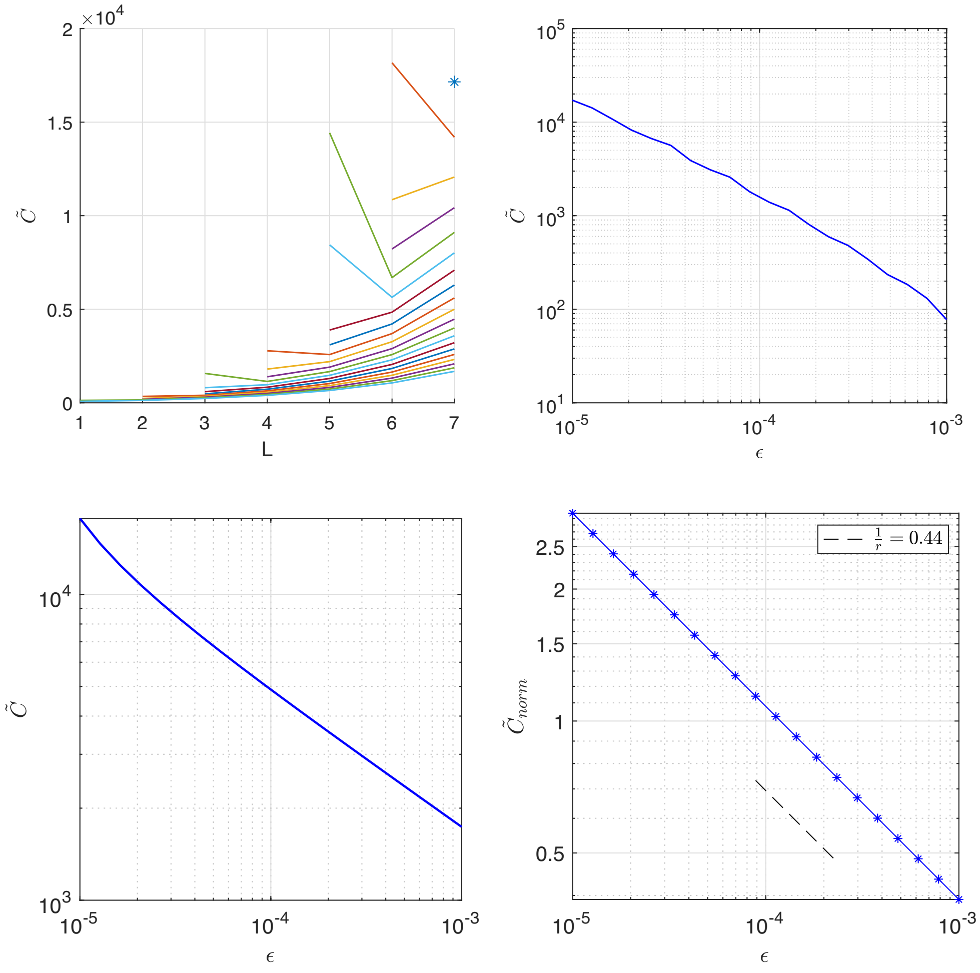

FIG. 12. *Optimal cost with respect to the number of levels (top-left), optimal cost versus total error with (top-right), cost versus error for a fixed $L = 7$ (bottom-left), verification of Corollary* 5.1 *with the estimated rate of convergence $1/r = 0.44$ matching the rate $r = 2.27$ in Figure* 11 *(bottom-right).*

7. Concluding remarks. A numerical method for systematic allocation of computational budget to different levels in hierarchical approximation of random PDEs is presented. Using the idea of multilevel Monte Carlo and recent MLSC methods, we propose a method that capitalizes on the flexibility of DQ for stochastic approximation with different levels of accuracy. DQ can be generated for any number of points to exactly integrate polynomials in a finite-dimensional polynomial space (equivalently the size of the index set). This approach substantially mitigates the issue of large growth in the number of points between grid levels, which was reported in previous MLSC methods. In all cases it is shown that the multilevel DQ approach is more efficient than the single-level computation. The algorithm is demonstrated for both cases of minimizing cost subject to error constraints and minimizing error subject to cost constraints. The proposed optimization problem is shown to be convex, and the analytical solution is provided in both cases and verified with numerical examples. We have demonstrated that the proposed method outperforms sparse grid

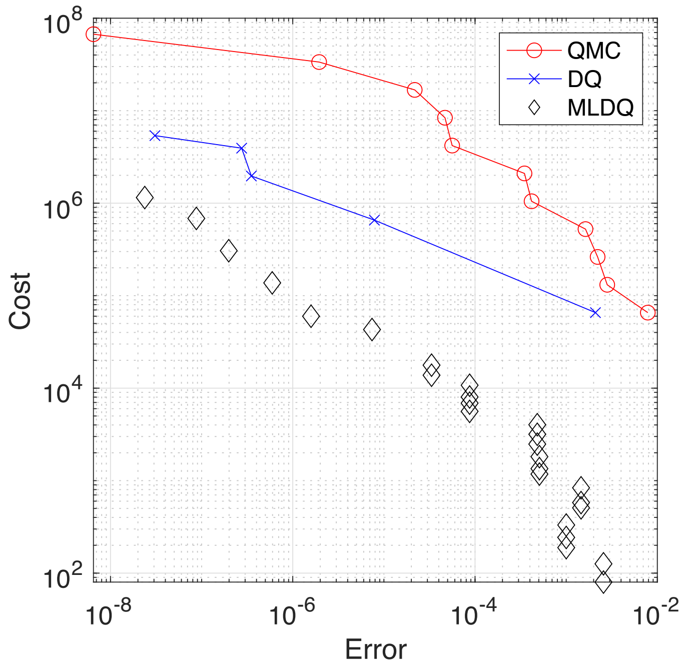

FIG. 13. *Actual cost versus error for MLDQ. The solid lines show the cost versus error for the finest mesh with two integration methods in high dimensions: DQ and quasi-Monte Carlo (QMC).*

quadrature and quasi-Monte Carlo approaches in the numerical examples considered on linear elasticity and heat diffusion problems with moderate stochastic dimensions. Future directions could explore extensions of this approach to a multi-index Monte Carlo-inspired method, which could help reduce the curse of dimensionality.

Acknowledgments. We thank the anonymous referees for substantial and detailed comments that greatly improved the quality of this paper.

REFERENCES

[1] *MATLAB and partial differential equation toolbox release* 2019*a*, The MathWorks, Inc., Natick, MA.

[2] E. Andreassen, A. Clausen, M. Schevenels, B. S. Lazarov, and O. Sigmund, *Efficient topology optimization in MATLAB using* 88 *lines of code*, Struct. Multidiscip. Optim., 43 (2011), pp. 1–16.

[3] J. Bäck, F. Nobile, L. Tamellini, and R. Tempone, *Stochastic spectral Galerkin and collocation methods for PDEs with random coefficients: A numerical comparison*, in Spectral and High Order Methods for Partial Differential Equations, J. S. Hesthaven and E. M. Rønguist, eds., Springer, Berlin, 2011, pp. 43–62.

[4] A. Barth, C. Schwab, and N. Zollinger, *Multi-level Monte Carlo finite element method for elliptic PDEs with stochastic coefficients*, Numer. Math., 119 (2011), pp. 123–161.

[5] J. Beck, R. Tempone, F. Nobile, and L. Tamellini, *On the optimal polynomial approximation of stochastic PDEs by Galerkin and collocation methods*, Math. Models Methods Appl. Sci., 22 (2012), 1250023.

[6] J. Charrier, *Strong and weak error estimates for elliptic partial differential equations with random coefficients*, SIAM J. Numer. Anal., 50 (2012), pp. 216–246.

[7] J. Charrier, R. Scheichl, and A. L. Teckentrup, *Finite element error analysis of elliptic PDEs with random coefficients and its application to multilevel Monte Carlo methods*, SIAM J. Numer. Anal., 51 (2013), pp. 322–352.

[8] 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), 3.

[9] A. Cohen and R. DeVore, *Approximation of high-dimensional parametric PDEs*, Acta Numer., 24 (2015), pp. 1–159.

[10] A. Cohen, R. DeVore, and C. Schwab, *Convergence rates of best n-term Galerkin approximations for a class of elliptic PDEs*, Found. Comput. Math., 10 (2010), pp. 615–646.

[11] A. Cohen, R. Devore, and C. Schwab, *Analytic regularity and polynomial approximation of parametric and stochastic elliptic PDEs*, Anal. Appl. (Singap.), 9 (2011), pp. 11–47.

[12] R. E. Curto and L. A. Fialkow, *A duality proof of Tchakaloff's theorem*, J. Math. Anal. Appl., 269 (2002), pp. 519–532.

[13] J. Dick, F. Y. Kuo, Q. T. Le Gia, and C. Schwab, *Multilevel higher order QMC Petrov–Galerkin discretization for affine parametric operator equations*, SIAM J. Numer. Anal., 54 (2016), pp. 2541–2568.

[14] I.-G. Farcaş, P. C. Sârbu, H.-J. Bungartz, T. Neckel, and B. Uekermann, *Multilevel adaptive stochastic collocation with dimensionality reduction*, in Sparse Grids and Applications – Miami 2016, J. Garcke, D. Pflüger, C. G. Webster, and G. Zhang, eds., Springer, Berlin, 2018, pp. 43–68.

[15] M. B. Giles, *Multilevel Monte Carlo methods*, Acta Numer., 24 (2015), pp. 259–328.

[16] M. B. Giles and C. Reisinger, *Stochastic finite differences and multilevel Monte Carlo for a class of SPDEs in finance*, SIAM J. Financial Math., 3 (2012), pp. 572–592.

[17] M. B. Giles and B. J. Waterhouse, *Multilevel quasi-Monte Carlo path simulation*, in Advanced Financial Modeling, Radon Ser. Comput. Appl. Math. 8, De Gruyter, Berlin, 2009, pp. 165–181.

[18] M. Griebel, H. Harbrecht, and M. D. Multerer, *Multilevel quadrature for elliptic parametric partial differential equations in case of polygonal approximations of curved domains*, SIAM J. Numer. Anal., 58 (2020), pp. 684–705.

[19] A.-L. Haji-Ali, F. Nobile, L. Tamellini, and R. Tempone, *Multi-index stochastic collocation for random PDEs*, Comput. Methods Appl. Mech. Engrg., 306 (2016), pp. 95–122.

[20] A.-L. Haji-Ali, F. Nobile, L. Tamellini, and R. Tempone, *Multi-index stochastic collocation for random PDEs*, Comput. Methods Appl. Mech. Engrg., 306 (2016), pp. 95–122.

[21] A.-L. Haji-Ali, F. Nobile, and R. Tempone, *Multi-index Monte Carlo: When sparsity meets sampling*, Numer. Math., 132 (2016), pp. 767–806.

[22] A.-L. Haji-Ali, F. Nobile, E. von Schwerin, and R. Tempone, *Optimization of mesh hierarchies in multilevel Monte Carlo samplers*, Stoch. Partial Differ. Equ. Anal. Comput., 4 (2016), pp. 76–112.

[23] H. Harbrecht, M. Peters, and M. Siebenmorgen, *On multilevel quadrature for elliptic stochastic partial differential equations*, in Sparse Grids and Applications, J. Garcke and M. Griebel, eds., Springer, Berlin, 2013, pp. 161–179.

[24] F. Heiss and V. Winschel, *Quadrature on Sparse Grids*, http://www.sparse-grids.de/.

[25] F. Heiss and V. Winschel, *Likelihood approximation by numerical integration on sparse grids*, J. Econometrics, 144 (2008), pp. 62–80.

[26] J. D. Jakeman, M. Eldred, G. Geraci, and A. Gorodetsky, *Adaptive multi-index collocation for uncertainty quantification and sensitivity analysis*, Numer. Methods Engrg., 121 (2020), pp. 1314–1343.

[27] V. Keshavarzzadeh, R. M. Kirby, and A. Narayan, *Numerical integration in multiple dimensions with designed quadrature*, SIAM J. Sci. Comput., 40 (2018), pp. A2033–A2061.

[28] F. Kuo, C. Schwab, and I. Sloan, *Multi-level quasi-Monte Carlo finite element methods for a class of elliptic PDEs with random coefficients*, Found. Comput. Math., 15 (2015), pp. 441–449.

[29] F. Y. Kuo, C. Schwab, and I. H. Sloan, *Quasi-Monte Carlo finite element methods for a class of elliptic partial differential equations with random coefficients*, SIAM J. Numer. Anal., 50 (2012), pp. 3351–3374.

[30] F. Nobile, R. Tempone, and C. G. Webster, *An anisotropic sparse grid stochastic collocation method for partial differential equations with random input data*, SIAM J. Numer. Anal., 46 (2008), pp. 2411–2442.

[31] A. L. Teckentrup, P. Jantsch, C. G. Webster, and M. Gunzburger, *A multilevel stochastic collocation method for partial differential equations with random input data*, SIAM/ASA J. Uncertain. Quantif., 3 (2015), pp. 1046–1074.

[32] 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.