SIAM J. SCI. COMPUT.
Vol. 37, No. 1, pp. A103–A133

A DOMAIN DECOMPOSITION APPROACH FOR UNCERTAINTY ANALYSIS*

QIFENG LIAO† AND KAREN WILLCOX†

Abstract. This paper proposes a decomposition approach for uncertainty analysis of systems governed by partial differential equations (PDEs). The system is split into local components using domain decomposition. Our domain-decomposed uncertainty quantification (DDUQ) approach performs uncertainty analysis independently on each local component in an "offline" phase, and then assembles global uncertainty analysis results using precomputed local information in an "online" phase. At the heart of the DDUQ approach is importance sampling, which weights the precomputed local PDE solutions appropriately so as to satisfy the domain decomposition coupling conditions. To avoid global PDE solves in the online phase, a proper orthogonal decomposition reduced model provides an efficient approximate representation of the coupling functions.

Key words. domain decomposition, model order reduction, uncertainty quantification, importance sampling

AMS subject classifications. 35R60, 60H15, 62D99, 65N55

DOI. 10.1137/140980508

1. Introduction.

Many problems arising in computational science and engineering are described by mathematical models of high complexity—involving multiple disciplines, characterized by a large number of parameters, and impacted by multiple sources of uncertainty. Decomposition of a system into subsystems or component parts has been one strategy to manage this complexity. For example, modern engineered systems are typically designed by multiple groups, usually decomposed along disciplinary or subsystem lines, sometimes spanning different organizations and even different geographical locations. Mathematical strategies have been developed for decomposing a simulation task (e.g., domain decomposition methods [45]), for decomposing an optimization task (e.g., domain decomposition for optimal design or control [28, 10, 29, 5]), and for decomposing a complex design task (e.g., decomposition approaches to multidisciplinary optimization [15, 34, 55]). In this paper we propose a decomposition approach for uncertainty quantification (UQ). In particular, we focus on the simulation of systems governed by stochastic partial differential equations (PDEs) and a domain decomposition approach to quantification of uncertainty in the corresponding quantities of interest.

In the design setting, decomposition approaches are not typically aimed at improving computational efficiency, although in some cases decomposition can admit parallelism that would otherwise be impossible [57]. Rather, decomposition is typically employed in situations for which it is infeasible or impractical to achieve tight coupling among the system subcomponents. This inability to achieve tight coupling becomes a particular problem when the goal is to wrap an outer loop (e.g., optimization) around the simulation. In the field of multidisciplinary design optimization (MDO), decomposition is achieved through mathematical formulations such as col-

*Submitted to the journal's Methods and Algorithms for Scientific Computing section August 1, 2014; accepted for publication October 13, 2014; published electronically January 8, 2015. This work was supported by AFOSR grant FA9550-12-1-0420, program manager F. Fahroo.
http://www.siam.org/journals/sisc/37-1/98050.html

†Department of Aeronautics and Astronautics, Massachusetts Institute of Technology, Cambridge, MA 02139 (qliao@mit.edu, kwillcox@mit.edu).

laborative optimization [35, 34, 53], bi-level integrated system synthesis [54, 55, 16], and analytic target cascading [43, 44]. These formulations involve optimization at a local subsystem level, communication among subsystems using coupling variables, and a system-level coordination to ensure overall compatibility among subsystems. In some cases, an added motivation for decomposition in MDO is to establish a mathematical formulation of the design problem that mimics design organizations and that promotes discipline autonomy [34, 57].

A decomposition approach to design has several important advantages. First, it manages complexity by performing computations at the local level (e.g., local optimization with discipline-specific design variables) and passing only the needed coupling information. Second, it permits the use of tailored optimization and solver strategies on different parts of the system (e.g., gradient-based optimization for disciplines with adjoint information, gradient-free optimization methods for disciplines with noisy objective functions). These subsystem optimizations can even be run on different computational platforms. Third, it permits disciplinary/subsystem experts to infuse their disciplinary expertise into the design of the corresponding part of the system, while accounting for interactions with other parts of the system through coupling variables. In this paper, we propose a distributed formulation for a decomposition approach to UQ with the goal of realizing similar advantages—in sum, decomposition can be an effective strategy for complex systems comprising multiple subcomponents or submodels, where it may be infeasible to achieve tight integration around which a UQ outer loop is wrapped.

Decomposition approaches are already a part of UQ in several settings with different goals. Alexander, Garcia, and Tartakovsky develop hybrid (multiscale) methods to decompose stochastic models into particle models and continuum stochastic PDE models, where interface coupling models are introduced to reconcile the noise (uncertainty) generated in both pieces [1, 2, 3]. Recent work has also considered the combination of domain decomposition methods and UQ, with a primary motivation to achieve computational efficiency by parallelizing the process of solving stochastic PDEs [47, 26]. Sarkar, Benabbou, and Ghanem introduce a domain decomposition method for solving stochastic PDEs [47], using a combination of Schur-complement-based domain decomposition [21] for the spatial domain and orthogonal decompositions of stochastic processes (i.e., polynomial chaos decomposition [24]) for the stochastic domain. Hadigol et al. consider coupled domain problems with high-dimensional random inputs [26], where the spatial domain is decomposed by finite element tearing and interconnecting [22], while the stochastic space is treated through separated representations [30]. Arnst et al. also consider stochastic modeling of coupled problems, focusing on the probabilistic information that flows between subsystem components [6, 7, 8]. They propose a method to reduce the dimension of the coupling random variables, using polynomial chaos expansion and Karhunen–Loève (KL) decomposition. For problems with high-dimensional coupling, the method of Arnst et al. could be applied to first reduce the coupling dimension before using our distributed UQ approach.

Here, we take a different approach that is motivated by decomposition-based design approaches. Just as the design decomposition approaches discussed above combine local optimization with overall system coordination, our domain-decomposed uncertainty quantification (DDUQ) approach combines UQ at the local subsystem level with a system-level coordination. The system-level coordination is achieved through importance sampling [51, 52]. Our primary goal is *not* to improve computational

efficiency—although indeed, our DDUQ approach could be combined with tailored surrogate models at the local subsystem level. Rather, through decomposing the uncertainty analysis task, we aim to realize advantages similar to those in the design setting. First, we aim to manage complexity by conducting stochastic analysis at the local level, avoiding the need for full integration of all parts of the system and thus making UQ accessible for those systems that cannot be tightly integrated. Second, we aim to break up the stochastic analysis task so that different strategies (e.g., different sampling strategies, different surrogate modeling strategies) can be used in different parts of the system, thus exploiting problem structure as much as possible. Third, we aim to move UQ from the system level to the disciplinary/subsystem group, thus infusing disciplinary expertise, ownership, and autonomy.

In this paper, we build upon the distributed method for forward propagation of uncertainty introduced by Amaral, Allaire, and Willcox [4]. We develop a mathematical framework using iterative domain decomposition. The combination of domain decomposition and importance sampling leads to a new approach that permits uncertainty analysis of each subdomain independently in an "offline" phase, followed by assembly of global uncertainty results using precomputed local information in an "online" phase. In section 2, we present the general problem setup. Section 3 reviews iterative domain decomposition methods for deterministic PDEs and presents our notation. Section 4 presents the DDUQ offline and online algorithms. Model reduction techniques for efficient handling of interface conditions are discussed in section 5, and a summary of our full approach is presented in section 6. Numerical studies are discussed in section 7. Finally, section 8 concludes the paper.

2. Problem setup. Let $D \subset \mathbb{R}^d$ $(d = 1, 2, 3)$ denote a spatial domain which is bounded, connected, and with a polygonal boundary ∂D (for $d = 2, 3$), and let $x \in \mathbb{R}^d$ denote a spatial variable. Let ξ be a vector which collects a finite number of random variables. The image of ξ is denoted by Γ and its probability density function (PDF) is denoted by $\pi_\xi(\xi)$. The physics of problems considered in this paper are governed by a PDE over the domain D and boundary conditions on the boundary ∂D. The global problem solves the governing equations which are stated as follows: find $u(x, \xi) : D \times \Gamma \to \mathbb{R}$ such that

$$\mathcal{L}\left(x, \xi; u\left(x, \xi\right)\right) = f(x, \xi) \qquad \forall (x, \xi) \in D \times \Gamma, \tag{2.1}$$

$$\mathfrak{b}\left(x, \xi; u\left(x, \xi\right)\right) = g(x, \xi) \qquad \forall (x, \xi) \in \partial D \times \Gamma, \tag{2.2}$$

where $\mathcal{L}$ is a partial differential operator and $\mathfrak{b}$ is a boundary operator, both of which can have random coefficients. f is the source function and g specifies the boundary conditions; these functions may also depend on random variables.

Our domain D can be represented by a finite number, M, of subdomains (components), i.e., $D = \cup_{i=1}^{M} D_i$. We consider the case where the intersection of two subdomains can only be a connected interface with positive $(d-1)$-dimensional measure or an empty set. For a subdomain D_i, ∂D_i denotes the set of its boundaries, and Λ_i denotes the set of its neighboring subdomain indices, i.e., $\Lambda_i := \{j \,|\, j \in \{1, \ldots, M\}, j \neq i \text{ and } \partial D_j \cap \partial D_i \neq \emptyset\}$. Moreover, the boundary set ∂D_i can be split into two parts: exterior boundaries $\partial_{\text{ex}} D_i := \partial D_i \cap \partial D$, and interfaces $\partial_{\text{in}} D_i := \cup_{j \in \Lambda_i} \{\partial_j D_i\}$ where $\partial_j D_i := \partial D_i \cap \partial D_j$, so that $\partial D_i = \partial_{\text{ex}} D_i \cup \partial_{\text{in}} D_i$. We label an interface $\partial_j D_i \in \partial_{\text{in}} D_i$ with the index pair (i, j); then, grouping all interface indices associated with all subdomains $\{D_i\}_{i=1}^{M}$ together, we define $\Lambda := \{(i, j) \,|\, i \in \{1, \ldots, M\} \text{ and } j \in \Lambda_i\}$. In this paper, it is also assumed that two different interfaces do not connect, i.e.,

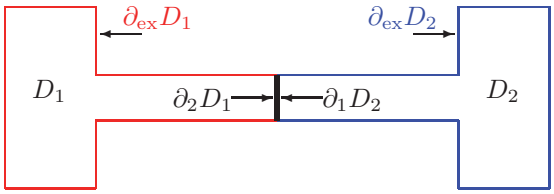


FIG. 1. *Example of a two-subdomain system:* $D = D_1 \cup D_2$, $\partial_{\text{in}} D_1 = \partial_2 D_1$, $\partial_{\text{in}} D_2 = \partial_1 D_2$, $\Lambda_1 = \{2\}$, $\Lambda_2 = \{1\}$, *and* $\Lambda = \{(1,2),(2,1)\}$.

$\partial_j D_i \cap \partial_k D_i = \emptyset$ for $j \neq k$. A notional example demonstrating this notation for two subdomains with single interface is shown in Figure 1.

In this paper, we assume that the random vector ξ can be decomposed into $\xi^T = [\xi_1^T, \ldots, \xi_M^T]$. Each ξ_i is a vector collecting $\mathfrak{N}_i$ random variables ($\mathfrak{N}_i$ is a positive integer and $\xi_i \in \mathbb{R}^{\mathfrak{N}_i}$), and it is associated with a spatial subdomain D_i. For each ξ_i, its image is denoted by Γ_i and its local joint PDF is denoted by $\pi_{\xi_i}(\xi_i)$.

We also define output quantities of interest. Our UQ task is to compute the uncertainty in outputs of interest, characterized by their PDFs and statistics of interest (mean, variance, etc.), as a result of uncertainty in the input parameters ξ. In the following, we develop our methodology for the case where the outputs are locally defined on each subdomain. Each output is denoted by $y_i\,(u(x,\xi)|_{D_i})$, $i = 1, \ldots, M$, and can be any quantity (e.g., integral quantities, nodal point values) provided it depends only on the solution in one local subdomain. However, our approach also applies to more general situations, such as outputs with contributions from multiple neighboring subdomains; the extension and application to such a case is presented in section 7.8.

In the next section, we present an iterative domain decomposition method for deterministic problems, then show how this framework can be combined with importance sampling to achieve a domain-decomposed approach to UQ.

3. Domain decomposition for deterministic problems. We review domain decomposition methods based on subdomain iterations for deterministic PDEs, following the presentation in [45]. In this section, we consider a realization of the random vector ξ; (2.1)–(2.2) then becomes a deterministic problem. Let $\{g_{i,j}(x, \tau_{j,i})\}_{(i,j)\in\Lambda}$ denote *interface functions*, that is, functions defined on the interfaces $\{\partial_j D_i\}_{(i,j)\in\Lambda}$. Each $g_{i,j}(x, \tau_{j,i})$ is dependent on a vector parameter $\tau_{j,i} \in \mathbb{R}^{\mathbf{N}_{j,i}}$, where $\mathbf{N}_{j,i}$ denotes the dimension of $\tau_{j,i}$. The parameter $\tau_{j,i}$ is used to describe interface data exchange between subdomains (details are in the following). For a given subdomain D_i, $\tau_{j,i}$ is called an *interface input parameter*, and $\tau_{i,j}$ is an *interface output parameter*. Grouping all the interface input parameters for D_i together, we define an entire interface input parameter $\tau_i \in \mathbb{R}^{\mathbf{N}_i}$ with $\mathbf{N}_i = \sum_{j\in\Lambda_i} \mathbf{N}_{j,i}$ such that

$$\tau_i := \otimes_{j\in\Lambda_i} \tau_{j,i},$$

where $\otimes$ combines vectors as follows:

$$\tau_{j_1,i} \otimes \tau_{j_2,i} := \begin{cases} \left[\tau_{j_1,i}^T, \tau_{j_2,i}^T\right]^T & \text{if } j_1 < j_2, \\ \left[\tau_{j_2,i}^T, \tau_{j_1,i}^T\right]^T & \text{if } j_1 > j_2, \\ \tau_{j_1,i} & \text{if } j_1 = j_2. \end{cases}$$

Taking the two-subdomain system in Figure 1, for example, $\tau_{1,2}$ represents interface data transferred from D_1 to D_2, and so it is the interface input parameter for

D_2 and the interface output parameter for D_1. Since there are only two subdomains in this example, we have $\tau_1 = \tau_{2,1}$ and $\tau_2 = \tau_{1,2}$.

Before presenting the domain decomposition method, we introduce decomposed local operators $\{\mathcal{L}_i := \mathcal{L}|_{D_i}\}_{i=1}^M$ and $\{\mathfrak{b}_i := \mathfrak{b}|_{D_i}\}_{i=1}^M$ and local functions $\{f_i := f|_{D_i}\}_{i=1}^M$ and $\{g_i := g|_{D_i}\}_{i=1}^M$, which are global operators and functions restricted to each subdomain D_i. Following [45], the domain decomposition methodology based on subdomain iterations proceeds as follows. Given initial guess $\tau_{i,j}^0$ (e.g., $\tau_{i,j}^0 = 0$) for each interface parameter, for each iteration step $k = 0, 1, \ldots$, we solve M local problems: find $u(x, \xi_i, \tau_i^k) : D_i \to \mathbb{R}$ for $i = 1, \ldots, M$, where $\tau_i^k = \otimes_{j\in\Lambda_i}\tau_{j,i}^k$ is the interface input parameter on the kth subdomain iteration, such that

$$\mathcal{L}_i\left(x, \xi_i; u\left(x, \xi_i, \tau_i^k\right)\right) = f_i\left(x, \xi_i\right) \qquad \text{in} \quad D_i, \tag{3.1}$$

$$\mathfrak{b}_i\left(x, \xi_i; u\left(x, \xi_i, \tau_i^k\right)\right) = g_i\left(x, \xi_i\right) \qquad \text{on} \quad \partial_{\text{ex}} D_i, \tag{3.2}$$

$$\mathfrak{b}_{i,j}\left(x, \xi_i; u\left(x, \xi_i, \tau_i^k\right)\right) = g_{i,j}\left(x, \tau_{j,i}^k\right) \qquad \text{on} \quad \partial_j D_i \ \forall j \in \Lambda_i. \tag{3.3}$$

Equation (3.3) defines the boundary conditions on interfaces and $\mathfrak{b}_{i,j}$ is an appropriate boundary operator posed on the interface $\partial_j D_i$. The interface parameters are updated for the next iteration by taking interface data of each local solution. We denote this updating process by the functional $\mathfrak{h}_{i,j}$ such that

$$\tau_{i,j}^{k+1} := \mathfrak{h}_{i,j}\left(u\left(x, \xi_i, \tau_i^k\right)\right). \tag{3.4}$$

For simplicity, (3.4) is rewritten as a function $h_{i,j} : \mathbb{R}^{\mathfrak{N}_i + \mathbf{N}_i} \to \mathbb{R}$ such that

$$\tau_{i,j}^{k+1} := h_{i,j}\left(\xi_i, \tau_i^k\right), \tag{3.5}$$

where, in the standard domain decomposition implementation, evaluating the values of $h_{i,j}$ involves solving local PDEs. In the following, $h_{i,j}$ is referred to as a *coupling function*.

In this paper, we assume that $u(x, \xi_i, \tau_i^k)$ and $u(x, \xi\,)|_{D_i}$ belong to a normed space with spatial function norm $\|\cdot\|_{D_i}$. The following standard (domain decomposition) DD-convergence definition [45] can be introduced.

DEFINITION 3.1 (DD-convergence). *A domain decomposition method is convergent if*

$$\lim_{k\to\infty} \left\| u\left(x, \xi_i, \tau_i^k\right) - u\left(x, \xi\right)\big|_{D_i} \right\|_{D_i} \to 0, \quad i = 1, \ldots, M. \tag{3.6}$$

The quantity $\tau_i^\infty := \otimes_{j\in\Lambda_i}\tau_{j,i}^\infty$ (where $\tau_{j,i}^\infty := \lim_{k\to\infty}\tau_{j,i}^k$) is called a target interface input parameter for subdomain D_i, and $u(x, \xi_i, \tau_i^\infty)$ is a local stationary solution. From (3.6), each stationary solution is consistent with the solution of the global PDEs (2.1)–(2.2) in the sense that $u(x, \xi_i, \tau_i^\infty) = u(x, \xi)|_{D_i}$.

In order to guarantee the DD-convergence condition in Definition 3.1, relaxation may be required at each updating step, which is to modify (3.5) as

$$\tau_{i,j}^{k+1} = \theta_{i,j} h_{i,j}\left(\xi_i, \tau_i^k\right) + (1 - \theta_{i,j})\tau_{i,j}^k, \tag{3.7}$$

where the $\theta_{i,j}$ are nonnegative acceleration parameters (see [45, pp. 118–119] for details).

4. DDUQ. Our goal is to peform propagation of uncertainty from the uncertain inputs ξ, with specified joint input PDF $\pi_\xi(\xi)$, to the outputs of interest $\{y_i(u(x,\xi)|_{D_i})\}_{i=1}^M$. In this section we show how uncertainty propagation via Monte Carlo sampling can be combined with the domain decomposition framework presented in section 3. We first present the DDUQ algorithm, then analyze its convergence properties.

4.1. DDUQ algorithm. We introduce a domain-decomposed uncertainty analysis approach based on the methodology of importance sampling [51, 52], which can be decomposed into two stages: offline (initial sampling) and online (reweighting) [4]. The offline stage of our method carries out Monte Carlo simulation at the local subdomain level. The online stage uses a reweighting technique to satisfy the interface conditions and ensure solution compatibility at the domain level. The offline stage involves local PDE solves which can be expensive, whereas the online stage avoids PDE solves and thus is cheap. To begin with, we denote the joint PDF of ξ_i and $\tau_i^\infty(\xi)$ by $\pi_{\xi_i,\tau_i}(\xi_i,\tau_i)$, which is referred to as the target input PDF for subdomain D_i.

In the offline stage of DDUQ, we first specify a proposal input PDF from which the samples will be drawn. We denote the proposal input PDF by $p_{\xi_i,\tau_i}(\xi_i,\tau_i)$. The proposal input PDF must be chosen so that its support is large enough to cover the support of $\tau_i^\infty(\xi)$. Provided this condition on the support is met, any proposal input PDF can be used; however, the better the proposal (in the sense that it generates sufficient samples over the support of the target input PDF) the better the performance of the importance sampling. A poor choice of proposal input PDF will lead to many wasted samples (i.e., requiring many local PDE solves). In our case, since the PDF of ξ_i is given, only the contribution of $\tau_i^\infty(\xi)$ in $\pi_{\xi_i,\tau_i}(\xi_i,\tau_i)$ is unknown. Thus, we choose the proposal PDF $p_{\xi_i,\tau_i}(\xi_i,\tau_i) = \pi_{\xi_i}(\xi_i)p_{\tau_i}(\tau_i)$, where $p_{\tau_i}(\tau_i)$ is a proposal for τ_i, chosen to be any PDF whose support is large enough to cover the expected support of $\tau_i^\infty(\xi)$.

The next step, still in the offline phase, is to perform UQ on each local system independently. This involves generating a large number N_i of samples $\{(\xi_i^{(s)},\tau_i^{(s)})\}_{s=1}^{N_i}$ drawn from $p_{\xi_i,\tau_i}(\xi_i,\tau_i)$ for each subdomain $i = 1,\dots,M$, where the superscript (s) denotes the sth sample. We then compute the local solutions $\{u(x,\xi_i^{(s)},\tau_i^{(s)})\}_{s=1}^{N_i}$ by solving the following deterministic problem for each sample:

$$\mathcal{L}_i\left(x,\xi_i^{(s)};u\left(x,\xi_i^{(s)},\tau_i^{(s)}\right)\right) = f_i\left(x,\xi_i^{(s)}\right) \quad \text{in} \quad D_i, \tag{4.1}$$

$$\mathfrak{b}_i\left(x,\xi_i^{(s)};u\left(x,\xi_i^{(s)},\tau_i^{(s)}\right)\right) = g_i\left(x,\xi_i^{(s)}\right) \quad \text{on} \quad \partial_{\text{ex}}D_i, \tag{4.2}$$

$$\mathfrak{b}_{i,j}\left(x,\xi_i^{(s)};u\left(x,\xi_i^{(s)},\tau_i^{(s)}\right)\right) = g_{i,j}\left(x,\tau_{j,i}^{(s)}\right) \quad \text{on} \quad \partial_j D_i \ \forall j \in \Lambda_i, \tag{4.3}$$

where $\tau_i^{(s)} = \otimes_{j\in\Lambda_i}\tau_{j,i}^{(s)}$. Once the local solutions are computed, we evaluate the local outputs of interest $\{y_i(u(x,\xi_i^{(s)},\tau_i^{(s)}))\}_{s=1}^{N_i}$ and store them. The offline process is summarized in Algorithm 1.

In the online stage of DDUQ, we first generate N_{on} samples, $\{\xi^{(s)}\}_{s=1}^{N_{\text{on}}}$, of the joint PDF of inputs $\pi_\xi(\xi)$. For each sample $\xi^{(s)}$, we use the domain decomposition iteration to evaluate the corresponding interface parameters $\tau_{i,j}^\infty(\xi^{(s)})$. The procedure is as follows:

1. set $k = 0$; take initial guesses $\{\tau_{i,j}^0(\xi^{(s)})\}_{(i,j)\in\Lambda}$ for the interface parameters;
2. evaluate each coupling function $h_{i,j}(\xi_i^{(s)},\tau_i^k(\xi^{(s)}))$ (see (3.5)) for $(i,j)\in\Lambda$;

Algorithm 1. DDUQ offline.

For each local subdomain D_i, $i = 1, \ldots, M$, the offline stage has the following steps:

1: Define a proposal PDF $p_{\xi_i,\tau_i}(\xi_i, \tau_i)$.
2: Generate samples $\left\{\left(\xi_i^{(s)}, \tau_i^{(s)}\right)\right\}_{s=1}^{N_i}$ of $p_{\xi_i,\tau_i}(\xi_i, \tau_i)$, where N_i is large.
3: Compute local solutions $\left\{u\left(x, \xi_i^{(s)}, \tau_i^{(s)}\right)\right\}_{s=1}^{N_i}$ by solving (4.1)–(4.3) for each sample.
4: Evaluate the local system outputs $\left\{y_i\left(u\left(x, \xi_i^{(s)}, \tau_i^{(s)}\right)\right)\right\}_{s=1}^{N_i}$.

Algorithm 2. DDUQ online.

1: Generate samples $\{\xi^{(s)}\}_{s=1}^{N_{\text{on}}}$ of the PDF $\pi_\xi(\xi)$.
2: **for** $s = 1 : N_{\text{on}}$ **do**
3: Initialize the interface parameters $\tau_{i,j}^0(\xi^{(s)})$ and $\tau_{i,j}^1(\xi^{(s)})$ for $(i, j) \in \Lambda$, such that $\max_{(i,j)\in\Lambda}(\|\tau_{i,j}^1(\xi^{(s)}) - \tau_{i,j}^0(\xi^{(s)})\|_\infty) > tol$.
4: Set $k = 0$.
5: **while** $\max_{(i,j)\in\Lambda}(\|\tau_{i,j}^{k+1}(\xi^{(s)}) - \tau_{i,j}^k(\xi^{(s)})\|_\infty) > tol$ **do**
6: **for** $i = 1, \ldots, M$ **do**
7: **for** $j \in \Lambda_i$ **do**
8: $\tau_{i,j}^{k+1}(\xi^{(s)}) = \theta_{i,j} h_{i,j}\left(\xi_i^{(s)}, \tau_i^k(\xi^{(s)})\right) + (1 - \theta_{i,j})\tau_{i,j}^k(\xi^{(s)})$.
9: **end for**
10: **end for**
11: Update $k = k + 1$.
12: **end while**
13: Set $\tau_{i,j}^\infty(\xi^{(s)}) = \tau_{i,j}^k(\xi^{(s)})$ for $(i, j) \in \Lambda$.
14: **end for**
15: Estimate a PDF $\hat{\pi}_{\xi_i,\tau_i}(\xi_i, \tau_i)$ from the samples $\{(\xi_i^{(s)}, \tau_i^\infty(\xi^{(s)}))\}_{s=1}^{N_{\text{on}}}$, for each subdomain D_i, $i = 1, \ldots, M$.
16: Reweight the precomputed offline local outputs: $\{w_i^{(s)} y_i(u(x, \xi_i^{(s)}, \tau_i^{(s)}))\}_{s=1}^{N_i}$ with $w_i^{(s)} = \frac{\hat{\pi}_{\xi_i,\tau_i}(\xi_i^{(s)}, \tau_i^{(s)})}{p_{\xi_i,\tau_i}(\xi_i^{(s)}, \tau_i^{(s)})}$, for each subdomain D_i, $i = 1, \ldots, M$.

3. update each interface parameter: $\tau_{i,j}^{k+1}(\xi^{(s)}) = \theta_{i,j} h_{i,j}\left(\xi_i^{(s)}, \tau_i^k(\xi^{(s)})\right) + (1 - \theta_{i,j})\tau_{i,j}^k(\xi^{(s)})$ (see (3.7)) for $(i, j) \in \Lambda$;
4. increment k; repeat steps 2 and 3 until an error indicator of the domain decomposition iteration is below some given tolerance tol.

To determine convergence in step 4, we use the error indicator $\max_{(i,j)\in\Lambda}(\|\tau_{i,j}^{k+1}(\xi^{(s)}) - \tau_{i,j}^k(\xi^{(s)})\|_\infty)$, where $\|\cdot\|_\infty$ is the vector infinity norm. When this error indicator is sufficiently small, the iteration terminates and we take $\tau_{i,j}^\infty(\xi^{(s)}) = \tau_{i,j}^k(\xi^{(s)})$ for $(i, j) \in \Lambda$ and $\tau_i^\infty(\xi^{(s)}) = \tau_i^k(\xi^{(s)})$ for $i = 1, \ldots, M$. We note that this may introduce additional approximation errors; however, by choosing a small value of the tolerance tol, the effects of these errors can be made small.

After the above process, we have obtained a set of samples $\{(\xi_i^{(s)}, \tau_i^\infty(\xi^{(s)}))\}_{s=1}^{N_{\text{on}}}$ drawn from each local target input PDF $\pi_{\xi_i,\tau_i}(\xi_i, \tau_i)$ for $i = 1, \ldots, M$. We next

estimate each local target input PDF from the samples using a density estimation technique, such as kernel density estimation (KDE) [27]. (See [4] for a detailed discussion of other density estimation methods.) In the following, the estimated target input PDF is denoted by $\hat{\pi}_{\xi_i,\tau_i}(\xi_i,\tau_i)$. Under an assumption that the underlying PDF is uniformly continuous in an unbounded domain and with an appropriate choice of density estimation method, the estimated PDF converges to the exact PDF as $N_{\text{on}} \to \infty$, i.e., $\lim_{N_{\text{on}}\to\infty} \hat{\pi}_{\xi_i,\tau_i}(\xi_i,\tau_i) = \pi_{\xi_i,\tau_i}(\xi_i,\tau_i)$. See [27, 58] for the convergence analysis of KDE. However, we note that for a finite set of samples, KDE can be prone to errors, especially for high-dimensional data.

The final step of the DDUQ online stage is to use importance sampling to reweight the precomputed outputs $\{y_i(u(x,\xi_i^{(s)},\tau_i^{(s)}))\}_{s=1}^{N_i}$ that we generated by PDE solves in the offline stage. The weights, $w_i^{(s)}$, are computed using standard importance sampling [52], by taking the ratio of the estimated target PDF to the proposal PDF for each local subdomain:

$$w_i^{(s)} = \frac{\hat{\pi}_{\xi_i,\tau_i}\left(\xi_i^{(s)},\tau_i^{(s)}\right)}{p_{\xi_i,\tau_i}\left(\xi_i^{(s)},\tau_i^{(s)}\right)}, \quad s = 1,\dots,N_i, \quad i = 1,\dots,M.$$

We will show in the next subsection that the probability computed from these weighted samples $\{w_i^{(s)} y_i(u(x,\xi_i^{(s)},\tau_i^{(s)}))\}_{s=1}^{N_i}$ is consistent with the true distributions of the outputs.

The details of the online computation strategy are summarized in Algorithm 2. We note that in order to achieve our goal of having no PDE solves in the online stage of the DDUQ algorithm, we still require a way to evaluate each coupling function $h_{i,j}$ (see (3.5)) in the online stage without requiring PDE solves. We address this by building surrogate models for the coupling functions $\{h_{i,j}\}_{(i,j)\in\Lambda}$ in the offline stage. Section 5 details the building of these coupling surrogates, which are denoted by $\{\tilde{h}_{i,j}\}_{(i,j)\in\Lambda}$ in the following.

4.2. Convergence analysis. For the following analysis we consider the outputs of interest $\{y_i(u(x,\xi)|_{D_i})\}_{i=1}^{M}$ of (2.1)–(2.2) to be scalar. It is straightforward to extend the analysis to the case of multiple outputs. For a given number a, we denote the actual probability of $y_i(u(x,\xi)|_{D_i}) \le a$ by $P(y_i \le a)$, i.e.,

$$P(y_i \le a) := \int_{\Gamma_i^a} \pi_\xi(\xi)\,\mathrm{d}\xi,$$

where

$$\Gamma_i^a := \left\{ \xi \,\middle|\, \xi \in \Gamma, y_i\left(u\left(x,\xi\right)\big|_{D_i}\right) \le a \right\}.$$

Next, we denote the support of the local proposal input PDF $p_{\xi_i,\tau_i}(\xi_i,\tau_i)$ by $\Gamma_{P,i}$ and the support of the local target input PDF $\pi_{\xi_i,\tau_i}(\xi_i,\tau_i)$ by $\Gamma_{T,i}$. The importance sampling approach requires that $p_{\xi_i,\tau_i}(\xi_i,\tau_i)$ is chosen so that $\Gamma_{T,i} \subseteq \Gamma_{P,i}$. We define the probability of the local output $y_i(u(x,\xi_i,\tau_i))$ associated with the local target input PDF $\pi_{\xi_i,\tau_i}(\xi_i,\tau_i)$ by

$$\tilde{P}(y_i \le a) := \int_{\tilde{\Gamma}_i^a} \pi_{\xi_i,\tau_i}\left(\xi_i,\tau_i\right)\,\mathrm{d}\xi_i\,\mathrm{d}\tau_i, \tag{4.4}$$

where

$$\tilde{\Gamma}_i^a := \left\{ \left[\begin{array}{c} \xi_i \\ \tau_i \end{array} \right] \middle| \left[\begin{array}{c} \xi_i \\ \tau_i \end{array} \right] \in \Gamma_{T,i}, \quad y_i \left(u \left(x, \xi_i, \tau_i \right) \right) \leq a \right\}. \tag{4.5}$$

At the last step of the DDUQ algorithm (line 16 of Algorithm 2), weighted samples are obtained. We next define the probability associated with the weighted samples. To begin, we consider the *exact* weights:

$$\mathbf{W}_i^{(s)} = \frac{\pi_{\xi_i,\tau_i}\left(\xi_i^{(s)}, \tau_i^{(s)}\right)}{p_{\xi_i,\tau_i}\left(\xi_i^{(s)}, \tau_i^{(s)}\right)}, \quad s = 1, \ldots, N_i, \quad i = 1, \ldots, M, \tag{4.6}$$

which are associated with the exact local target input PDF $\pi_{\xi_i,\tau_i}(\xi_i, \tau_i)$ rather than the estimated PDF $\hat{\pi}_{\xi_i,\tau_i}(\xi_i, \tau_i)$ on line 15 of Algorithm 2. With these exact weights, we define

$$P_{\mathbf{W}_i}(y_i \leq a) := \frac{\sum_{s=1}^{N_i} \mathbf{W}_i^{(s)} \tilde{I}_i^a(\xi_i^{(s)}, \tau_i^{(s)})}{\sum_{s=1}^{N_i} \mathbf{W}_i^{(s)}}, \tag{4.7}$$

where $\tilde{I}_i^a(\cdot, \cdot)$ is an indicator function for local outputs:

$$\tilde{I}_i^a(\xi_i, \tau_i) := \begin{cases} 1 & \text{if } y_i \left(u \left(x, \xi_i, \tau_i \right) \right) \leq a, \\ 0 & \text{if } y_i \left(u \left(x, \xi_i, \tau_i \right) \right) > a. \end{cases} \tag{4.8}$$

For the DDUQ outputs (see line 16 of Algorithm 2) with weights $\{w_i^{(s)}\}_{s=1}^{N_i}$, obtained from the estimated target PDFs $\hat{\pi}_{\xi_i,\tau_i}(\xi_i, \tau_i)$ for $i = 1, \ldots, M$, we define

$$P_{w_i}(y_i \leq a) := \frac{\sum_{s=1}^{N_i} w_i^{(s)} \tilde{I}_i^a(\xi_i^{(s)}, \tau_i^{(s)})}{\sum_{s=1}^{N_i} w_i^{(s)}}. \tag{4.9}$$

In the following, we show that $P_{\mathbf{W}_i}(y_i \leq a)$ approaches the actual probability $P(y_i \leq a)$ as the offline sample size N_i $(i = 1, \ldots, M)$ increases. The analysis takes two steps: the first step is to prove $\tilde{P}(y_i \leq a) = P(y_i \leq a)$ and the second step is to show $P_{\mathbf{W}_i}(y_i \leq a) \to \tilde{P}(y_i \leq a)$ as N_i goes to infinity.

LEMMA 4.1. *If the DD-convergence condition is satisfied for all $\xi \in \Gamma$, then for any given number a, $P(y_i \leq a) = \tilde{P}(y_i \leq a)$.*

Proof. Let $\{\xi^{(s)}\}_{s=1}^N$ be N realizations of ξ. Using the convergence property of the Monte Carlo integration [14], we have

$$\begin{aligned} P(y_i \leq a) &= \int_{\Gamma_i^a} \pi_\xi(\xi) \, \mathrm{d}\xi \\ &= \lim_{N \to \infty} \frac{1}{N} \sum_{s=1}^{N} I_i^a \left(\xi^{(s)} \right), \end{aligned} \tag{4.10}$$

where $I_i^a(\cdot)$ is an indicator function defined by

$$I_i^a(\xi) := \begin{cases} 1 & \text{if } y_i \left(u \left(x, \xi \right) \big|_{D_i} \right) \leq a, \\ 0 & \text{if } y_i \left(u \left(x, \xi \right) \big|_{D_i} \right) > a. \end{cases}$$

Since the DD-convergence condition is assumed to be satisfied for all $\xi \in \Gamma$, we have $u(x, \xi_i, \tau_i^\infty(\xi)) = u(x, \xi)|_{D_i}$. Thus, $I_i^a(\xi) = \tilde{I}_i^a(\xi_i, \tau_i^\infty(\xi))$. From section 4.1, we have that $\{(\xi_i^{(s)}, \tau_i^\infty(\xi^{(s)}))\}_{s=1}^N$ are realizations of $\pi_{\xi_i,\tau_i}(\xi_i, \tau_i)$. So,

$$\begin{aligned}\lim_{N\to\infty} \frac{1}{N}\sum_{s=1}^{N} I_i^a\left(\xi^{(s)}\right) &= \lim_{N\to\infty} \frac{1}{N}\sum_{s=1}^{N} \tilde{I}_i^a\left(\xi_i^{(s)}, \tau_i^\infty(\xi^{(s)})\right) \\ &= \int_{\tilde{\Gamma}_i^a} \pi_{\xi_i,\tau_i}(\xi_i, \tau_i)\, \mathrm{d}\xi_i\, \mathrm{d}\tau_i = \tilde{P}(y_i \le a). \end{aligned} \tag{4.11}$$

Combining (4.10) and (4.11), this lemma is proved. ∎

THEOREM 4.2. *If the DD-convergence condition is satisfied for all $\xi \in \Gamma$, and the support of the local proposal input PDF $p_{\xi_i,\tau_i}(\xi_i, \tau_i)$ covers that of the local target input PDF $\pi_{\xi_i,\tau_i}(\xi_i, \tau_i)$, i.e., $\Gamma_{P,i} \supseteq \Gamma_{T,i}$ for $i = 1, \ldots, M$, then for any given number a, the following equality holds:*

$$\lim_{N_i\to\infty} P_{\mathbf{W}_i}(y_i \le a) = P(y_i \le a). \tag{4.12}$$

Proof. Our proof follows the standard importance sampling convergence analysis techniques [4, 52]. From (4.6)–(4.7),

$$\begin{aligned} P_{\mathbf{W}_i}(y_i \le a) &= \frac{\sum_{s=1}^{N_i} \mathbf{W}_i^{(s)} \tilde{I}_i^a(\xi_i^{(s)}, \tau_i^{(s)})}{\sum_{s=1}^{N_i} \mathbf{W}_i^{(s)}} \\ &= \frac{\frac{1}{N_i}\sum_{s=1}^{N_i} \frac{\pi_{\xi_i,\tau_i}\left(\xi_i^{(s)},\tau_i^{(s)}\right)}{p_{\xi_i,\tau_i}\left(\xi_i^{(s)},\tau_i^{(s)}\right)} \tilde{I}_i^a(\xi_i^{(s)}, \tau_i^{(s)})}{\frac{1}{N_i}\sum_{s=1}^{N_i} \frac{\pi_{\xi_i,\tau_i}\left(\xi_i^{(s)},\tau_i^{(s)}\right)}{p_{\xi_i,\tau_i}\left(\xi_i^{(s)},\tau_i^{(s)}\right)}}. \end{aligned}$$

Using the convergence property of Monte Carlo integration [14] and due to $\Gamma_{T,i} \subseteq \Gamma_{P,i}$, we have

$$\begin{aligned} \lim_{N_i\to\infty} P_{\mathbf{W}_i}(y_i \le a) &= \frac{\int_{\Gamma_{P,i}} \frac{\pi_{\xi_i,\tau_i}(\xi_i,\tau_i)}{p_{\xi_i,\tau_i}(\xi_i,\tau_i)} \tilde{I}_i^a(\xi_i, \tau_i) p_{\xi_i,\tau_i}(\xi_i, \tau_i)\, \mathrm{d}\xi_i\, \mathrm{d}\tau_i}{\int_{\Gamma_{P,i}} \frac{\pi_{\xi_i,\tau_i}(\xi_i,\tau_i)}{p_{\xi_i,\tau_i}(\xi_i,\tau_i)} p_{\xi_i,\tau_i}(\xi_i, \tau_i)\, \mathrm{d}\xi_i\, \mathrm{d}\tau_i} \\ &= \frac{\int_{\Gamma_{T,i}} \pi_{\xi_i,\tau_i}(\xi_i, \tau_i)\, \tilde{I}_i^a(\xi_i, \tau_i)\, \mathrm{d}\xi_i\, \mathrm{d}\tau_i}{\int_{\Gamma_{T,i}} \pi_{\xi_i,\tau_i}(\xi_i, \tau_i)\, \mathrm{d}\xi_i\, \mathrm{d}\tau_i} \\ &= \int_{\tilde{\Gamma}_i^a} \pi_{\xi_i,\tau_i}(\xi_i, \tau_i)\, \mathrm{d}\xi_i\, \mathrm{d}\tau_i \\ &= \tilde{P}(y_i \le a). \end{aligned} \tag{4.13}$$

Using Lemma 4.1 and (4.13), this theorem is proved. ∎

Remark 4.3. If we have a strongly convergent density estimation of local target input PDFs, i.e., $\lim_{N_{\text{on}}\to\infty} \hat{\pi}_{\xi_i,\tau_i}(\xi_i, \tau_i) = \pi_{\xi_i,\tau_i}(\xi_i, \tau_i)$ for $i = 1, \ldots, M$, then from (4.6)–(4.9), for any given number a, we have $P_{\mathbf{W}_i}(y_i < a) = P_{w_i}(y_i < a)$ as $N_{\text{on}} \to \infty$. This combined with Theorem 4.2 leads to the overall convergence of the DDUQ approach, i.e., $\lim_{N_i\to\infty, N_{\text{on}}\to\infty} P_{w_i}(y_i \le a) = P(y_i \le a)$. KDE is proved to be strongly convergent in [27, 58], when the underlying PDF is uniformly continuous with an unbounded range. However, it still remains an open question to prove the strong convergence of density estimation techniques for PDFs with bounded ranges. We will discuss this again in section 7.2.

5. Reduced-order modeling for coupling functions. In this section we develop an efficient surrogate model for each coupling function $h_{i,j}$, $(i,j) \in \Lambda$. This model is needed so that the online DDUQ process of assimilating local sample information to the system level can be conducted without PDE solves. The approach consists of the following two steps. First, we construct a proper orthogonal decomposition (POD) basis in which we represent the interface function $g_{i,j}$. With this representation, the interface parameter $\tau_{i,j}$ becomes the coefficients of the POD expansion and thus has a reduced dimension. Second, we build surrogate models for $h_{i,j}$. The first step is important to keep the dependency of the surrogates low dimensional, and the interface parameters must be low dimensional since we need to estimate their PDFs in the online stage.

5.1. Dimension reduction for interface parameters. We consider solution of the local problems (3.1)–(3.3) or (4.1)–(4.3) by a numerical discretization method, such as the finite element method. For a finite element discretization, an interface function $g_{i,j}$ defined on an interface $\partial_j D_i$, $(i,j) \in \Lambda$, has the following form:

$$g_{i,j} = \sum_{t=1}^{\mathbf{N}_{i,j}} c_t^{i,j} \phi_t^{i,j}(x), \tag{5.1}$$

where $\{\phi_t^{i,j}(x)\}_{t=1}^{\mathbf{N}_{i,j}}$ are finite element basis functions on the interface $\partial_j D_i$, $\mathbf{N}_{i,j}$ is the number of basis functions, and $\{c_t^{i,j}\}_{t=1}^{\mathbf{N}_{i,j}}$ are their corresponding coefficients. One option would be to parameterize the interface functions by the finite element coefficients, i.e., the interface parameters introduced in section 3 would be defined as $\tau_{i,j} = [c_1^{i,j}, \ldots, c_{\mathbf{N}_{i,j}}^{i,j}]^T$. However, since the finite element basis functions are dependent on the spatial grid, this choice of parameterization would typically lead to a large dimension $\mathbf{N}_{i,j}$ of the interface parameters.

To obtain a lower-dimensional representation of the interface parameters, we use the POD method [25, 31, 50]. That is, we first generate a small number $\tilde{N}$ of realizations $\{\xi^{(s)}\}_{s=1}^{\tilde{N}}$ of the random input vector ξ. Next, we carry out standard domain decomposition iterations (see section 3) for each realization. This process generates a sample (or so-called POD snapshot) of the (discretized) interface function corresponding to each sampled value of ξ: $\{g_{i,j}(x, \tau_{j,i}^{\infty}(\xi^{(s)}))\}_{s=1}^{\tilde{N}}$. Applying the POD to this snapshot set results in a POD basis for the interface functions. Then, in (5.1) we let $\{\phi_t^{i,j}(x)\}_{t=1}^{\mathbf{N}_{i,j}}$ refer to the POD basis and $\{c_t^{i,j}\}_{t=1}^{\mathbf{N}_{i,j}}$ become the corresponding POD coefficients. Due to the compression typically obtained using POD, the dimension $\mathbf{N}_{i,j}$ of the interface parameter $\tau_{i,j}$ will now be small (and in particular independent of the spatial discretization dimension). This approximation of the interface functions in a reduced basis is also used in the static condensation reduced basis method, where it is referred to as "port reduction" [18]. Another option for reducing the dimension of the coupling variables is to use the method of Arnst et al. [6].

5.2. Surrogate modeling. In this subsection, we build cheap surrogates $\{\tilde{h}_{i,j}\}_{(i,j)\in\Lambda}$ to approximate the coupling functions $\{h_{i,j}\}_{(i,j)\in\Lambda}$. For simplicity, $h(\zeta)$ is used to generically denote a function dependent on a n-dimensional vector variable $\zeta = [\zeta_1, \ldots, \zeta_n]^T$, where ζ_i denotes the ith element of the vector ζ. A response surface approximation for $h(\zeta)$ is constructed by fitting a polynomial to a set of samples. For example, a linear response surface approximation with $n+1$ degrees of freedom of $h(\zeta)$ is

$$\tilde{h}(\zeta) := a_0 + \sum_{i=1}^{n} a_i \zeta_i,$$

where the coefficients $\{a_i\}_{i=1}^n$ are determined using a least squares fit to sample points.

Samples of the coupling functions $\{h_{i,j}\}_{(i,j)\in\Lambda}$ can be computed in the offline stage. As shown in Algorithm 1, for each subdomain D_i, $i = 1, \ldots, M$, a large number of local solutions $\{u(x, \xi_i^{(s)}, \tau_i^{(s)})\}_{s=1}^{N_i}$ are generated, and the values of the coupling functions $\{h_{i,j}(\xi_i^{(s)}, \tau_i^{(s)})\}_{j\in\Lambda_i}$ corresponding to the sample points $\{(\xi_i^{(s)}, \tau_i^{(s)})\}_{s=1}^{N_i}$ can be obtained by postprocessing the local solutions.

We note that there are other popular surrogate modeling methods, which for some problems may provide more accurate approximations than response surface methods (e.g., radial basis function methods [49] and Kriging methods [46]). Another option, particularly for complicated coupling function behavior, would be to use a goal-oriented model reduction technique [13]. In the numerical studies of this paper, we use the linear response surface method and the Kriging method through the MATLAB toolbox DACE [41] to construct the surrogates $\{\tilde{h}_{i,j}\}_{(i,j)\in\Lambda}$.

6. DDUQ summary. To summarize the strategies introduced in sections 4 and 5, the full approach of DDUQ comprises the following three steps:

pre-step: generate POD basis for interface functions;

offline step: generate local solution samples and construct surrogates for the coupling functions;

online step: estimate target input PDFs using surrogates and reweight offline output samples.

The pre-step is cheap, since although it performs standard domain decomposition iterations, the number of input samples is small. In the offline stage, expensive computational tasks are fully decomposed into subdomains, i.e., there is no data exchange between subdomains. The online stage is relatively cheap, since no PDE solve is required. The online stage requires density estimation, which has a computational cost that scales linearly with the sample size when using efficient density estimation techniques [36, 62]. However, in the numerical studies of this paper, we use the traditional KDE as implemented in the MATLAB KDE toolbox [32] with cost quadratically dependent on the sample size. This full approach is also summarized in Figure 2, where communication refers to data exchange between subdomains.

For the problems of interest (systems governed by PDEs), the cost of the DDUQ approach is dominated by the local PDE solves in the offline step. We perform a rough order of magnitude analysis of the computational costs by taking the cost of each local PDE solve to be C_{solve} (i.e., the costs of all local PDE solves are taken to be equal for simplicity). The dominant cost of DDUQ is the total number of local PDE solves, $\sum_{i=1}^{M} N_i C_{\text{solve}}$. If we consider equal offline sample sizes for each subdomain, $N_i = N_{\text{off}}$ for all subdomains $\{D_i\}_{i=1}^{M}$, then this cost can be written as $MN_{\text{off}}C_{\text{solve}}$.

For comparison, we consider the corresponding cost of system-level Monte Carlo combined with parallel domain decomposition for N samples. To simplify the analysis, assume that K_{DD} domain decomposition iterations are performed for each PDE solve. In practice, the number of domain decomposition iterations may vary from case to case, but they will typically be of similar magnitude. The cost of the system-level Monte Carlo is then $K_{\text{DD}}MNC_{\text{solve}}$.

For the same number of samples, $N = N_{\text{off}}$, the cost of the system-level Monte Carlo is a factor of K_{DD} times larger than the DDUQ cost. This is because the system-level Monte Carlo solves local PDEs at every domain decomposition iteration

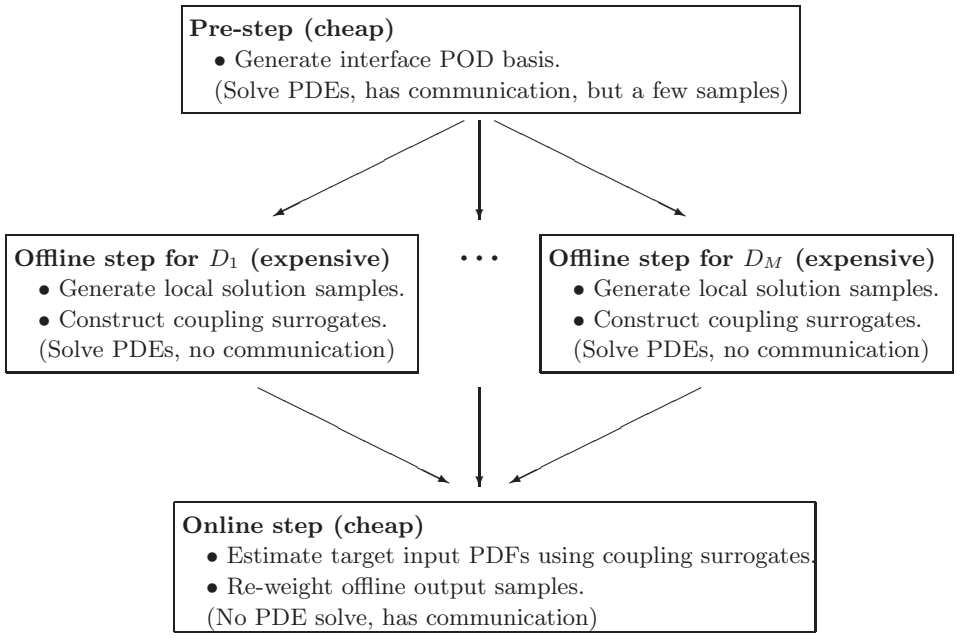


FIG. 2. *DDUQ summary.*

step. In contrast, DDUQ only solves the local PDEs once at each sample point in the offline step and uses cheap surrogates to perform the domain decomposition iterations in the online step. We note that the system-level Monte Carlo could also be made more efficient by combining surrogate models with parallel domain decomposition, but this situation is not considered in this paper. However, it is important to note that this comparison does not tell the full story of the relative computational costs of DDUQ, since setting $N = N_{\text{off}}$ will not result in the same level of accuracy between the system-level Monte Carlo and DDUQ estimates. This is because the DDUQ approach introduces a statistical inefficiency through the importance sampling step.

To complete the comparison, we consider the effective sample size [33, 40, 17] for each subdomain D_i, defined as

$$N_i^{\text{eff}} = \frac{\left(\sum_{s=1}^{N_{\text{off}}} w_i^{(s)}\right)^2}{\sum_{s=1}^{N_{\text{off}}} \left(w_i^{(s)}\right)^2}, \quad i = 1, \ldots, M. \tag{6.1}$$

The effective sample size takes on values $1 \leq N_i^{\text{eff}} \leq N_{\text{off}}$ and is a measure of the quality of the proposal distribution relative to the target distribution. The smaller the value of N_i^{eff}, the more "wasted" samples drawn from the proposal distribution. We cannot exactly quantify the effect on accuracy of the inefficiency introduced through the importance sampling; however, a roughly fair comparison is to consider a system-level Monte Carlo with $N = N^{\text{eff}} := \min_{1 \leq i \leq M} N_i^{\text{eff}}$, where by choosing the minimum effective sample size over all subdomains, we consider the worst case. Thus, our final cost comparison is $MN_{\text{off}}C_{\text{solve}}$ for DDUQ compared to $K_{\text{DD}}MN^{\text{eff}}C_{\text{solve}}$ for system-level Monte Carlo. It can be seen now that the efficiency of the DDUQ approach

depends very much on the quality of the proposals chosen for the local subdomain sampling. If the proposal distributions are good representatives of the target distributions and yield $N^{\text{eff}} \gtrsim N_{\text{off}}/K_{\text{DD}}$, then DDUQ will be computationally competitive with (or possibly even cheaper than) system-level Monte Carlo. If, however, the proposals are chosen conservatively so as to result in $N^{\text{eff}} \ll N_{\text{off}}/K_{\text{DD}}$, then the decomposition will incur a computational penalty. Even if this is the case, there may be other advantageous reasons to employ a decomposition-based approach, as discussed in section 1.

7. Numerical study. This section presents results for the DDUQ approach and compares its performance with system-level Monte Carlo.

7.1. Problem setup. We illustrate the DDUQ approach using the following diffusion problem. The governing equations considered are

$$-\nabla \cdot (a(x,\xi)\nabla u(x,\xi)) = f(x,\xi) \quad \text{in} \quad D \times \Gamma, \tag{7.1}$$

$$\frac{\partial u(x,\xi)}{\partial n} = 0 \quad \text{on} \quad \partial D_N \times \Gamma, \tag{7.2}$$

$$a(x,\xi)\frac{\partial u(x,\xi)}{\partial n} = -b(x,\xi)u \quad \text{on} \quad \partial D_R \times \Gamma, \tag{7.3}$$

where a is the permeability coefficient, b is the Robin coefficient, f is the source function, $\partial u/\partial n$ is the outward normal derivative of u on the boundaries, $\partial D_N \cap \partial D_R$ has measure zero, and $\partial D = \partial D_N \cup \partial D_R$. Note that a, b, and f are dependent on the system input vector ξ.

This stochastic diffusion equation is widely used in groundwater hydrology. As discussed in [56], modeling groundwater flow in heterogeneous media composed of multiple materials remains an active research area. Composite media models can typically be divided into the following three kinds [61]. First, the material locations are assumed to be known with certainty but the permeability within each material is set to a random field [42, 23]. Second, the locations of materials are treated as random variables but the permeability within each material is considered to be deterministic [37]. Finally, a two-scale representation of uncertainty considers both the locations of materials and the permeability within each material to be random [38, 59, 60]. In [38], a nonstationary spectral method was developed to approximate the statistics of solutions of this two-scale model. However, the spectral method proposed in [38] requires the random multimaterial distributions to have a uniform correlation structure and, as a result, does not apply to problems with highly heterogeneous media. The random domain decomposition method, proposed by Winter and Tartakovsky [59, 60], develops a general solution method for diffusion in highly heterogeneous media composed of multiple materials.

In this section, we consider each material as a local component and test two kinds of models: (1) one-scale model—fixed interfaces with random parameters within each component; (2) two-scale model—random interfaces with random parameters within each component. In the following test problems, we first consider three one-scale models posed on two-dimensional spatial domains, with two, four, and seven random parameters, respectively. Implementation details of the DDUQ approach are demonstrated through the first test problem (with two parameters), then numerical results are presented for all three test problems. Finally, we extend our approach to solve a two-scale model posed on a one-dimensional spatial domain.

We apply the Monte Carlo method to the global problem (7.1)–(7.3) for the purpose of generating reference results for comparison. The weak form of (7.1)–(7.3)

is to find $u(x,\xi) \in H^1(D) := \{u : D \to \mathbb{R}, \int_D u^2 \,\mathrm{d}D < \infty, \int_D (\partial u/\partial x_l)^2 \,\mathrm{d}D < \infty, l = 1,\ldots,d\}$ such that

$$(a\nabla u, \nabla v) + (bu, v)_{\partial D_R} = (f, v) \quad \forall v \in H^1(D). \tag{7.4}$$

A finite element approximation of (7.4) is obtained by introducing a finite dimensional space $X^h(D)$ to approximate $H^1(D)$. We discretize in space using a bilinear finite element approximation [11, 20] with a mesh size $h = 1/16$ for the two-dimensional test problems and using a linear finite element approximation with 101 grid points on each subdomain of the one-dimensional random domain decomposition test problem in section 7.8.

7.2. Two-component system with two parameters. We consider the diffusion problem posed on the spatial domain shown in Figure 1. In order to show the dimensions, we re-plot this spatial domain in Figure 3. The Neumann boundary condition (7.2) is applied on $\partial D_N := \{(1,5) \times \{0.75\}\} \cup \{(1,5) \times \{1.25\}\}$, while the other boundaries are Robin boundaries ∂D_R.

We consider a two-dimensional random parameter $\xi = [\xi_1, \xi_2]^T$. The permeability coefficient, the Robin coefficient, and the source function are defined by

$$\begin{aligned} f(x,\xi) &= \xi_1 \quad \text{when} \quad x \in [0,1] \times [0,2], \\ f(x,\xi) &= 0 \quad \text{when} \quad x \in D \setminus [0,1] \times [0,2], \\ a(x,\xi) &= 1 \quad \text{when} \quad x \in D_1, \\ a(x,\xi) &= \xi_2 \quad \text{when} \quad x \in D_2, \\ b(x,\xi) &= 1 \quad \text{when} \quad x \in \partial D_R, \end{aligned}$$

where the random variables ξ_1 and ξ_2 are specified to be independently distributed as follows: ξ_1 is a truncated Gaussian distribution with mean 100, standard deviation 1, and range $[80, 120]$, and ξ_2 is a truncated Gaussian distribution with mean 1, standard deviation 0.01, and range $[0.9, 1.1]$. We note that the convergence analysis of KDE in [27] assumes an unbounded range for the random variables. However, in our test problems we use bounded ranges for the distributions, reflecting constraints on the values assumed by the underlying physical variables. This means that for our problems, we cannot guarantee the convergence of the density estimation step in the DDUQ algorithm and thus cannot theoretically guarantee the overall convergence of the DDUQ approach as in section 4.2. Density estimation remains a limitation in the approach—both in terms of the assumptions needed for its convergence and in its lack of scalability to problems with more than a handful of random variables. We further discuss this limitation in section 8.

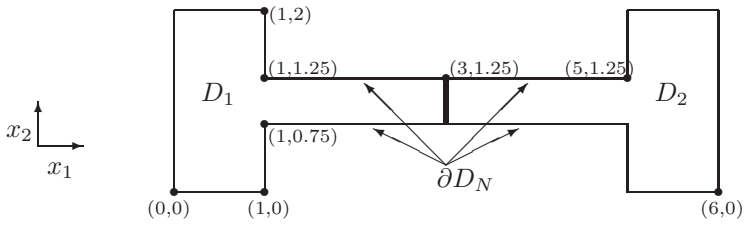


FIG. 3. *Dimensions of the spatial domain with two components.*

The outputs of interest are defined by the integrals of the solution over the left and the right boundaries of the domain, i.e.,

$$y_1(\xi) = \int_{Z_1} u(x,\xi)\,\mathrm{d}x_2, \tag{7.5}$$

$$y_2(\xi) = \int_{Z_2} u(x,\xi)\,\mathrm{d}x_2, \tag{7.6}$$

where $x = [x_1, x_2]^T$, $Z_1 := \{x | x_1 = 0, 0 \le x_2 \le 2\}$, and $Z_2 := \{x | x_1 = 6, 0 \le x_2 \le 2\}$.

In order to decompose the global problem (7.4), we use the parallel Dirichlet–Neumann domain decomposition method [45, pp. 24–25]. That is, on subdomain D_1, we solve a local problem with a Dirichlet condition posed on $\partial_2 D_1$, whereas we solve a local problem with a Neumann condition posed on $\partial_1 D_2$ for subdomain D_2. The coupling functions are

$$h_{2,1} := u\left(x, \xi_2, \tau_2\right)\big|_{\partial_1 D_2}, \tag{7.7}$$

$$h_{1,2} := \left.\frac{\partial u\left(x, \xi_1, \tau_1\right)}{\partial n}\right|_{\partial_2 D_1}. \tag{7.8}$$

See [45, p. 13] for the weak form of the coupling functions. Equation (3.7) specifies the definitions of the interface parameters τ_1 and τ_2 (since this test problem has only two subdomains, we have $\tau_1 = \tau_{2,1}$ and $\tau_2 = \tau_{1,2}$); we set the acceleration parameters to $\theta_{2,1} = 0.5$ and $\theta_{1,2} = 0$ for this problem.

7.3. DDUQ pre-step (generate POD interface bases). In numerical studies in this paper, the number of snapshots in the pre-step is set to $\tilde{N} = 10$, and the tolerance for the domain decomposition iteration is set to $tol = 10^{-6}$. We collect the 10 snapshots of each (discrete) interface function (see (3.3) and (5.1)), and compute the corresponding POD basis that represents them. The POD singular values for each interface function are plotted in Figure 4. It can be seen that the singular values decrease quickly for both interface functions and that in both cases retaining only the first singular vector to define the POD basis is sufficient to obtain accurate interface surrogate models. The fact that only one POD vector is needed for each interface function is a reflection of the simplicity of the solution along the interface for this particular problem (in this case, the solution is close to constant along the interface). Other problems may have more complicated interface behavior; this would be revealed through the POD singular values, which would decay more slowly, indicating that more basis vectors would be needed in the surrogate model. In this case, our POD surrogate reduces the dimension of the interface parameters to one, i.e., $\tau_{i,j} = c_1^{i,j}$ in section 5.1.

7.4. Implementation of the DDUQ offline and online methods. We now implement the DDUQ offline and online strategies as described in the algorithms of section 4.1. The first step of the offline stage defines a proposal PDF $p_{\xi_i,\tau_i}(\xi_i, \tau_i)$ for each subdomain D_i. In the numerical experiments in this paper, we take each proposal to be $p_{\xi_i,\tau_i}(\xi_i, \tau_i) = \pi_{\xi_i}(\xi_i) p_{\tau_i}(\tau_i)$, where $\pi_{\xi_i}(\xi_i)$ is given and $p_{\tau_i}(\tau_i)$ is chosen to be a uniform PDF. We set the range of $p_{\tau_i}(\tau_i)$ by first identifying the minimum and maximum values of the sampled interface parameters in the pre-step. We define $\tilde{\alpha}_{i,j} := \min_{s=1:10}(\tau_{i,j}^{\infty}(\xi^{(s)}))$ and $\tilde{\beta}_{i,j} := \max_{s=1:10}(\tau_{i,j}^{\infty}(\xi^{(s)}))$. We set $p_{\tau_i}(\tau_i) \sim \mathbb{U}[\alpha_{i,j}, \beta_{i,j}]$,

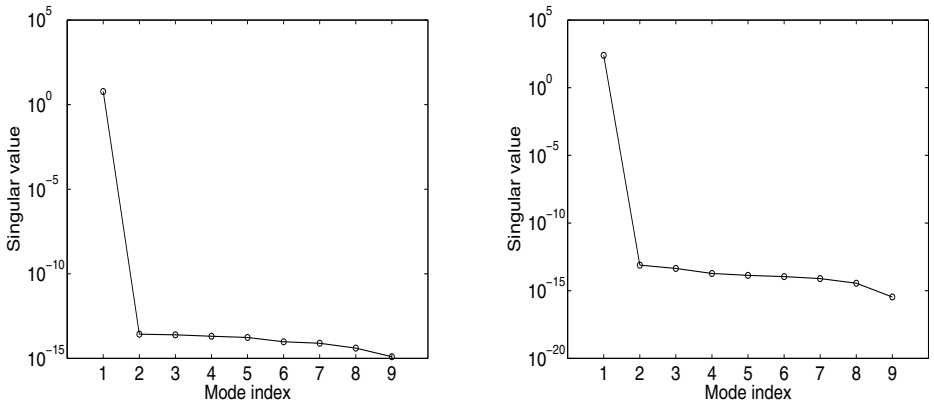


FIG. 4. *Singular values for the pre-step interface parameter matrices for the two-parameter problem: left, for interface function $g_{2,1}$, and right, for interface function $g_{1,2}$.*

with

$$\alpha_{i,j} = \tilde{\alpha}_{i,j} - tol^L_{i,j}, \quad \beta_{i,j} = \tilde{\beta}_{i,j} + tol^U_{i,j},$$

where $tol^L_{i,j}$ and $tol^U_{i,j}$ are tolerances, and $[\alpha_{i,j}, \beta_{i,j}]$ is an estimate of the range of $\tau^\infty_{i,j}$. Sharp estimates for the ranges are not necessary for DDUQ, but the estimated ranges should be conservatively wide, so that they cover the support of the (as yet unknown) corresponding target PDF (noting that the more conservative the range specified for $p_{\tau_i}(\tau_i)$, the more "wasted" samples and the lower the effective sample size). Here, we take relatively large $tol^L_{i,j}$ and $tol^U_{i,j}$, which are both set to $(\tilde{\beta}_{i,j} - \tilde{\alpha}_{i,j})$ (essentially tripling the range obtained from the 10 snapshots).

Next, we generate N_{off} samples $\{(\xi_i^{(s)}, \tau_i^{(s)})\}_{s=1}^{N_{\text{off}}}$ from the defined proposal PDF for each local subdomain D_i. We set $N_{\text{off}} = N_i = N_{\text{on}}$ to simplify the illustration. For each sample $(\xi_i^{(s)}, \tau_i^{(s)})$, we solve local problems (4.1)–(4.3) to obtain the local solution $u(x, \xi_i^{(s)}, \tau_i^{(s)})$ and its corresponding output $y_i(\xi_i^{(s)}, \tau_i^{(s)})$. The last step of the offline stage is to build coupling surrogates. That is, for all input samples, we compute the values of coupling functions $\{h_{i,j}(\xi_i, \tau_i)\}_{(i,j)\in\Lambda}$ (see (7.7)–(7.8)) and then construct the corresponding surrogates $\{\tilde{h}_{i,j}(\xi_i, \tau_i)\}_{(i,j)\in\Lambda}$ based on the methods introduced in section 5.2.

The online algorithm is stated in Algorithm 2. The first online step is to generate samples for the system input parameter ξ. In the numerical results reported in this paper, we use the same samples of ξ in both offline and online (we also tested the situation with different samples of ξ between offline and online, and no significantly different results were found). We then perform the domain decomposition iteration using the coupling surrogates $\{\tilde{h}_{i,j}(\xi_i, \tau_i)\}_{(i,j)\in\Lambda}$ instead of the coupling functions $\{h_{i,j}(\xi_i, \tau_i)\}_{(i,j)\in\Lambda}$, to obtain samples of target interface parameters. Once the target samples are obtained, target PDFs can be estimated using density estimation techniques. We use a MATLAB KDE toolbox [32] for density estimation. At the last step of the online stage, the precomputed offline output samples $\{y_i(\xi_i^{(s)}, \tau_i^{(s)})\}_{s=1}^{N_{\text{off}}}$ are reweighted by weights $\{w_i^{(s)}\}_{s=1}^{N_i}$ for each subdomain D_i (see line 16 of Algorithm 2).

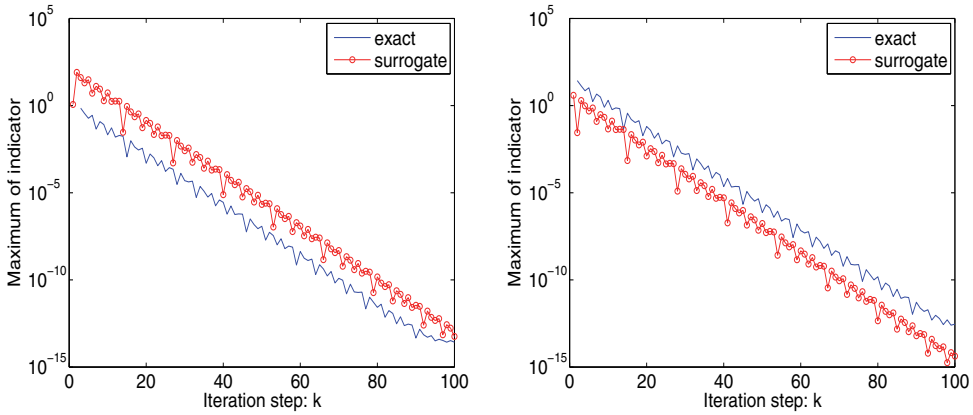
7.5. Results for two-parameter problem. First, we set $N_{\text{off}} = 10^3$ and assess the domain decomposition iteration convergence property by computing the maximum

of the error indicator, $\max_{s=1:N_{\text{off}}} |\tau_i^{k+1}(\xi^{(s)}) - \tau_i^k(\xi^{(s)})|$, $i = 1, 2$, for both the coupling functions $\{h_{i,j}(\xi_i, \tau_i)\}_{(i,j)\in\Lambda}$ and the surrogates $\{\tilde{h}_{i,j}(\xi_i, \tau_i)\}_{(i,j)\in\Lambda}$. Figure 5 shows that the error indicators associated with the exact coupling functions and the surrogate coupling functions match well and both reduce exponentially as iteration step k increases.

Samples of the joint PDF of target interface parameters for each subdomain are plotted in Figure 6. There is no visual difference between the samples generated by the coupling functions and those generated by the surrogates. In Figure 7, the target interface parameter samples (using surrogates) are again plotted and overlaid with the corresponding samples from the proposal interface parameter PDFs. The figure shows that the range of each proposal covers that of each target. It also shows the inefficiency that the decomposition introduces—many of the proposal samples (with corresponding local PDE solves) will have near-zero weights in the importance sampling reweighting process. The effective sample sizes for the two cases in Figure 7 are $N_1^{\text{eff}} = 323$ and $N_2^{\text{eff}} = 361$, respectively.

Next, we focus on the ultimate goal of the analysis: quantification of the uncertainty of the outputs of interest defined in (7.5)–(7.6). The PDF of each y_i is estimated by applying KDE to the weighted output samples generated by DDUQ. For comparison, the Monte Carlo method at the system level (solving the global problem (7.4)) with $N_{\text{ref}} = 10^6$ samples is used to generate reference results. The output samples associated with the reference solution are denoted by $\{y_1^{\text{ref}}(\xi^{(s)})\}_{s=1}^{N_{\text{ref}}}$ and $\{y_2^{\text{ref}}(\xi^{(s)})\}_{s=1}^{N_{\text{ref}}}$. By applying KDE to $\{y_i^{\text{ref}}(\xi^{(s)})\}_{s=1}^{N_{\text{ref}}}$, we obtain the reference PDFs of the outputs. Figure 8 shows that as N_{off} increases, the DDUQ estimates of the PDFs approach the reference results.

To assess the accuracy of DDUQ outputs in more detail, we consider the errors in the mean and variance estimates. The mean and variance of each output estimated using a system-level Monte Carlo simulation with N samples, denoted by



(a) Error indicator on D_1 (b) Error indicator on D_2

FIG. 5. *Maximum of the error indicator on subdomain D_1 ($\max_{s=1:N_{\text{off}}} |\tau_1^{k+1}(\xi^{(s)}) - \tau_1^k(\xi^{(s)})|$) and that on subdomain D_2 ($\max_{s=1:N_{\text{off}}} |\tau_2^{k+1}(\xi^{(s)}) - \tau_2^k(\xi^{(s)})|$), for the coupling functions (exact) and the surrogates, with $N_{\text{off}} = 10^3$.*

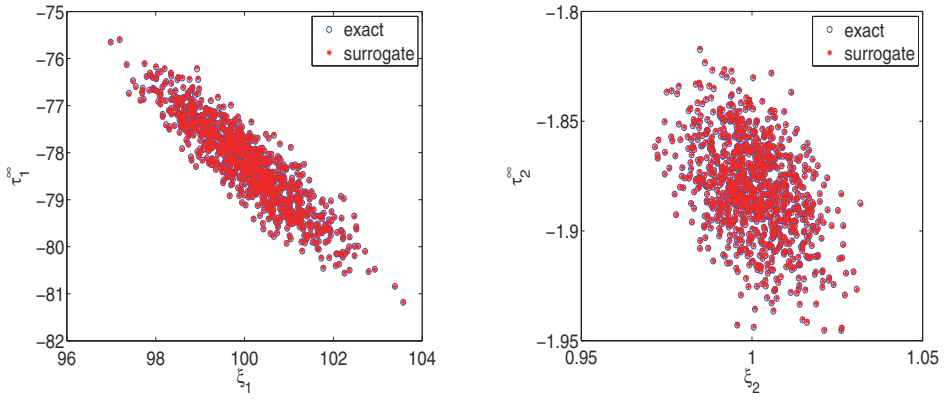


FIG. 6. *Target samples generated by coupling functions (exact) and surrogates with* $N_{\text{off}} = 10^3$.

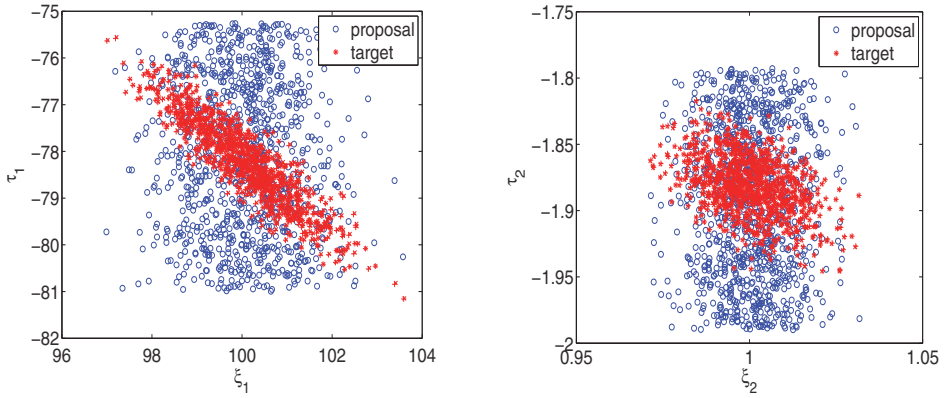


FIG. 7. *Proposal samples* (ξ_i, τ_i^0) *and target samples* (ξ_i, τ_i^∞) *with* $N_{\text{off}} = 10^3$.

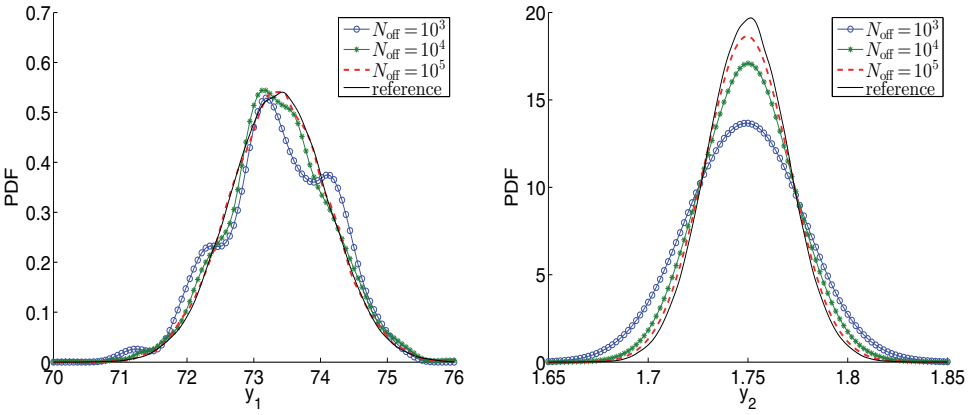


FIG. 8. *PDFs of the outputs of interest for the two-parameter test problem.*

$\{y_i^{\mathrm{MC}}(\xi^{(s)})\}_{s=1}^N$, are computed as

$$\mathbf{E}_N\left(y_i^{\mathrm{MC}}\right) := \sum_{s=1}^{N} \frac{1}{N} y_i^{\mathrm{MC}}\left(\xi^{(s)}\right), \tag{7.9}$$

$$\mathbf{V}_N\left(y_i^{\mathrm{MC}}\right) := \sum_{s=1}^{N} \frac{1}{N}\left(y_i^{\mathrm{MC}}\left(\xi^{(s)}\right) - \mathbf{E}_N\left(y_i^{\mathrm{MC}}\right)\right)^2. \tag{7.10}$$

Putting the reference samples $\{y_i^{\mathrm{ref}}(\xi^{(s)})\}_{s=1}^{N_{\mathrm{ref}}}$, $i = 1, 2$, into (7.9)–(7.10), the reference mean and variance values are obtained, which are denoted by $\mathbf{E}_{N_{\mathrm{ref}}}\left(y_i^{\mathrm{ref}}\right)$ and $\mathbf{V}_{N_{\mathrm{ref}}}\left(y_i^{\mathrm{ref}}\right)$, respectively.

The mean and variance of each output estimated using DDUQ are computed as

$$\mathbf{E}_{w,N_{\mathrm{off}}}\left(y_i\right) := \frac{\sum_{s=1}^{N_{\mathrm{off}}} w_i^{(s)} y_i\left(\xi_i^{(s)}, \tau_i^{(s)}\right)}{\sum_{s=1}^{N_{\mathrm{off}}} w_i^{(s)}},$$

$$\mathbf{V}_{w,N_{\mathrm{off}}}\left(y_i\right) := \frac{\sum_{s=1}^{N_{\mathrm{off}}} w_i^{(s)}\left(y_i\left(\xi_i^{(s)}, \tau_i^{(s)}\right) - \mathbf{E}_{w,N_{\mathrm{off}}}\left(y_i\right)\right)^2}{\sum_{s=1}^{N_{\mathrm{off}}} w_i^{(s)}}.$$

In order to assess the errors of DDUQ in estimating the mean and variance, the following quantities are introduced:

$$\epsilon_i := \left|\left(\mathbf{E}_{w,N_{\mathrm{off}}}\left(y_i\right) - \mathbf{E}_{N_{\mathrm{ref}}}\left(y_i^{\mathrm{ref}}\right)\right) \Big/ \mathbf{E}_{N_{\mathrm{ref}}}\left(y_i^{\mathrm{ref}}\right)\right|,$$

$$\eta_i := \left|\left(\mathbf{V}_{w,N_{\mathrm{off}}}\left(y_i\right) - \mathbf{V}_{N_{\mathrm{ref}}}\left(y_i^{\mathrm{ref}}\right)\right) \Big/ \mathbf{V}_{N_{\mathrm{ref}}}\left(y_i^{\mathrm{ref}}\right)\right|.$$

Moreover, for $N < N_{\mathrm{ref}}$, errors of the system-level Monte Carlo simulation are measured by

$$\hat{\epsilon}_i := \left|\left(\mathbf{E}_N\left(y_i^{\mathrm{MC}}\right) - \mathbf{E}_{N_{\mathrm{ref}}}\left(y_i^{\mathrm{ref}}\right)\right) \Big/ \mathbf{E}_{N_{\mathrm{ref}}}\left(y_i^{\mathrm{ref}}\right)\right|,$$

$$\hat{\eta}_i := \left|\left(\mathbf{V}_N\left(y_i^{\mathrm{MC}}\right) - \mathbf{V}_{N_{\mathrm{ref}}}\left(y_i^{\mathrm{ref}}\right)\right) \Big/ \mathbf{V}_{N_{\mathrm{ref}}}\left(y_i^{\mathrm{ref}}\right)\right|.$$

Fixing the reference mean $\mathbf{E}_{N_{\mathrm{ref}}}(y_i^{\mathrm{ref}})$ and variance $\mathbf{V}_{N_{\mathrm{ref}}}(y_i^{\mathrm{ref}})$ for each output, we repeat the DDUQ process 30 times for each N_{off} and compute the averages of ϵ_i and η_i. These averages are denoted by $\mathbf{E}(\epsilon_i)$ and $\mathbf{E}(\eta_i)$, respectively. In addition, we repeat the system-level Monte Carlo simulation 30 times for $N < N_{\mathrm{ref}}$ and denote the average mean and variance errors by $\mathbf{E}(\hat{\epsilon}_i)$ and $\mathbf{E}(\hat{\eta}_i)$. As discussed in section 6, for a given sample size ($N = N_{\mathrm{off}}$), the system-level Monte Carlo combined with parallel domain decomposition requires more local PDE solves than DDUQ, since the system-level Monte Carlo solves local PDEs at every domain decomposition iteration step. In order to make a fair comparison, we consider two cases: (a) comparing DDUQ and system-level Monte Carlo with the same number of samples ($N = N_{\mathrm{off}}$); (b) comparing them with the same number of local PDE solves (the system-level Monte Carlo combined with parallel domain decomposition then has smaller sample sizes, $N < N_{\mathrm{off}}$). When counting the number of local PDE solves for DDUQ, we only

consider the local solves in the offline step without counting the number of solves in the pre-step, since the pre-step just has a small number of PDE solves (here just 10 snapshots were generated).

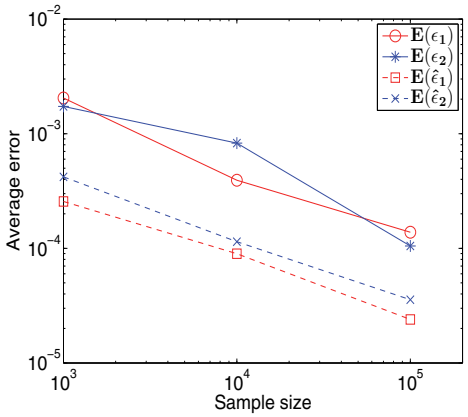
Figure 9 shows the average mean and variance errors for this test problem. First, we focus on the comparison based on sample sizes, which are shown in Figure 9(a) and Figure 9(c). It can be seen that all errors reduce as the sample size increases, and although the errors of the system-level Monte Carlo are smaller than that of DDUQ, they reduce at nearly the same rate of $\sqrt{N_{\text{off}}}$ (N_{off} is the sample size) for this test problem. We also see that the DDUQ estimates have errors roughly a factor of five times that of the system-level Monte Carlo for a given sample size. For example, if we want to achieve an accuracy in estimating the mean with error smaller than 10^{-4}, 10^5 samples are required for DDUQ, while only 10^4 samples are required for the system-level Monte Carlo. The larger errors obtained by the DDUQ approach are caused largely by the sampling inefficiency introduced by the importance sampling step. This inefficency can be seen in Figure 7, where the proposal samples lying in regions of low target probability have small weights (essentially zero) in the online reweighting stage. As a result, these samples play little role in estimating the output PDFs and output statistics. For this test problem, the averages (from the 30 repeats) of the overall effective sample sizes N^{eff} (defined in section 6) associated with $N_{\text{off}} = 10^3, 10^4, 10^5$ are $313, 2761, 25, 411$, respectively.

Next, we focus on the comparisons based on the number of local PDE solves, which are shown in Figure 9(b) and Figure 9(d). For the same number of local PDE solves, it can be seen that the errors in the mean estimates of DDUQ are smaller than that of the system-level Monte Carlo. The errors in the variance estimates of both methods are similar. From Figure 5, in order to reach the stopping criterion of $tol = 10^{-6}$, the parallel Dirichlet–Neumann method requires around 50 iterations for this test problem, i.e., $K_{\text{DD}} \approx 50$. So, DDUQ has around 50 times more samples than that of the system-level Monte Carlo in this situation. Again, it is important to note that the position of the DDUQ error curves relative to the system-level Monte Carlo results depends on the efficiency of the proposal distributions. The results in Figure 9 show that even with our conservative choice of proposal distribution, the computational penalty of the DDUQ approach for this problem is small to nonexistent.

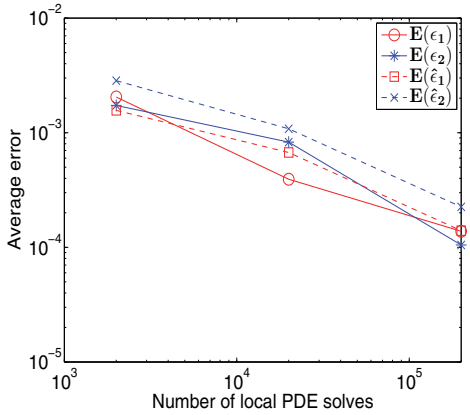
7.6. Two-component system with four parameters. We again consider the diffusion problem posed on the spatial domain shown in Figure 3, where we take the same definitions of ∂D_N and ∂D_R as specified in section 7.2. We still decompose $\xi = [\xi_1^T, \xi_2^T]^T$, and now each ξ_i is a two-dimensional vector, $\xi_i = [\xi_{i,1}, \xi_{i,2}]^T$, $i = 1, 2$. The permeability coefficient, the Robin coefficient, and the source function are now defined by

$$\begin{aligned}
f(x, \xi) &= \xi_{1,1} \quad \text{when} \quad x \in [0,1] \times [0,2], \\
f(x, \xi) &= 0 \quad \text{when} \quad x \in D \setminus [0,1] \times [0,2], \\
a(x, \xi) &= \xi_{1,2} \quad \text{when} \quad x \in D_1, \\
a(x, \xi) &= \xi_{2,1} \quad \text{when} \quad x \in D_2, \\
b(x, \xi) &= \xi_{2,2} \quad \text{when} \quad x \in [5,6] \times \{0\}, \\
b(x, \xi) &= 1 \quad \text{when} \quad x \in \partial D_R \setminus [5,6] \times \{0\},
\end{aligned}$$

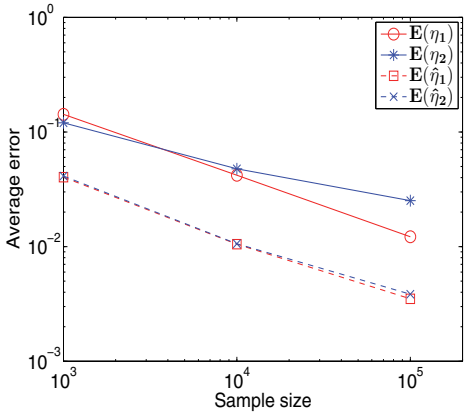
where the random variables $\{\xi_{i,j}\}_{i,j=1:2}$ are specified to be independently distributed as follows: $\xi_{1,1}$ is a truncated Gaussian distribution with mean 100, standard deviation 1, and range $[80, 120]$; $\xi_{1,2}$ and $\xi_{2,1}$ are both truncated Gaussian distributions with mean



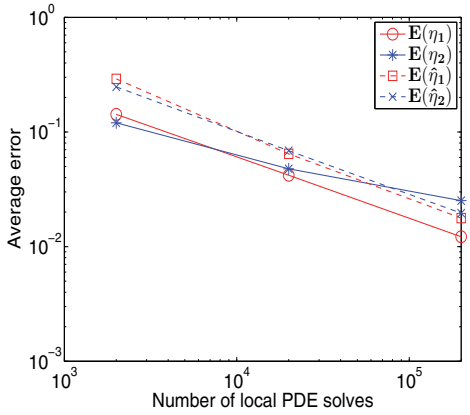
(a) Mean errors w.r.t. sample sizes



(b) Mean errors w.r.t. # of PDE solves



(c) Variance errors w.r.t. sample sizes



(d) Variance errors w.r.t. # of PDE solves

FIG. 9. *Average DDUQ errors in output mean and variance estimates for output y_1 ($\mathbf{E}(\epsilon_1)$ and $\mathbf{E}(\eta_1)$) and output y_2 ($\mathbf{E}(\epsilon_2)$ and $\mathbf{E}(\eta_2)$) are compared to the average errors in mean and variance estimates computed using system-level Monte Carlo ($\mathbf{E}(\hat{\epsilon}_1)$, $\mathbf{E}(\hat{\eta}_1)$, $\mathbf{E}(\hat{\epsilon}_2)$, and $\mathbf{E}(\hat{\eta}_2)$) for the two-parameter test problem.*

1, standard deviation 0.01, and range $[0.9, 1.1]$; $\xi_{2,2}$ is a truncated Gaussian distribution with mean 1, standard deviation 0.1, and range $[0.5, 1.5]$. The outputs of this problem are also defined by (7.5)–(7.6).

In the pre-step, we take $\tilde{N} = 10$ samples and collect only the first singular vector for constructing the POD basis (see section 7.3 for details). In the offline step, the linear response surface method is used to construct the coupling surrogates for this test problem. The PDFs of the outputs of this problem are shown in Figure 10, where we see that as N_{off} increases the PDFs generated by DDUQ approach the reference PDFs (generated using the system-level Monte Carlo simulation with $N_{\text{ref}} = 10^6$ samples). Figure 11 shows the errors for this test problem. We see that trends in the errors in the mean and variance estimates are similar to those observed for the two-parameter test problem. Figure 11(a) and Figure 11(c) show that again system-level Monte Carlo has smaller errors for the same sample size, while Figure 11(b) and Figure 11(d) show that DDUQ competes favorably in computational cost when considering the errors with respect to the number of local PDE solves. For this test problem, the

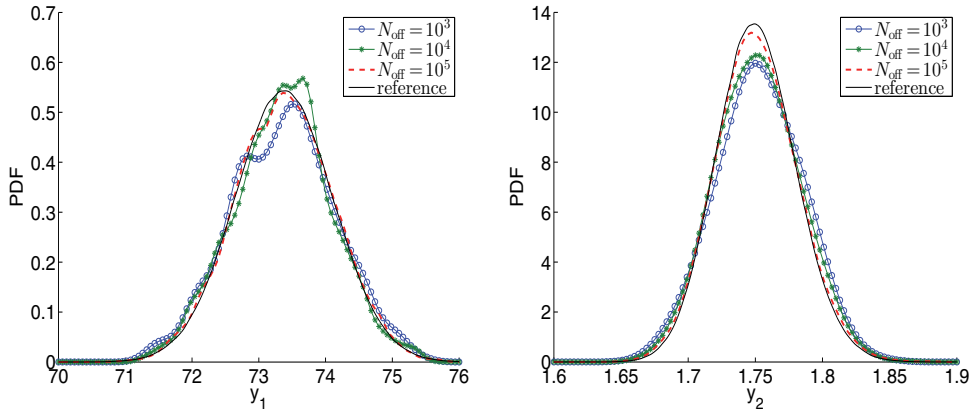


FIG. 10. *PDFs of the outputs of interest for the four-parameter test problem.*

averages of the overall effective sample sizes N^{eff} associated with $N_{\text{off}} = 10^3, 10^4, 10^5$ are $155, 1391, 13, 309$, respectively.

7.7. Three-component system with seven parameters. The diffusion problem is now posed on the spatial domain shown in Figure 12, which consists of three components. Robin boundaries ∂D_R are marked in red in Figure 12, and a homogeneous Neumann boundary condition is applied on the other boundaries. The Robin coefficient is set to $b = 1$ for $x \in \partial D_R$, while the source function $f = 1$ for $x \in [5, 6] \times [0, 2]$ and $f = 0$ for the other part of the spatial domain.

On each subdomain D_i, $i = 1, 2, 3$, the permeability coefficient $a(x, \xi)$ is assumed to be a random field with mean function $a_{i,0}(x)$, constant standard deviation σ, and covariance function $C(x, x')$,

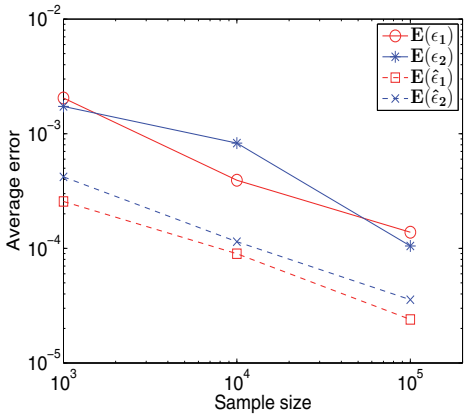
$$C(x, x') = \sigma^2 \exp\left(-\frac{|x_1 - x_1'|}{c} - \frac{|x_2 - x_2'|}{c}\right), \tag{7.11}$$

where $x = [x_1, x_2]^T$, $x' = [x_1', x_2']^T$, and c is the correlation length. The random fields are assumed to be independent between different subdomains. Here, we set $a_{1,0}(x) = 1$, $a_{2,0}(x) = 5$, $a_{3,0}(x) = 1$, $\sigma = 0.5$, and $c = 20$. Each random field can be approximated by a truncated KL expansion [9, 19, 24],

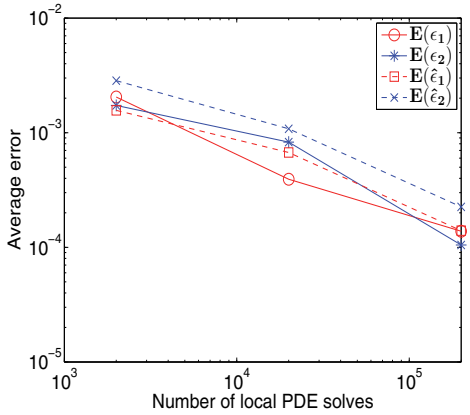
$$a(x, \xi)|_{D_i} \approx a_{i,0}(x) + \sum_{k=1}^{\mathfrak{N}_i} \sqrt{\lambda_{i,k}}\, a_{i,k}(x) \xi_{i,k}, \quad i = 1, 2, 3, \tag{7.12}$$

where $\{a_{i,k}(x)\}_{k=1}^{\mathfrak{N}_i}$ and $\{\lambda_{i,k}\}_{k=1}^{\mathfrak{N}_i}$ are the eigenfunctions and eigenvalues of (7.11) posed on each subdomain D_i, $\mathfrak{N}_i$ is the number of KL modes retained for subdomain D_i, and $\{\xi_{i,k} : i = 1, \ldots, M \text{ and } k = 1, \ldots, \mathfrak{N}_i\}$ are uncorrelated random variables. In this paper, we set the random variables $\{\xi_{i,k} : i = 1, \ldots, M \text{ and } k = 1, \ldots, \mathfrak{N}_i\}$ to be independent truncated Gaussian distributions with mean 0, standard deviation 0.5, and range $[-1, 1]$.

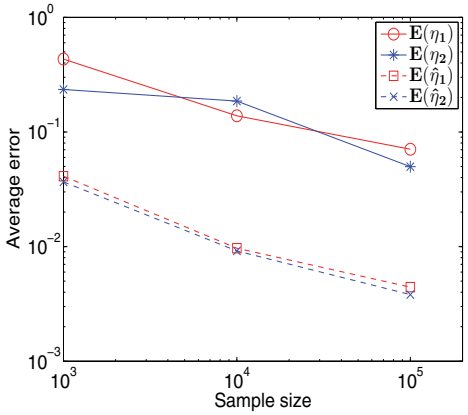
The error of each truncated KL expansion depends on the amount of total variance captured, $\delta_i := (\sum_{k=1}^{\mathfrak{N}_i} \lambda_{i,k})/(|D_i|\sigma^2)$, where $|D_i|$ denotes the area of subdomain D_i [12, 48]. We choose $\mathfrak{N}_1 = 2$, $\mathfrak{N}_2 = 3$, and $\mathfrak{N}_3 = 2$, so that each $\delta_i > 95\%$. This gives a total of seven parameters in the input vector ξ.



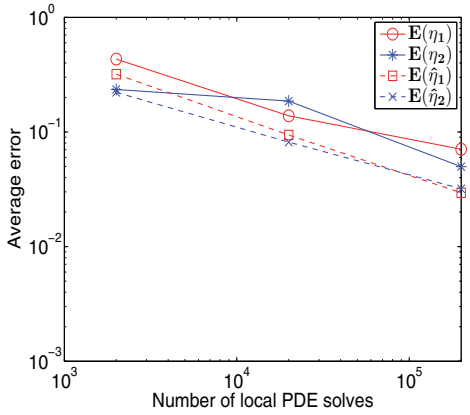
(a) Mean errors w.r.t. sample sizes



(b) Mean errors w.r.t. # of PDE solves



(c) Variance errors w.r.t. sample sizes



(d) Variance errors w.r.t. # of PDE solves

FIG. 11. *Average DDUQ errors in output mean and variance estimates for output* y_1 ($\mathbf{E}(\epsilon_1)$ *and* $\mathbf{E}(\eta_1)$) *and output* y_2 ($\mathbf{E}(\epsilon_2)$ *and* $\mathbf{E}(\eta_2)$) *are compared to the average errors in mean and variance estimates computed using system-level Monte Carlo* ($\mathbf{E}(\hat{\epsilon}_1)$, $\mathbf{E}(\hat{\eta}_1)$, $\mathbf{E}(\hat{\epsilon}_2)$, *and* $\mathbf{E}(\hat{\eta}_2)$) *for the four-parameter test problem.*

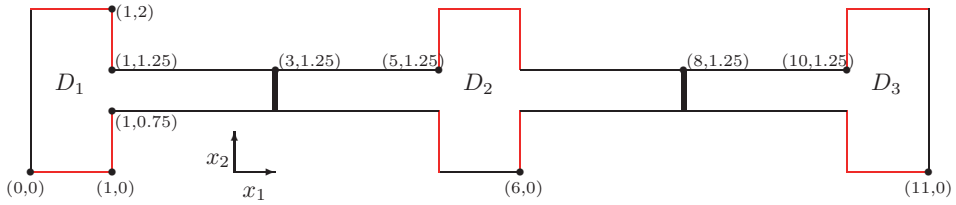


FIG. 12. *Three-component system.*

The outputs of this problem are

$$y_1(\xi) = \int_{Z_1} u(x,\xi)\,\mathrm{d}x_2, \tag{7.13}$$

$$y_2(\xi) = \int_{Z_2} u(x,\xi)\,\mathrm{d}x_1, \tag{7.14}$$

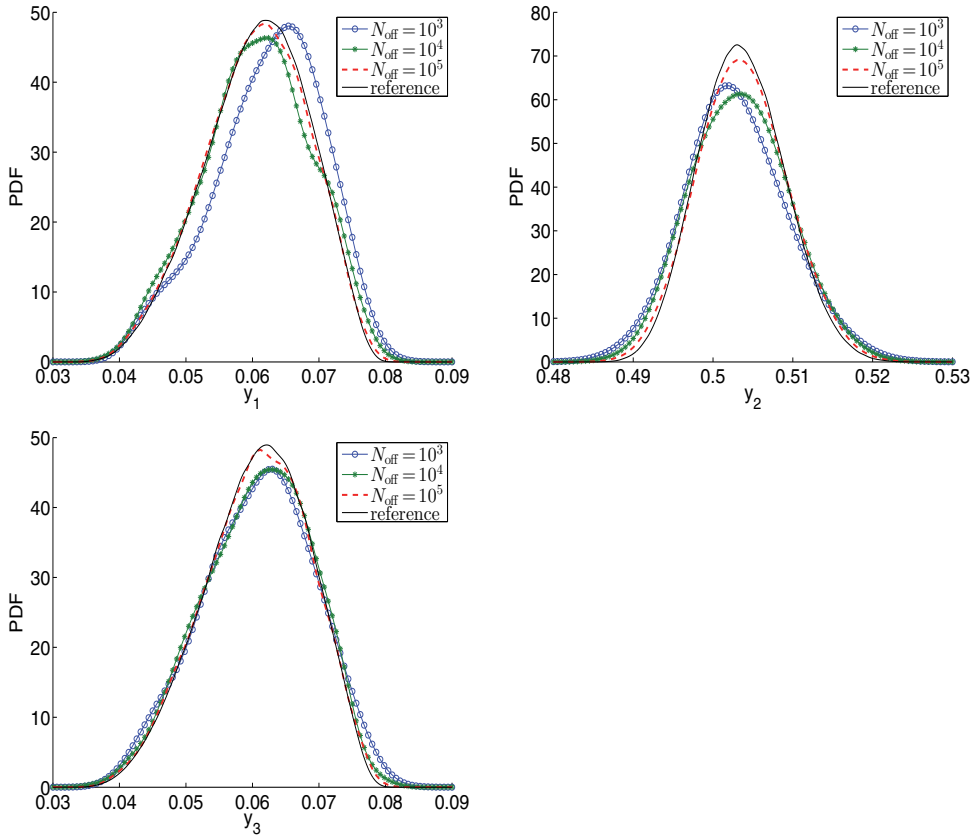
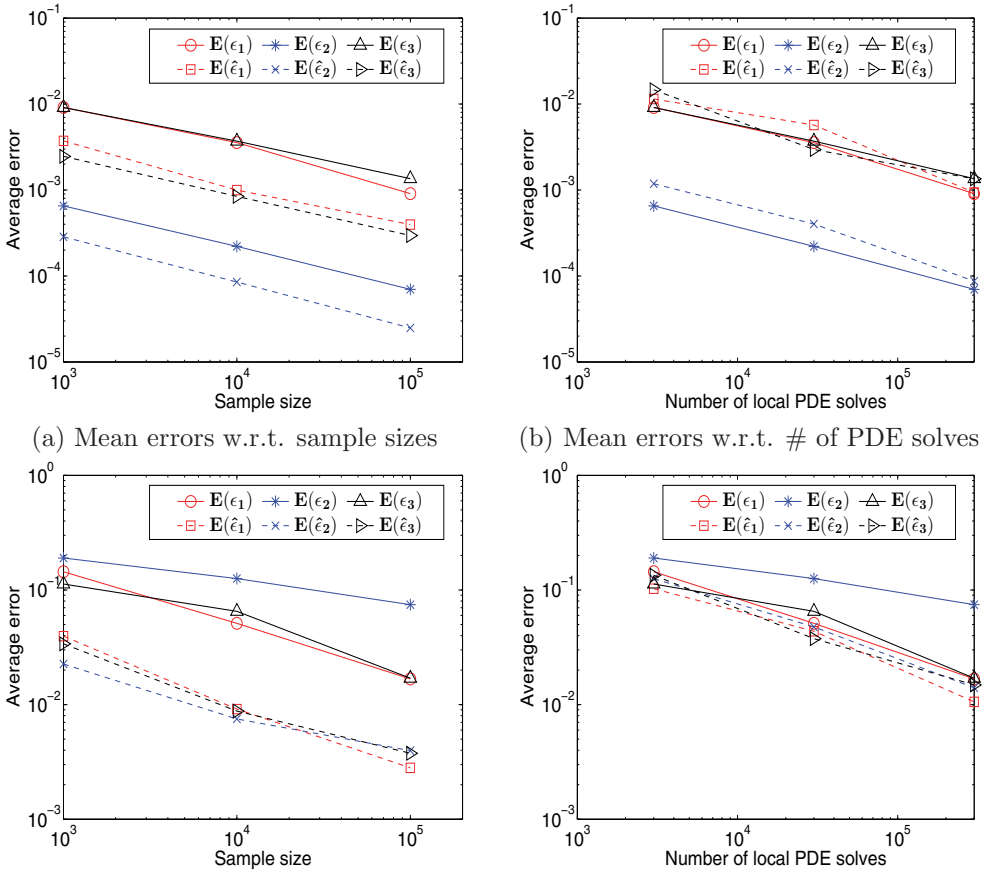


FIG. 13. *PDFs of the outputs of interest for the seven-parameter test problem.*

$$y_3(\xi) = \int_{Z_3} u(x, \xi)\, \mathrm{d}x_2, \tag{7.15}$$

where $x = [x_1, x_2]^T$, $Z_1 := \{x |\ x_1 = 0,\ 0 \le x_2 \le 2\}$, $Z_2 := \{x |\ 5 \le x_1 \le 6,\ x_2 = 0\}$, and $Z_3 := \{x |\ x_1 = 11,\ 0 \le x_2 \le 2\}$.

Ten samples are again used in the DDUQ pre-step, and the resulting POD bases for the interface functions each contain only one basis vector. In this test problem, we use a Kriging model to build surrogates of the coupling functions, constructed with 10^3 samples through the DACE Kriging toolbox [41]. A reference solution is again generated using the system-level Monte Carlo method with $N_{\text{ref}} = 10^6$ samples. The PDFs generated by DDUQ are shown in Figure 13. Again, the results of the DDUQ and the system-level Monte Carlo match well when N_{off} approaches N_{ref}. Figure 14(a) and Figure 14(c) show that the errors in the mean and variance estimates for this test problem reduce as the sample size increases, with system-level Monte Carlo giving more accurate results for the same sample size, as expected. Figure 14(b) and Figure 14(d) show that DDUQ and system-level Monte Carlo with the same number of local PDE solves have similar errors, except the errors of the variance estimate of y_2. For this test problem, the averages of the overall effective sample sizes N^{eff} associated with $N_{\text{off}} = 10^3, 10^4, 10^5$ are $150, 1545, 12,679$, respectively.



(a) Mean errors w.r.t. sample sizes

(b) Mean errors w.r.t. # of PDE solves

(c) Variance errors w.r.t. sample sizes

(d) Variance errors w.r.t. # of PDE solves

FIG. 14. *Average DDUQ errors in output mean and variance estimates for each output y_i ($\mathbf{E}(\epsilon_i)$ and $\mathbf{E}(\eta_i)$, $i = 1, 2, 3$) are compared to the average errors in mean and variance estimates computed using system-level Monte Carlo ($\mathbf{E}(\hat{\epsilon}_i)$ and $\mathbf{E}(\hat{\eta}_i)$, $i = 1, 2, 3$) for the seven-parameter test problem.*

7.8. Two-component system with a random interface. We extend our approach to consider the random domain decomposition model [59, 60]. The diffusion equation (7.1) is now posed on the one-dimensional domain shown in Figure 15, where the location of the interface is described by a uniform random variable β with range $[-0.2, 0.2]$. The permeability coefficient is set to $a = \xi_1$ for $x \in D_1$ and $a = \xi_2$ for $x \in D_2$, where ξ_1 and ξ_2 are independent uniform distributions with ranges $[0.8, 1.2]$ and $[8, 12]$, respectively. The source function is set to $f = 1$ throughout the spatial domain and a homogeneous Dirichlet boundary condition $u = 0$ is applied on $x = -1$ and $x = 1$. The global solution $u(x, \xi, \beta)$ now depends on the uncertain variables $\xi = [\xi_1, \xi_2]^T$ and β. The outputs of interest are defined by

$$y_1(\xi, \beta) := u(-0.1, \xi, \beta), \quad y_2(\xi, \beta) := u(0.1, \xi, \beta). \tag{7.16}$$

For each subdomain D_i $(i = 1, 2)$, the local solution is denoted by $u(x, \xi_i, \beta, \tau_i)$, where τ_i is the interface parameter defined in section 3. Since the range of β covers the points $x = -0.1$ and $x = 0.1$ where the outputs are evaluated, each local solution has

FIG. 15. *Two different materials with a random interface* β.

the possibility to provide both outputs, and we denote the outputs of local systems by

$$y_{1,1} = u(-0.1, \xi_1, \beta, \tau_1), \tag{7.17}$$

$$y_{1,2} = u(0.1, \xi_1, \beta, \tau_1), \tag{7.18}$$

$$y_{2,1} = u(-0.1, \xi_2, \beta, \tau_2), \tag{7.19}$$

$$y_{2,2} = u(0.1, \xi_2, \beta, \tau_2). \tag{7.20}$$

The DDUQ offline input samples are denoted by $\{(\xi_i^{(s)}, \beta^{(s)}, \tau_i^{(s)})\}_{s=1}^{N_i}$, $i = 1, 2$, where samples of β can be different for different local systems. For each input sample, the outputs are obtained based on the following conditions: for subdomain D_1, the output $y_{1,1}$ is obtained when $\beta \geq -0.1$, and $y_{1,2}$ is obtained when $\beta \geq 0.1$; for subdomain D_2, $y_{2,1}$ is obtained when $\beta \leq -0.1$, and $y_{2,2}$ is obtained when $\beta \leq 0.1$. We again use the Kriging method with 10^3 samples to build surrogates of the coupling functions (we note that the exact coupling functions can be written explicitly for this test problem, but we use surrogates for the purpose of illustrating our methodology).

The target output PDF estimated through the reweighted samples (see section 4.1) of each output $y_{i,j}$, $i, j = 1, 2$, is denoted by $\hat{\pi}_{y_{i,j}}(y_{i,j})$, while the estimated mean and variance are denoted by $\mathbf{E}(y_{i,j})$ and $\mathbf{V}(y_{i,j})$.

As introduced in [59], the estimated PDF of y_i, $i = 1, 2$, can be obtained through the combination of contributions from both subdomains:

$$\hat{\pi}_{y_1}(y_1) = P(\beta \geq -0.1)\hat{\pi}_{y_{1,1}}(y_{1,1}) + P(\beta \leq -0.1)\hat{\pi}_{y_{2,1}}(y_{2,1}), \tag{7.21}$$

$$\hat{\pi}_{y_2}(y_2) = P(\beta \geq 0.1)\hat{\pi}_{y_{1,2}}(y_{1,2}) + P(\beta \leq 0.1)\hat{\pi}_{y_{2,2}}(y_{2,2}), \tag{7.22}$$

where P denotes the probability measure, and $P(\beta \geq -0.1) = P(\beta \leq 0.1) = 3/4$ and $P(\beta \leq -0.1) = P(\beta \geq 0.1) = 1/4$ for this test problem. Similarly, the estimated mean and variance of y_i can be computed [59]:

$$\begin{aligned}
\mathbf{E}(y_1) &= P(\beta \geq -0.1)\mathbf{E}(y_{1,1}) + P(\beta \leq -0.1)\mathbf{E}(y_{2,1}),\\
\mathbf{V}(y_1) &= P(\beta \geq -0.1)\mathbf{V}(y_{1,1}) + P(\beta \leq -0.1)\mathbf{V}(y_{2,1})\\
&\quad + P(\beta \geq -0.1)P(\beta \leq -0.1)\left(\mathbf{E}(y_{1,1}) - \mathbf{E}(y_{2,1})\right)^2,\\
\mathbf{E}(y_2) &= P(\beta \geq 0.1)\mathbf{E}(y_{1,2}) + P(\beta \leq 0.1)\mathbf{E}(y_{2,2}),\\
\mathbf{V}(y_2) &= P(\beta \geq 0.1)\mathbf{V}(y_{1,2}) + P(\beta \leq 0.1)\mathbf{V}(y_{2,2})\\
&\quad + P(\beta \geq 0.1)P(\beta \leq 0.1)\left(\mathbf{E}(y_{1,2}) - \mathbf{E}(y_{2,2})\right)^2.
\end{aligned}$$

For comparison, a reference solution is obtained by system-level Monte Carlo with 10^6 samples. Figure 16 shows that the PDFs generated by DDUQ approach the reference PDFs as the sample size increases. Figure 17 shows the errors in the mean and variance estimates for this random domain decomposition problem. Again, as expected DDUQ has larger errors than those of the system-level Monte Carlo when considering errors with respect to sample sizes. However, when considering errors with respect to the number of local PDE solves, DDUQ is again competitive.

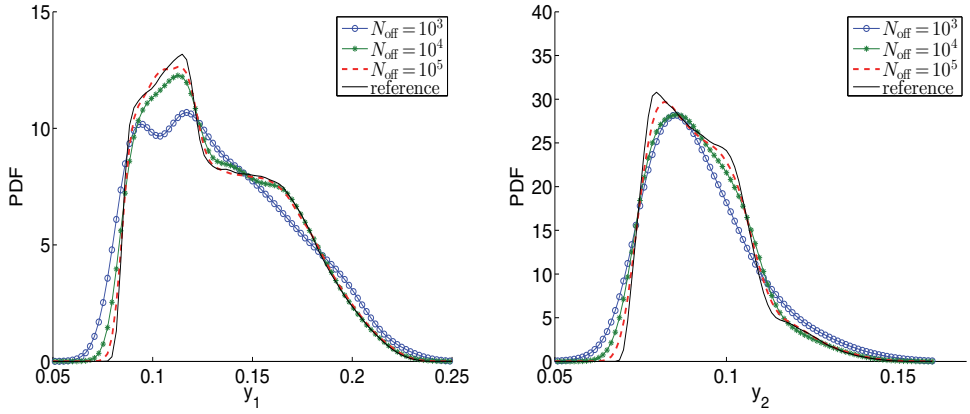
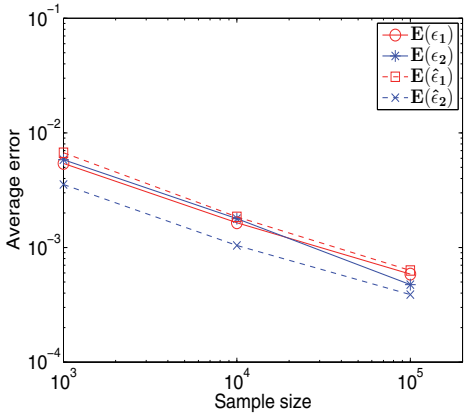
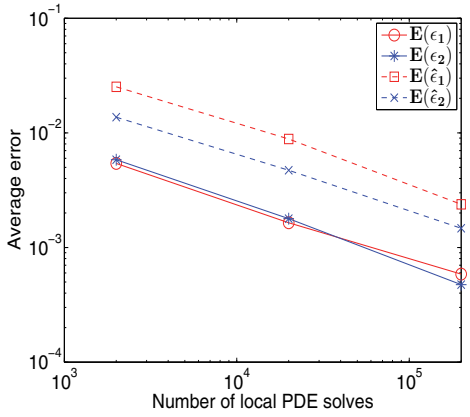


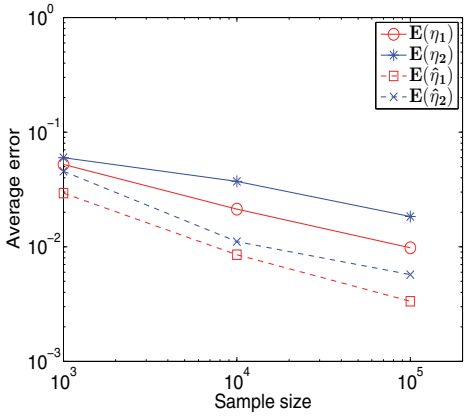
FIG. 16. *PDFs of the outputs of interest for the random domain decomposition test problem.*



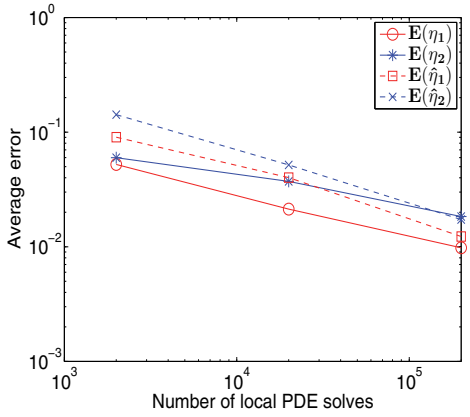
(a) Mean errors w.r.t. sample sizes



(b) Mean errors w.r.t. # of PDE solves



(c) Variance errors w.r.t. sample sizes



(d) Variance errors w.r.t. # of PDE solves

FIG. 17. *Average DDUQ errors in output mean and variance estimates for output* y_1 *(*$\mathbf{E}(\epsilon_1)$ *and* $\mathbf{E}(\eta_1)$*) and output* y_2 *(*$\mathbf{E}(\epsilon_2)$ *and* $\mathbf{E}(\eta_2)$*) are compared to the average errors in mean and variance estimates computed using system-level Monte Carlo (*$\mathbf{E}(\hat{\epsilon}_1)$*,* $\mathbf{E}(\hat{\eta}_1)$*,* $\mathbf{E}(\hat{\epsilon}_2)$*, and* $\mathbf{E}(\hat{\eta}_2)$*) for the random domain decomposition test problem.*

Finally, we note that it remains an open question for applying the DDUQ approach to random domain decomposition models where the KL expansion (7.12) is used to approximate random permeability fields. The eigenfunctions and the eigenvalues in (7.12) depend on the locations of the interfaces. Therefore, in order to capture 95% of the total variance, the dimension of each local system input ξ_i may vary with different interface locations [39]. Thus, it remains a challenging problem to specify proposal input distributions in the DDUQ offline step for this situation.

8. Concluding remarks. This paper has presented a new decomposed approach to uncertainty quantification. A combination of domain decomposition and importance sampling permits uncertainty analyses to be conducted entirely at the local subsystem level in an offline phase. The specific method presented here uses domain decomposition to divide the system into local subdomains; however, the general strategy can be extended to other decomposition strategies, including those that partition the system along disciplinary lines. Our method lays a foundation for uncertainty quantification to become an integral part of complex system simulations. It supports the vision of having all disciplinary/subsystem analyses accompanied by a corresponding local uncertainty analysis. Then, just as local analysis results are combined to form a system simulation, precomputed local uncertainty information can be manipulated to achieve the overall system uncertainty assessment.

In theory our approach is scalable to complex systems with many subcomponents and high-dimensional uncertain parameters. In practice, the method suffers from two computational bottlenecks. The first is the density estimation step, needed to construct the target probability densities, which are used to compute the importance sampling weights. We employ kernel density estimation and use its convergence properties to establish convergence of our DDUQ method. However, this density estimation step limits us to problems with coupling parameters of dimension less than about 10. Dimension reduction of the coupling parameters has proved an effective strategy to overcome this limitation for the problems studied, although the bottleneck remains. The second shortcoming of the DDUQ method is in the pre-step, which requires a handful of full coupled system simulations to generate the POD basis for the interface functions. Our current efforts are focused on alternative strategies to avoid these simulations, so that the method becomes fully decomposed.

REFERENCES

[1] F. J. Alexander, A. L. Garcia, and D. M. Tartakovsky, *Algorithm refinement for stochastic partial differential equations:* 1. *Linear diffusion*, J. Comput. Phys., 182 (2002), pp. 47–66.

[2] F. J. Alexander, A. L. Garcia, and D. M. Tartakovsky, *Algorithm refinement for stochastic partial differential equations:* II. *Correlated systems*, J. Comput. Phys., 207 (2005), pp. 769–787.

[3] F. J. Alexander, A. L. Garcia, and D. M. Tartakovsky, *Noise in algorithm refinement methods*, Comput. Sci. Eng., 7 (2005), pp. 32–38.

[4] S. Amaral, D. Allaire, and K. Willcox, *A decomposition-based approach to uncertainty analysis of feed-forward multicomponent systems*, Internat. J. Numer. Methods Engrg., 100 (2014), pp. 982–1005.

[5] H. Antil, M. Heinkenschloss, R. H. W. Hoppe, and D. C. Sorensen, *Domain decomposition and model reduction for the numerical solution of PDE constrained optimization problems with localized optimization variables*, Comput. Vis. Sci., 13 (2010), pp. 249–264.

[6] M. Arnst, R. Ghanem, E. Phipps, and J. Red-Horse, *Dimension reduction in stochastic modeling of coupled problems*, Internat. J. Numer. Methods Engrg., 92 (2012), pp. 940–968.

[7] M. Arnst, R. Ghanem, E. Phipps, and J. Red-Horse, *Measure transformation and efficient quadrature in reduced-dimensional stochastic modeling of coupled problems*, Internat. J. Numer. Methods Engrg., 92 (2012), pp. 1044–1080.
[8] M. Arnst, R. Ghanem, E. Phipps, and J. Red-Horse, *Reduced chaos expansions with random coefficients in reduced-dimensional stochastic modeling of coupled problems*, Internat. J. Numer. Methods Engrg., 97 (2014), pp. 352–376.
[9] I. Babuška, F. Nobile, and R. Tempone, *A stochastic collocation method for elliptic partial differential equations with random input data*, SIAM J. Numer. Anal., 45 (2007), pp. 1005–1034.
[10] R. Bartlett, M. Heinkenschloss, D. Ridzal, and B. van Bloemen Waanders, *Domain decomposition methods for advection dominated linear-quadratic elliptic optimal control problems*, Comput. Methods Appl. Mech. Engrg., 195 (2006), pp. 6428–6447.
[11] D. Braess, *Finite Elements*, Cambridge University Press, London, 1997.
[12] J. L. Brown, Jr., *Mean square truncation error in series expansions of random functions*, J. SIAM, 8 (1960), pp. 28–32.
[13] T. Bui-Thanh, K. Willcox, and O. Ghattas, *Goal-oriented, model-constrained optimization for reduction of large-scale systems*, J. Comput. Phys., 224 (2007), pp. 880–896.
[14] R. E. Caflisch, *Monte Carlo and quasi-Monte Carlo methods*, Acta Numer., 7 (1998), pp. 1–49.
[15] E. J. Cramer, J. E. Dennis, Jr., P. D. Frank, R. M. Lewis, and G. R. Shubin, *Problem formulation for multidisciplinary optimization*, SIAM J. Optim., 4 (1994), pp. 754–776.
[16] A. DeMiguel and W. Murray, *An analysis of collaborative optimization methods*, in Proceedings of 8th AIAA/USAF/NASA/ISSMO Symposium on Multidisciplinary Analysis and Optimization, Long Beach, CA, 2000.
[17] A. Doucet, S. Godsill, and C. Andrieu, *On sequential Monte Carlo sampling methods for Bayesian filtering*, Statist. Comput., 10 (2000), pp. 197–208.
[18] J. Eftang and A. Patera, *Port reduction in parametrized component static condensation: Approximation and a posteriori error estimation*, Internat. J. Numer. Methods Engrg., 96 (2013), pp. 269–302.
[19] H. Elman, C. Miller, E. Phipps, and R. Tuminaro, *Assessment of collocation and Galerkin approaches to linear diffusion equations with random data*, Int. J. Uncertain. Quantif., 1 (2011), pp. 19–34.
[20] H. Elman, D. Silvester, and A. Wathen, *Finite Elements and Fast Iterative Solvers*, Oxford University Press, New York, 2005.
[21] C. Farhat, *A Lagrange multiplier based divide and conquer finite element algorithm*, Comput. Systems Engrg., 2 (1991), pp. 149–156.
[22] C. Farhat and F.-X. Roux, *A method of finite element tearing and interconnecting and its parallel solution algorithm*, Internat. J. Numer. Methods Engrg., 32 (1991), pp. 1205–1227.
[23] L. W. Gelhar, A. L. Gutjahr, and R. L. Naff, *Stochastic analysis of macrodispersion in a stratified aquifer*, Water Resour. Res., 15 (1979), pp. 1387–1397.
[24] R. Ghanem and P. Spanos, *Stochastic Finite Elements: A Spectral Approach*, Dover Publications, New York, 2003.
[25] M. Gunzburger, J. Peterson, and J. Shadid, *Reduced-order modeling of time-dependent PDEs with multiple parameters in the boundary data*, Comput. Methods Appl. Mech. Engrg., 196 (2007), pp. 1030–1047.
[26] M. Hadigol, A. Doostan, H. G. Matthies, and R. Niekamp, *Partitioned treatment of uncertainty in coupled domain problems: A separated representation approach*, Comput. Methods Appl. Mech. Engrg., 274 (2014), pp. 103–124.
[27] B. E. Hansen, *Uniform convergence rates for kernel estimation with dependent data*, Economet. Theory, 24 (2008), pp. 726–748.
[28] M. Heinkenschloss, *Time-domain decomposition iterative methods for the solution of distributed linear quadratic optimal control problems*, J. Comput. Appl. Math., 173 (2005), pp. 169–198.
[29] M. Heinkenschloss and M. Herty, *A spatial domain decomposition method for parabolic optimal control problems*, J. Comput. Appl. Math., 201 (2007), pp. 88–111.
[30] F. Hitchcock, *The expression of a tensor or a polyadic as a sum of products*, J. Math. Phys., 6 (1927), pp. 164–189.
[31] P. Holmes, J. L. Lumley, and G. Berkooz, *Turbulence, Coherent Structures, Dynamical Systems and Symmetry*, Cambridge, New York, 1996.
[32] A. Ihler, *Kernel Density Estimation Toolbox for MATLAB*, http://www.ics.uci.edu/ihler/code/kde.html.
[33] A. Kong, J. Liu, and W. Wong, *Sequential imputations and Bayesian missing data problems*, J. Amer. Statist. Assoc., 89 (1994), pp. 278–288.

[34] I. Kroo, *Distributed Multidisciplinary Design and Collaborative Optimization*, in VKI Lecture Series on Optimization Methods & Tools for Multicriteria/Multidisciplinary Design, 2004.
[35] I. Kroo and V. Manning, *Collaborative optimization: Status and directions*, in Proceedings of 8th AIAA/USAF/NASA/ISSMO Symposium on Multidisciplinary Analysis and Optimization, Long Beach, CA, 2000.
[36] C. G. Lambert, S. E. Harrington, C. R. Harvey, and A. Glodjo, *Efficient on-line nonparametric kernel density estimation*, Algorithmica, 25 (1999), pp. 37–57.
[37] C. D. Levermore, G. C. Pomraning, D. L. Sanzo, and J. Wong, *Linear transport theory in a random medium*, J. Math. Phys., 27 (1986), pp. 2526–2536.
[38] S. Li and D. McLaughlin, *A nonstationary spectral method for solving stochastic groundwater problems: Unconditional analysis*, Water Resour. Res., 27 (1991), pp. 1589–1605.
[39] G. Lin, A. M. Tartakovsky, and D. M. Tartakovsky, *Uncertainty quantification via random domain decomposition and probabilistic collocation on sparse grids*, J. Comput. Phys., 229 (2010), pp. 6995–7012.
[40] J. Liu, *Metropolized independent sampling with comparisons to rejection sampling and importance sampling*, Statist. Comput., 6 (1996), pp. 113–119.
[41] S. Lophaven, H. Nielsen, and J. Søndergaard, *A MATLAB Kriging Toolbox*, Technical report IMM-TR-2002-12, Technical University of Denmark, 2002.
[42] G. Matheron and G. De Marsily, *Is transport in porous media always diffusive? A counterexample*, Water Resour. Res., 16 (1980), pp. 901–917.
[43] N. Michelena, H. Park, and P. Y. Papalambros, *Convergence properties of analytical target cascading*, AIAA J., 42 (2003), pp. 897–905.
[44] H. Min Kim, N. F. Michelena, P. Y. Papalambros, and T. Jiang, *Target cascading in optimal system design*, J. Mechanical Design, 125 (2003), pp. 474–480.
[45] A. Quarteroni and A. Valli, *Domain Decomposition Methods for Partial Differential Equations*, Oxford University Press, New York, 1999.
[46] J. Sacks, W. Welch, T. Mitchell, and H. Wynn, *Design and analysis of computer experiments*, Statist. Sci., 4 (1989), pp. 409–423.
[47] A. Sarkar, N. Benabbou, and R. Ghanem, *Domain decomposition of stochastic PDEs: Theoretical formulations*, Internat. J. Numer. Methods Engrg., 77 (2009), pp. 689–701.
[48] C. Schwab and R. A. Todor, *Karhunen-Loève approximation of random fields by generalized fast multipole methods*, J. Comput. Phys., 217 (2006), pp. 100–122.
[49] S. Shan and G. Wang, *Survey of modeling and optimization strategies to solve high-dimensional design problems with computationally-expensive black-box functions*, Struct. Multidiscip. Optim., 41 (2010), pp. 219–241.
[50] L. Sirovich, *Turbulence and the dynamics of coherent structures, Part* I*: Coherent structures*, Quart. Appl. Math., 45 (1987), pp. 561–571.
[51] I. Skare, E. Blviken, and L. Holden, *Improved sampling-importance resampling and reduced bias importance sampling*, Scand. J. Statist., 30 (2007), pp. 719–737.
[52] A. F. M. Smith and A. E. Gelfand, *Bayesian statistics without tears: A sampling-resampling perspective*, Ameri. Statist., 46 (1992), pp. 84–88.
[53] I. P. Sobieski and I. M. Kroo, *Collaborative optimization using response surfaces*, AIAA J., 38 (2000), pp. 1931–1938.
[54] J. Sobieszczanski-Sobieski, J. Agte, and R. Sandusky, *Bilevel integrated system synthesis*, AIAA J., 38 (2000), pp. 164–172.
[55] J. Sobieszczanski-Sobieski, T. Altus, M. Phillips, and R. Sandusky, *Bilevel integrated system synthesis for concurrent and distributed processing*, AIAA J., 41 (2003), pp. 1996–2003.
[56] D. M. Tartakovsky, *Assessment and management of risk in subsurface hydrology: A review and perspective*, Adv. Water Resour., 51 (2013), pp. 247–260.
[57] N. P. Tedford and J. R. R. A. Martins, *Benchmarking multidisciplinary design optimization algorithms*, Optim. Eng., 11 (2010), pp. 159–183.
[58] D. Wied and R. Weißbach, *Consistency of the kernel density estimator: A survey*, Statist. Papers, 53 (2012), pp. 1–21.
[59] C. L. Winter and D. M. Tartakovsky, *Mean flow in composite porous media*, Geophys. Res. Lett., 27 (2000), pp. 1759–1762.
[60] C. L. Winter and D. M. Tartakovsky, *Groundwater flow in heterogeneous composite aquifers*, Water Resour. Res., 38 (2002), pp. 23–1–23–11.
[61] C. L. Winter, D. M. Tartakovsky, and A. Guadagnini, *Moment differential equations for flow in highly heterogeneous porous media*, Surv. Geophys., 24 (2003), pp. 81–106.
[62] C. Yang, R. Duraiswami, N. A. Gumerov, and L. Davis, *Improved fast Gauss transform and efficient kernel density estimation*, in Proceedings of ICCV, 2003, pp. 464–471.