Parallel Domain Decomposition Strategies for Stochastic Elliptic Equations
Part B: Accelerated Monte-Carlo Sampling with Local PC Expansions

Andres A. Contreras[*] Paul Mycek[†] Olivier P. Le Maître[‡]
Francesco Rizzi[§] Bert Debusschere [¶] Omar M. Knio[‖][**]

June 12, 2017

Abstract

Solving Stochastic Differential Equations (SPDE) can be a computationally intensive task, particularly when the underlying parametrization of the stochastic input field involves a large number of random variables. Direct Monte Carlo (MC) sampling methods are well suited for this type of situation, since their cost is independent of the input complexity. Unfortunately, MC sampling methods suffer from slow convergence. In this manuscript, we propose an acceleration framework for elliptic SPDEs that relies on Domain Decomposition techniques and Polynomial Chaos (PC) expansions of local operators to reduce the cost of solving a SPDE via MC sampling. The approach exploits the fact that, at the subdomain level, the number of variables required to accurately parametrize the input stochastic field can be significantly reduced, as covered in detail in the prequel ("Part A") to this paper. This makes it feasible to construct PC expansions of the local contributions to the condensed problem (*i.e.* the Schur Complement of the discretized operator). The approach basically consists of two main stages: 1) a preprocessing stage in which PC expansions of the condensed problem are computed and 2) a Monte Carlo sampling stage where random samples of the solution are computed. The proposed method its naturally parallelizable. Extensive numerical tests are used to validate the methodology and assess its serial and parallel performance.

Keywords: Stochastic Elliptic Equations, Domain Decomposition, Polynomial Chaos expansion, Monte Carlo method

1 Introduction

Stochastic Partial Differential Equations (SPDEs) are of great importance in a wide range of applications. Computational approaches for the solution of SPDEs conceptually involve three essential steps: the modeling of the input uncertainty, the solution of the governing equations, and ultimately the post-processing the output to characterize the uncertainty. This paper ("Part B") and its prequel ("Part A") focus on the first two steps. In Part A [10], we discussed a domain decomposition strategy to approximate random fields (input uncertainty) using local reduced bases and local coordinates. Now (in Part B), the structure of local representations is exploited to accelerate the Monte Carlo sampling of the solution.

[*]Duke University, 121 Hudson Hall, Box 90287, Durham, NC 27708 (`andres.contreras@duke.edu`).
[†]Duke University, 144 Hudson Hall, Box 90300, Durham, NC 27708 (`paul.mycek@duke.edu`).
[‡]CNRS, LIMSI, Université Paris Saclay, Orsay, France (`olm@limsi.fr`).
[§]Sandia National Laboratories, Livermore, CA. (`fnrizzi@sandia.gov`).
[¶]Sandia National Laboratories, Livermore, CA. (`bjdebus@sandia.gov`).
[‖]Duke University, 144 Hudson Hall, Box 90300, Durham, NC 27708 (`omar.knio@duke.edu`).
[**]King Abdullah University of Science and Technology, Thuwal, Saudi Arabia.

Two common approaches to solving SPDE are the Stochastic Spectral method [15, 21] and Monte Carlo (MC) sampling methods [5, 30, 2, 8]. One particular class of Stochastic Spectral methods uses polynomial chaos (PC) expansions. PC expansions have been studied extensively [15, 21, 37] and perform very well in a number of applications, including elliptic and parabolic problems with random coefficients [22, 11, 1, 12, 14] and fluid flow models [23, 19, 25]. Unfortunately, the use of PC expansions is subject to the curse of dimensionality, and can quickly become computationally intractable when the stochastic dimension is large (the complexity can grow exponentially as a function of the stochastic dimension). Different methods have been devised to address the complexity issue, e.g. using low-rank approximations [27, 28, 29, 35] and adaptive strategies [4, 3, 7], but when the stochastic dimension is very large some of the challenges can still remain [9]. On the other hand, the cost of direct MC sampling methods is independent of the stochastic dimension, which is a desirable feature when dealing with high dimensional problems. However, it is known that MC methods have a slow convergence rate, with the root mean squared error inversely proportional to the square root of the number of samples. So, if the computational cost of obtaining an individual sample is high, these methods can be quite costly.

The stochastic dimension of the problem is closely related to the stochastic discretization method used to solve the problem. One common approach to discretize the stochastic space is the Karhunen-Loève (KL) expansion [24, 17, 16]. The number of terms in the KL expansion is what sets the stochastic dimension, and it turns out that for a given accuracy level, the number of terms in the expansion is actually proportional to the size of the physical domain. Thus, the smaller the domain, the fewer terms that are necessary in the expansion to achieve a desired level of accuracy. In [6], Chen et al. rely on Domain Decomposition technique to exploit this fact. By partitioning the global domain into smaller subdomains, a set of local problems is obtained, each with a significantly reduced stochastic dimension. In [31] Gosh and Pranesh present a closely related approach based on the Spectral Stochastic Finite Element Method.

In both of the approaches mentioned above, PC approximations of the local problem solution at the subdomain level are constructed. The cost of obtaining these local PC approximations is reduced by using the local random variables over each subdomain (i.e. the lower stochastic dimension at the subdomain level reduces the cost). An important point, that is acknowledged in these two papers, is that the local random variables in one subdomain have a dependence structure on the local random variables in other subdomains. Nonetheless, in these two works the local random variables are treated as independent across subdomains (corrections are made with the introduction of additional global random variables). In contrast, in our proposed approach developed in the two-parts manuscript, we consider the actual dependence structure of the local variables and use them in the construction of local boundary-to-boundary maps that help us accelerate the solution of Stochastic Elliptic PDEs via MC sampling. Part A [10] analyzed in detail the local KL expansions approach and the dependences of the local random variables. The present Part B concerns the solution of the Stochastic Elliptic PDEs by means of a MC sampling method that is accelerated by constructing PC expansions of the boundary-to-boundary maps (and not of the local problem solutions).

More precisely, our proposed approach is divided into two main stages: 1) a preprocessing stage in which PC expansions of a condensed problem are computed, and 2) a Monte Carlo sampling stage where samples of the solution are computed. First, the physical domain is discretized using the finite element method; then the global domain Ω is divided into D non-overlapping subdomains. This results in a condensed problem for the nodal values at the subdomains' interfaces. Given this discretization, the preprocessing stage starts by breaking the condensed problem into individual contributions from each subdomain, and computing local KL expansions over each subdomain (as described in [10]). Then, using the local KL expansion, and taking advantage of the reduced stochastic dimension of the local problems, PC expansions of the local contributions to the condensed problem are constructed. The second stage then consists of generating samples (this requires taking into consideration the dependence structure of the local random variables), evaluating the PC expansion of the reduced problem for said samples, and solving the reduced problem to obtain samples of the solution.

In summary, our approach is a parallel solver that takes advantage of the PC method at the local level to reduce the cost of using the MC sampling method at a global level. There are two main contributions of the current work; first we take into account the dependence structure of the local random variables across subdomains. Furthermore, these local random variables are jointly sampled with the convenient approach described in [10], which allows us to accurately characterize the random process on which the SPDE depends. (In general this is not possible when the local random variables are assumed independent across subdomains.) The second contribution is that we use the local expansions to construct local PC expansions of the condensed problem (as opposed to constructing local PC expansion of the solutions at the subdomain level), and from these local expansions we build a PC expansion of the global condensed problem, which significantly reduces the sampling cost in the MC sampling method. We remark that by building the global condensed system in this manner we preserve the proper dependence structure in the overall solution sought.

The outline of this paper is as follows. In section 2, we first recall how the domain decomposition method is applied, both to a deterministic and to a stochastic PDE, and also discuss the Monte Carlo sampling method. In section 3, we discuss the limitations of constructing a PC expansion of the solution and describe how instead we proceed with the construction of the PC expansion of the Condensed Problem. We also address the sampling of the condensed problem and outline some of the implementation details. Next, in section 4 the method is validated with some numerical results. In section 5, we analyze for the test case the performance of the method terms of complexity and parallel efficiency. Finally, in section 6, some concluding remarks are provided.

2 Elliptic Problem

2.1 Deterministic case

We consider the following elliptic problem in a bounded domain $\Omega \subset \mathbb{R}^m$, with boundary $\partial\Omega$:

$$\begin{cases} \boldsymbol{\nabla} \cdot (\kappa(\boldsymbol{x})\boldsymbol{\nabla} u) = -f(\boldsymbol{x}), & \boldsymbol{x} \in \Omega \\ \mathcal{B}(\boldsymbol{x}, u) = 0, & \boldsymbol{x} \in \partial\Omega, \end{cases} \tag{1}$$

where $\mathcal{B}$ is the (linear) boundary condition operator and $0 < \kappa_{\min} < \kappa(\boldsymbol{x}) < \kappa_{\max} < +\infty$ is the diffusion coefficient. For simplicity, we shall restrict ourselves to the case of homogeneous Dirichlet and Neumann boundary conditions, that is

$$u(\boldsymbol{x} \in \partial\Omega_D) = 0, \quad \partial_n u(\boldsymbol{x} \in \partial\Omega_N) = 0, \tag{2}$$

where ∂_n is the derivative in the normal direction, and Ω_D and Ω_N are the Dirichlet and Neumann parts of the boundary, such that $\partial\Omega_N \cup \partial\Omega_D = \partial\Omega$, $\partial\Omega_N \cap \partial\Omega_D = \emptyset$.

To solve (1) we consider standard finite element (FE) methods based on a conforming triangulation of Ω into a set, $\mathcal{T}$, of $\mathrm{N_e}$ non-overlapping elements, Σ_{e}. The FE approximation is based on a nodal basis representation. Let n be a node of the mesh, with position $\boldsymbol{x}_{\mathrm{n}}$, we denote by $\mathcal{N}$ the set of nodes that do not belong to the Dirichlet boundary $\partial\Omega_D$, and $\mathrm{N_n} \doteq |\mathcal{N}|$ the number of nodes in $\mathcal{N}$. The approximation of u is sought as

$$u(\boldsymbol{x}) \approx \sum_{\mathrm{n}\in\mathcal{N}} \Phi_{\mathrm{n}}(\boldsymbol{x}) u_{\mathrm{n}}, \tag{3}$$

where the functions Φ_{n} are nodal basis functions satisfying:

$$\forall \mathrm{n}, \mathrm{n}' \in \mathcal{N}, \Phi_{\mathrm{n}}(\boldsymbol{x}_{\mathrm{n}'}) = \begin{cases} 1, & \mathrm{n} = \mathrm{n}' \\ 0, & \mathrm{n}' \neq \mathrm{n}, \end{cases} \tag{4}$$

and $\Phi_{\mathrm{n}}(\boldsymbol{x} \in \partial\Omega_D) = 0$. It is further assumed that the support of Φ_{n} is limited to the elements that have n as one of their nodes. The weak form of problem (1) is:
Find $u \in V^{\mathrm{FE}}$ such that

$$\int_\Omega \kappa(\boldsymbol{x})\boldsymbol{\nabla} u(\boldsymbol{x}) \cdot \boldsymbol{\nabla} v(\boldsymbol{x}) d\boldsymbol{x} = \int_\Omega f(\boldsymbol{x}) v(\boldsymbol{x}) d\boldsymbol{x} \quad \forall v \in V^{\mathrm{FE}}, \tag{5}$$

where V^{FE} is the linear span of nodal functions $\{\Phi_{\mathrm{n}}, \mathrm{n} \in \mathcal{N}\}$. The variational problem can be recast as a linear system of equations for the vector, $\mathbf{u}$, of unknown nodal values,

$$[\mathbf{A}]\mathbf{u} = \mathbf{b}, \tag{6}$$

where $\mathbf{u}$ and $\mathbf{b} \in \mathbb{R}^{\mathrm{N_n}}$. The system matrix $[\mathbf{A}] \in \mathbb{R}^{\mathrm{N_n}\times\mathrm{N_n}}$ is symmetric positive definite, with entries

$$[\mathbf{A}]_{\mathrm{nn'}} = \int_\Omega \kappa(\boldsymbol{x})\boldsymbol{\nabla}\Phi_{\mathrm{n}}(\boldsymbol{x}) \cdot \boldsymbol{\nabla}\Phi_{\mathrm{n'}}(\boldsymbol{x})d\boldsymbol{x}. \tag{7}$$

The components of the system right-hand-side are given by

$$\mathbf{b}_{\mathrm{n}} = \int_\Omega f(\boldsymbol{x})\Phi_{\mathrm{n}}(\boldsymbol{x})d\boldsymbol{x}. \tag{8}$$

2.1.1 Domain Decomposition method

Owing to the compact support of the nodal basis functions, the matrix $[\mathbf{A}]$ is sparse and efficient iterative methods (*e.g.* Preconditioned Conjugate Gradient) can be employed to solve (6). However the system size $\mathrm{N_n}$ may be large, inducing a significant resolution cost and motivating the introduction of domain decomposition methods [20, 32, 34, 36].

Domain partitioning. To this end, we first partition Ω into a set of D non-overlapping subdomains $\Omega^{(d)}$ consisting of subsets $\mathcal{T}^{(d)}$ of neighboring elements; we have

$$\overline{\Omega^{(d)}} \doteq \bigcup_{\mathrm{e}\in\mathcal{T}^{(d)}} \Sigma_{\mathrm{e}}, \quad \bigcup_{d=1}^{\mathrm{D}} \mathcal{T}^{(d)} = \mathcal{T}, \quad \mathcal{T}^{(d)} \cap \mathcal{T}^{(d'\neq d)} = \varnothing. \tag{9}$$

where $\overline{\Omega^{(d)}}$ is the closure of $\Omega^{(d)}$. We denote $\mathrm{N}_{\mathrm{e}}^{(d)} = |\mathcal{T}^{(d)}|$ the number of elements in $\Omega^{(d)}$ and $\mathcal{N}^{(d)}$ the subset of nodes in $\mathcal{N}$ belonging to $\Omega^{(d)}$:

$$\mathcal{N}^{(d)} = \left\{\mathrm{n} \in \mathcal{N}; \boldsymbol{x}_{\mathrm{n}} \in \overline{\Omega^{(d)}}\right\}. \tag{10}$$

The sets $\mathcal{N}^{(d)}$ can be further split into disjoint subsets of interior nodes belonging to $\Omega^{(d)}$ only, and boundary nodes lying at the interface of more than one subdomain:

$$\mathcal{N}_{\mathrm{in}}^{(d)} = \left\{\mathrm{n} \in \mathcal{N}^{(d)}; \mathrm{n} \notin \mathcal{N}^{(d'\neq d)}\right\}, \quad \mathcal{N}_\Gamma^{(d)} = \mathcal{N}^{(d)} \backslash \mathcal{N}_{\mathrm{in}}^{(d)}. \tag{11}$$

Clearly, the sets $\mathcal{N}_{\mathrm{in}}^{(d)}$ are disjoint, while $\mathcal{N}_\Gamma^{(d)} \cap \mathcal{N}_\Gamma^{(d')}$ is not empty for two neighboring subdomains such that $\partial\Omega^{(d)} \cap \partial\Omega^{(d')} \neq \varnothing$. We then define the full set of inner and boundary nodes of the partitioned domain through

$$\mathcal{N}_{\mathrm{in}} = \bigcup_{d=1}^{\mathrm{D}} \mathcal{N}_{\mathrm{in}}^{(d)}, \quad \mathcal{N}_\Gamma = \bigcup_{d=1}^{\mathrm{D}} \mathcal{N}_\Gamma^{(d)}, \tag{12}$$

and set $\mathrm{N_{in}} = |\mathcal{N}_{\mathrm{in}}|$, $\mathrm{N}_\Gamma = |\mathcal{N}_\Gamma|$. We can now rewrite the FE approximation of u in (3) as

$$u(\boldsymbol{x}) \approx \sum_{\mathrm{n}\in\mathcal{N}_{\mathrm{in}}} u_{\mathrm{n}}\Phi_{\mathrm{n}}(\boldsymbol{x}) + \sum_{\mathrm{n}\in\mathcal{N}_\Gamma} u_{\mathrm{n}}\Phi_{\mathrm{n}}(\boldsymbol{x}). \tag{13}$$

Iterative Domain Decomposition solver. Upon reordering of the nodes, the linear system in (6) can be recast in the following block matrix form,

$$\begin{bmatrix} [\mathbf{A}_{\boldsymbol{\Gamma},\boldsymbol{\Gamma}}] & [\mathbf{A}_{\boldsymbol{\Gamma},\mathrm{in}}] \\ [\mathbf{A}_{\mathrm{in},\boldsymbol{\Gamma}}] & [\mathbf{A}_{\mathrm{in},\mathrm{in}}] \end{bmatrix} \begin{pmatrix} \mathbf{u}_\Gamma \\ \mathbf{u}_{\mathrm{in}} \end{pmatrix} = \begin{pmatrix} \mathbf{b}_\Gamma \\ \mathbf{b}_{\mathrm{in}} \end{pmatrix}, \tag{14}$$

with the previous expressions for the matrix and right-hand-sides entries. This system can be further manipulated to eliminate the internal unknowns in $\mathbf{u}_{\mathrm{in}}$ to come up with the condensed problem for the nodal values at the subdomains' interfaces,

$$\widehat{[\mathbf{A}]}\mathbf{u}_\Gamma = \widehat{\mathbf{b}}, \tag{15}$$

where

$$\widehat{[\mathbf{A}]} \doteq [\mathbf{A}_{\boldsymbol{\Gamma},\boldsymbol{\Gamma}}] - [\mathbf{A}_{\boldsymbol{\Gamma},\mathrm{in}}][\mathbf{A}_{\mathrm{in},\mathrm{in}}]^{-1}[\mathbf{A}_{\mathrm{in},\boldsymbol{\Gamma}}], \quad \widehat{\mathbf{b}} \doteq \mathbf{b}_{\Gamma} - [\mathbf{A}_{\boldsymbol{\Gamma},\mathrm{in}}][\mathbf{A}_{\mathrm{in},\mathrm{in}}]^{-1}\mathbf{b}_{\mathrm{in}}. \tag{16}$$

Considering an iterative method to solve (15), the main computationally heavy task amounts to performing matrix-vector products between $\widehat{[\mathbf{A}]}$ and successive iterates vectors of $\mathbb{R}^{\mathrm{N}_\Gamma}$. A closer inspection reveals that multiplying a vector by $\widehat{[\mathbf{A}]}$ involves solving for $\mathbf{v}$ by inverting a system of the form $[\mathbf{A}_{\mathrm{in},\mathrm{in}}]\mathbf{v} = \mathbf{w}$. This step is actually the heaviest one in the iterative solution, as it requires the solution of a linear system whose dimension, $\mathrm{N_n} - \mathrm{N}_\Gamma$, is generally close to the dimension of the non-condensed problem, that is $\mathrm{N_n}$. However, it is crucial to remark that $[\mathbf{A}_{\mathrm{in},\mathrm{in}}]$ has diagonal block structure when the nodes in $\mathcal{N}_{\mathrm{in}}$ are ordered by subdomains; in this case, we have

$$[\mathbf{A}_{\mathrm{in},\mathrm{in}}]\mathbf{v} = \begin{bmatrix} \left[\mathbf{A}^{(1)}_{\mathrm{in},\mathrm{in}}\right] & [\mathbf{0}] & \cdots & [\mathbf{0}] \\ [\mathbf{0}] & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & [\mathbf{0}] \\ [\mathbf{0}] & \cdots & [\mathbf{0}] & \left[\mathbf{A}^{(\mathrm{D})}_{\mathrm{in},\mathrm{in}}\right] \end{bmatrix} \begin{pmatrix} \mathbf{v}^{(1)} \\ \vdots \\ \mathbf{v}^{(\mathrm{D})} \end{pmatrix} = \begin{pmatrix} \mathbf{w}^{(1)} \\ \vdots \\ \mathbf{w}^{(\mathrm{D})} \end{pmatrix} \quad \Rightarrow \mathbf{v}^{(d)} = \left[\mathbf{A}^{(d)}_{\mathrm{in},\mathrm{in}}\right]^{-1}\mathbf{w}^{(d)}. \tag{17}$$

It shows that computing $\mathbf{v} = [\mathbf{A}_{\mathrm{in},\mathrm{in}}]^{-1}\mathbf{w}$, given $\mathbf{w} \in \mathbb{R}^{\mathrm{N_{in}}}$, amounts to solving D subsystems or local problems *independently* over each subdomain. Not only does this call for the inversion of systems with much smaller sizes, typically $\mathrm{N}^{(d)}_{\mathrm{in}} \approx \mathrm{N_n}/\mathrm{D}$, but these computations can be carried out in parallel for different subdomains. The same remark also applies to the determination of the right-hand-side $\widehat{\mathbf{b}}$ of the reduced problem (15). The possibility of applying efficiently the condensed operator $\widehat{[\mathbf{A}]}$ on a vector $\mathbf{u} \in \mathbb{R}^{\mathrm{N}_\Gamma}$, through local solves over subdomains, motivates the use of matrix-free type iterative methods where $\widehat{[\mathbf{A}]}$ is never formally assembled. In such an approach, one eventually only computes the sparse matrices $[\mathbf{A}_{\boldsymbol{\Gamma},\boldsymbol{\Gamma}}]$ and $[\mathbf{A}_{\mathrm{in},\boldsymbol{\Gamma}}]$ and the local problem matrices $\left[\mathbf{A}^{(d)}_{\mathrm{in},\mathrm{in}}\right]$. The latter, owing to their low dimension, can even be factorized to speed-up subsequent products with $\widehat{[\mathbf{A}]}$.

Finally, when the reduced problem solution $\mathbf{u}_\Gamma$ is obtained, one can compute the solution over selected subdomains solving local problems with corresponding Dirichlet boundary conditions in $\mathbf{u}_\Gamma$ (see below).

2.1.2 Subdomains expansion of the condensed operator

The discussion above highlighted the role of the local problems in the structure of the condensed problem. In fact, the system in (15) can be formally recast to highlight independent contributions from the subdomains, namely according to:

$$\widehat{[\mathbf{A}]}\mathbf{u}_\Gamma = \sum_{d=1}^{\mathrm{D}} \widehat{[\mathbf{A}]}^{(d)}\mathbf{u}^{(d)}_\Gamma, \quad \widehat{\mathbf{b}} = \mathbf{b}_\Gamma + \sum_{d=1}^{\mathrm{D}} \widehat{\mathbf{b}}^{(d)}. \tag{18}$$

Focusing first on the right-hand-side expansion, we identify

$$\widehat{\mathbf{b}}^{(d)} = -\left[\mathbf{A}^{(d)}_{\boldsymbol{\Gamma},\mathrm{in}}\right]\left[\mathbf{A}^{(d)}_{\mathrm{in},\mathrm{in}}\right]^{-1}\mathbf{b}^{(d)}_{\mathrm{in}}. \tag{19}$$

Note that for simplicity, the expressions above are formal and involved an abuse of notations. In particular, we allow varying the size of matrices and vectors, by removing unnecessary entries or nodal components, or by padding with zeros when necessary. For instance, depending on the context the matrix $\left[\mathbf{A}^{(d)}_{\mathrm{in},\mathrm{in}}\right]^{-1}$ can be understood as a $\mathrm{N}^{(d)}_{\mathrm{in}} \times \mathrm{N}^{(d)}_{\mathrm{in}}$ matrix or as its padded to zero to form the $\mathrm{N_{in}} \times \mathrm{N_{in}}$ version with only a nonzero (d)-th diagonal block. With this abuse of notation, the inverse of $[\mathbf{A}_{\mathrm{in},\mathrm{in}}]$ in (17) can be formally written as

$$[\mathbf{A}_{\mathrm{in},\mathrm{in}}]^{-1} = \sum_{d=1}^{\mathrm{D}} \left[\mathbf{A}^{(d)}_{\mathrm{in},\mathrm{in}}\right]^{-1}.$$

Similarly, the expansion of $\widehat{[\mathbf{A}]}$ in (18) means that the expansion term $\widehat{[\mathbf{A}]}^{(d)}\mathbf{u}_\Gamma^{(d)}$ accounts for the effect of the (d)-th subdomain only. To derive an expression for these matrices, we fix a subdomain d, select $\mathrm{n} \in \mathcal{N}_\Gamma^{(d)}$ and consider the solution of

$$\left[\mathbf{A}_{\mathrm{in,in}}^{(d)}\right]\mathbf{u}_{\mathrm{in,n}}^{(d)} = -\left[\mathbf{A}_{\mathrm{in},\boldsymbol{\Gamma}}^{(d)}\right]\mathbf{e}_{\mathrm{n}}^{(d)}, \tag{20}$$

where $\mathbf{e}_{\mathrm{n}}^{(d)}$ is the canonical vector with all zero component except the n-th one equal to 1. The solution $\mathbf{u}_{\mathrm{in,n}}^{(d)}$ are the (internal) nodal values of the finite element approximation of the elliptic problem over $\Omega^{(d)}$ for homogeneous boundary conditions all over $\partial\Omega^{(d)}$, except at node $\mathrm{n} \in \mathcal{N}_\Gamma^{(d)}$ where the nodal value is set to one. From this family of elementary solutions we define the vector

$$\boldsymbol{I}_{\mathrm{n}}^{(d)} \doteq \left[\mathbf{A}_{\boldsymbol{\Gamma},\mathrm{in}}^{(d)}\right]\mathbf{u}_{\mathrm{in,n}}^{(d)} + \left[\mathbf{A}_{\boldsymbol{\Gamma},\boldsymbol{\Gamma}}^{(d)}\right]\mathbf{e}_{n}^{(d)}, \tag{21}$$

where

$$\left[\mathbf{A}_{\boldsymbol{\Gamma},\boldsymbol{\Gamma}}^{(d)}\right]_{\mathrm{n,n'}} \doteq \begin{cases} \displaystyle\int_{\Omega^{(d)}} \kappa \boldsymbol{\nabla}\Phi_{\mathrm{n}} \cdot \boldsymbol{\nabla}\Phi_{\mathrm{n'}} d\boldsymbol{x}, & \mathrm{n,n'} \in \mathcal{N}_\Gamma^{(d)} \\ 0, & \text{otherwise.} \end{cases} \tag{22}$$

We observe that the computation of the vector $\boldsymbol{I}_{\mathrm{n}}^{(d)}$ involves only quantities and operators localized on the considered subdomain. In particular, we note that the definition of the matrix in (22) involves an integral restricted to $\Omega^{(d)}$, such that $[\mathbf{A}_{\boldsymbol{\Gamma},\boldsymbol{\Gamma}}] = \sum_d \left[\mathbf{A}_{\boldsymbol{\Gamma},\boldsymbol{\Gamma}}^{(d)}\right]$. Finally, exploiting the linearity of elliptic equation and superposition principle, we obtain:

$$\widehat{[\mathbf{A}]}\mathbf{u}_\Gamma = \sum_{d=1}^{\mathrm{D}} \widehat{[\mathbf{A}]}^{(d)}\mathbf{u}_\Gamma^{(d)}, \quad \widehat{[\mathbf{A}]}^{(d)}\mathbf{u}_\Gamma^{(d)} = \sum_{\mathrm{n}\in\mathcal{N}_\Gamma^{(d)}} \boldsymbol{I}_{\mathrm{n}}^{(d)} \left(\mathbf{u}_\Gamma^{(d)}\right)_{\mathrm{n}}, \tag{23}$$

showing that the columns of the matrices $\widehat{[\mathbf{A}]}^{(d)}$ are made of the vectors $\boldsymbol{I}^{(d)}_{\mathrm{n}\in\mathcal{N}_\Gamma^{(d)}}$.

Constructing the condensed operator expansion in (18) involves the solution over each subdomain of a set of local elliptic problems (20), in fact the same elliptic problem with $\mathcal{N}_\Gamma^{(d)}$ right-hand-sides. Although it can be performed efficiently in parallel, the explicit construction of the condensed operator is generally not considered in the practical implementation of domain decomposition approaches for elliptic problems, because of its computational complexity which is generally larger than that of the direct matrix-free iterative method described in the previous section. However, the case of stochastic elliptic problems is different as many stochastic samples may have to be computed, so that having an explicit representation of the (now stochastic) condensed operator may be interesting. We expand on this idea in the following sections.

2.2 Stochastic elliptic problem

2.2.1 Formulation of the stochastic problem

We now extend the deterministic problem in (1) to the stochastic case. The case of stochastic forcing f induces no particular difficulty and can be treated in the framework to be introduced below. For simplicity with restrict the presentation to the case of a random diffusion field κ. Let (Θ, Σ, μ) be a probability triplet; the problem now becomes

$$\boldsymbol{\nabla} \cdot (\kappa(\boldsymbol{x},\theta)\boldsymbol{\nabla}u(\boldsymbol{x},\theta)) = -f(\boldsymbol{x}), \quad \boldsymbol{x} \in \Omega, \theta \in \Theta, \tag{24}$$

with additional (almost sure) homogenous Neumann and Dirichlet boundary conditions for $\boldsymbol{x} \in \partial\Omega$. For the well posedness of the problem, we assume that the random field $\kappa(\boldsymbol{x},\theta)$ is almost surely bounded below and above for almost every $\boldsymbol{x}$. Then the stochastic solution u has finite second order moments,

$$\mathbb{E}\left[u(\boldsymbol{x},\cdot)^2\right] = \int_\Theta u(\boldsymbol{x},\theta)^2 d\mu(\theta) < +\infty, \tag{25}$$

where $\mathbb{E}[\cdot]$ denotes the expectation operator.

As in the deterministic case, we proceed with the spatial discretization over a deterministic finite element space V^{FE}, expressing the discrete solution from its random nodal values over the mesh,

$$u(\boldsymbol{x},\theta) = \sum_{\mathrm{n}\in\mathcal{N}} u_{\mathrm{n}}(\theta)\Phi_{\mathrm{n}}(\boldsymbol{x}) \in \mathbf{V}^{\mathrm{FE}}. \tag{26}$$

Above, we denoted the solution space $\mathbf{V}^{\mathrm{FE}}$ which results from the tensorization of the spatial FE space with the space of second order random variables: $\mathbf{V}^{\mathrm{FE}} = V^{\mathrm{FE}} \otimes L_2(\Theta,\mu)$. The (semi) weak form is obtained multiplying (24) by $v \in V^{\mathrm{FE}}$, and integrating (by parts) first over Ω; this results in

$$\int_\Omega \kappa(\boldsymbol{x},\theta)\boldsymbol{\nabla}u(\boldsymbol{x},\theta)\cdot\boldsymbol{\nabla}v(\boldsymbol{x})d\boldsymbol{x} = \int_\Omega f(\boldsymbol{x})v(\boldsymbol{x})d\boldsymbol{x}, \quad \forall v \in V^{\mathrm{FE}}.$$

Note that the equality stands in the almost sure sense. Given the approximation form in (26), the variational formulation can be recast in a linear system of equations involving the vector of random nodal values $\mathbf{u}(\theta)$, the stochastic analogous of (6),

$$[\mathbf{A}](\theta)\mathbf{u}(\theta) = \mathbf{b}, \quad [\mathbf{A}]_{\mathrm{n},\mathrm{n}'}(\theta) = \int_\Omega \kappa(\boldsymbol{x},\theta)\boldsymbol{\nabla}\Phi_{\mathrm{n}}(\boldsymbol{x})\cdot\boldsymbol{\nabla}\Phi_{\mathrm{n}'}(\boldsymbol{x})d\boldsymbol{x}. \tag{27}$$

2.2.2 Direct Monte Carlo sampling

A common approach to solve the discrete stochastic problem is to resort to Monte Carlo (MC) sampling methods. In a MC approach, samples $\kappa(\boldsymbol{x},\theta_i)$ of the the random field are generated, leading to samples of the stochastic matrix $[\mathbf{A}](\theta_i)$ and corresponding realizations $u(\boldsymbol{x},\theta_i) \in V^{\mathrm{FE}}$ of the stochastic solution. Note that different samples can be computed in parallel. Different moments and statistics of the solution can be computed, in particular the solution mean and the two-points correlations can be estimated from

$$\mathbb{E}\left[u(\boldsymbol{x},\cdot)\right] = \lim_{M\to\infty}\frac{1}{M}\sum_{i=1}^{M} u(\boldsymbol{x},\theta_i), \quad \mathbb{E}\left[u(\boldsymbol{x},\cdot),u(\boldsymbol{x}',\cdot)\right] = \lim_{M\to\infty}\frac{1}{M}\sum_{i=1}^{M} u(\boldsymbol{x},\theta_i)u(\boldsymbol{x}',\theta_i).$$

The computational complexity of the method is thus proportional to the number of samples M one uses in the MC estimation, and there is an obvious interest in reducing the computational cost of generating individual samples. Applying efficient deterministic strategies is therefore critical, and for this purpose MC is well suited to reuse the domain decomposition method detailed in the previous section. To do so, we can first derive formally the stochastic form of the condensed problem,

$$\widehat{[\mathbf{A}]}(\theta)\ \mathbf{u}_\Gamma(\theta) = \hat{\mathbf{b}}(\theta), \tag{28}$$

where,

$$\widehat{[\mathbf{A}]}(\theta) = [\mathbf{A}_{\boldsymbol{\Gamma},\boldsymbol{\Gamma}}](\theta) - [\mathbf{A}_{\boldsymbol{\Gamma},\mathrm{in}}](\theta)[\mathbf{A}_{\mathrm{in},\mathrm{in}}]^{-1}(\theta)[\mathbf{A}_{\mathrm{in},\boldsymbol{\Gamma}}](\theta), \quad \hat{\mathbf{b}}(\theta) = \mathbf{b}_\Gamma - [\mathbf{A}_{\boldsymbol{\Gamma},\mathrm{in}}](\theta)[\mathbf{A}_{\mathrm{in},\mathrm{in}}]^{-1}(\theta)\mathbf{b}_{\mathrm{in}}, \tag{29}$$

and subsequently proceed with the MC sampling of the condensed problem to yield samples of subdomain boundary nodal values $\mathbf{u}_\Gamma(\theta_i)$ and solution $u(\boldsymbol{x},\theta_i)$. If one uses a matrix-free iterative scheme without explicit construction of $\widehat{[\mathbf{A}]}(\theta_i)$, the heaviest part of the computation is dedicated to the assembly of the local problem operators $[\mathbf{A}_{\mathrm{in},\mathrm{in}}]^{(d)}(\theta_i)$ and possibly their factorizations.

At this point we remark that, contrary to the deterministic case, the stochastic condensed problem is going to be queried multiple times, as large values of M are generally needed to obtain well converged MC estimators. This is quite a different situation from the deterministic case where the actual assembly of $\widehat{[\mathbf{A}]}$ appears computationally too expensive if it is to be queried only once. This observation suggests that there could be an interest in actually assembling the stochastic condensed problem, to sample from, as the overhead of the assembly would be factorized (amortized) over the subsequent M samples. If such a strategy is feasible, one would jointly sample directly the matrix $\widehat{[\mathbf{A}]}(\theta)$ and right-hand-side $\hat{\mathbf{b}}(\theta)$, to get samples of the boundary solution $\mathbf{u}_\Gamma(\theta)$ by means of a matrix-based iterative method. As a result, one would only have to solve a unique local problem per subdomain for each sample, and only for the subdomain where the solution is sought.

To be effective, the approach just sketched would have to fulfill two conditions. First, the stochastic condensed problem matrix and right-hand-side must be represented in a format amenable to sampling. Second, the assembly overhead must remain reasonable for the method to be practical. Below, we rely on stochastic spectral expansions to approximate the problem in a suitable format; then we exploit the underlying structure of the condensed problem, namely its expression as a sum of *local* stochastic operators, to come up with representation having manageable complexity.

3 Stochastic Spectral Expansion of the Condensed Problem

3.1 PC expansion of the elliptic solution

Stochastic spectral expansions have been proposed as an alternative to Monte-Carlo methods. The key observation supporting the spectral approach is the smooth dependences of the elliptic equation solution with respect to the diffusivity coefficients. This fact motivates the expansion of the solution $u(\boldsymbol{x},\theta)$ as a series of the form

$$u(\boldsymbol{x},\theta) = \sum_{\boldsymbol{\alpha}} u_{\boldsymbol{\alpha}}(\boldsymbol{x})\Psi_{\boldsymbol{\alpha}}(\theta),$$

where the $\Psi_{\boldsymbol{\alpha}}$ are random functionals. Typically, one starts by approximating the diffusion field κ as a functional of a *finite* set of $\mathrm{N}_\kappa \geqslant 1$ independent random variables $\boldsymbol{\eta}(\theta)$ with known density:

$$\kappa(\boldsymbol{x},\theta) \approx \hat{\kappa}(\boldsymbol{x},\boldsymbol{\eta}(\theta)). \tag{30}$$

Such parametrization of κ can be obtained for instance by computing Karhunen-Loève expansions as in the following sections. As a result, the solution is a functional of $\boldsymbol{\eta}(\theta)$, and the *truncated* spectral expansion becomes

$$u(\boldsymbol{x},\theta) \approx \tilde{u}(\boldsymbol{x},\boldsymbol{\eta}(\theta)) = \sum_{\boldsymbol{\alpha}\in\mathcal{A}} u_{\boldsymbol{\alpha}}(\boldsymbol{x})\Psi_{\boldsymbol{\alpha}}(\boldsymbol{\eta}(\theta)). \tag{31}$$

Classically, one considers expansions using orthonormal functionals $\Psi_{\boldsymbol{\alpha}}$, in particular polynomials in $\boldsymbol{\eta}$. In this case, the expansion in (31) is called the Polynomial Chaos expansion of u. The multi-index $\boldsymbol{\alpha} = (\alpha_1,\ldots,\alpha_{\mathrm{N}_\kappa}) \in \mathbb{N}^{\mathrm{N}_\kappa}$ indicates the maximal polynomial degree α_k in each component $\boldsymbol{\eta}_k$, and we shall denote $|\boldsymbol{\alpha}| = \sum_{k=1}^{\mathrm{N}_\kappa} \alpha_k$ the total degree of $\Psi_{\boldsymbol{\alpha}}$. The functionals are orthonormal in the sense that

$$\mathbb{E}\left[\Psi_{\boldsymbol{\alpha}}\Psi_{\boldsymbol{\beta}}\right] = \int_\Theta \Psi_{\boldsymbol{\alpha}}(\boldsymbol{\eta}(\theta))\Psi_{\boldsymbol{\beta}}(\boldsymbol{\eta}(\theta))d\mu(\theta) = \begin{cases} 1, & \boldsymbol{\alpha}=\boldsymbol{\beta}, \\ 0, & \text{otherwise.} \end{cases}$$

Finally, the summation in (31) is restricted to $\boldsymbol{\alpha}$ belonging to the multi-index set $\mathcal{A} \subset \{\boldsymbol{\alpha} \in \mathbb{N}^{\mathrm{N}_\kappa}\}$. Different strategies can be used to define this set; without loss of generality and unless specified otherwise, we shall control $\mathcal{A}$ by the maximal total polynomial degree N_o of the expansion, setting

$$\mathcal{A} = \{\boldsymbol{\alpha} \in \mathbb{N}^{\mathrm{N}_\kappa}, |\boldsymbol{\alpha}| \leqslant \mathrm{N}_\mathrm{o}\}.$$

Under mild assumptions on κ, the solution u has exponentially converging expansions with respect to the number of random variables in $\boldsymbol{\eta}$ and with the polynomial degree N_o of the truncated form of the expansion. Regarding the computation of the expansion coefficients $u_{\boldsymbol{\alpha}}$, different approaches have been proposed and improved over the last 25 years. These include the Galerkin and non-intrusive methods. In Galerkin type methods, one requires the equation residual to be orthogonal to the stochastic approximation space, with possibly the need of deriving from the original stochastic elliptic operator a set of coupled problems for the expansion coefficient $u_{\boldsymbol{\alpha}}$. Non-intrusive methods are sampling-based approaches, where one directly estimate the $u_{\boldsymbol{\alpha}}$ from a set of resolutions of the deterministic elliptic problem corresponding to realizations of $\boldsymbol{\eta}$.

The main limitation in the applicability of the spectral expansions to the solution of stochastic elliptic problems comes with the number of terms in the series that can be prohibitively large in some situations. This has motivated adaptive strategies, in particular low rank approximations. However, the case of diffusion fields κ with large variances and short correlation lengths remains challenging because it requires, first, a large number N_κ of random variables for their parametrization in (30) and, second, a high degree $\mathrm{N_o}$ for the polynomial expansions. The issue can be seen from the expression of the number of terms in an expansion (with total degree truncation) involving N_κ random variables and degree $\mathrm{N_o}$:

$$\mathrm{P} = |\mathcal{A}| = \frac{(\mathrm{N}_\kappa + \mathrm{N_o})!}{\mathrm{N}_\kappa!\mathrm{N_o}!}. \tag{32}$$

Although more advanced truncation strategies have been proposed, in particular adapting the expansion order in the different variables of $\boldsymbol{\eta}$ [], the relation (32) shows that cases of large dimensional problems ($\boldsymbol{\eta}$) remain critical even for low orders, and that it is highly desirable to keep the dimension of $\boldsymbol{\eta}$ as small as possible.

3.2 Spectral expansion of the condensed problem

3.2.1 Local parametrization

It is well known that the dimensionality of $\boldsymbol{\eta}$ relates to the intrinsic stochastic dimensionality of κ which, roughly speaking, corresponds to the minimal number of random variables in its parametrization. It is also known from the properties of second-order orthogonal decompositions *à la* Karhunen-Loève, that the stochastic dimensionality of a field over a *fixed* domain increases as its correlation length decreases. The stochastic dimensionality of a stationary process is actually governed by the ratio of correlation length and domain extension, expressing the fact that a lower number of random variables can be used to parametrize the process over a subdomain. This feature is exploited in [10] where we proposed a reduced basis method to perform Karhunen-Loève decompositions (factorization of correlation functions) within a domain decomposition framework. Specifically, the stochastic parametrization of κ is written as

$$\hat{\kappa}(\boldsymbol{x},\theta) = \sum_{d=1}^{\mathrm{D}} \mathbb{1}_{\Omega^{(d)}}(\boldsymbol{x})\hat{\kappa}^{(d)}(\boldsymbol{x},\boldsymbol{\eta}^{(d)}), \quad \mathbb{1}_{\Omega^{(d)}}(\boldsymbol{x}) = \begin{cases} 1 & \boldsymbol{x} \in \Omega^{(d)}, \\ 0 & \text{otherwise.} \end{cases} \tag{33}$$

In (33), $\mathbb{1}_{\Omega^{(d)}}$ is the indicator function of a subdomain and $\hat{\kappa}^{(d)}$ is a *local* approximation of κ over $\Omega^{(d)}$ which uses local random variables $\boldsymbol{\eta}^{(d)}$ whose number $\mathrm{N}_\kappa^{(d)}$ will be shown to be much less than for the global parametrization of κ over the whole domain Ω, see [10].

One cannot express the elliptic equation solution u in a format similar to (33), using the same local random vectors $\boldsymbol{\eta}^{(d)}$ as for the parametrization of κ. Indeed, the stochastic solution u over a subdomain $\Omega^{(d)}$ depends on the whole set of local random variables $\{\boldsymbol{\eta}^{(d)}, d = 1,\dots,\mathrm{D}\}$, because of the elliptic nature of the problem. In other words, it is not possible to expand u for $\boldsymbol{x} \in \Omega^{(d)}$ in terms of the local random variables $\boldsymbol{\eta}^{(d)}$ only. This prevents the direct construction of a local expansion for $u(\boldsymbol{x} \in \Omega^{(d)}, \theta)$ using a low dimensional polynomial basis constructed on the reduced set of $\mathrm{N}_\kappa^{(d)}$ local random variables in $\boldsymbol{\eta}^{(d)}$. Alternatively, the construction of a global expansion of $u(\boldsymbol{x} \in \Omega, \theta)$ using the whole set of local variables would require a prohibitively large PC basis as it would involve $\mathrm{N}_\kappa = \sum_{d=1}^{\mathrm{D}} \mathrm{N}_\kappa^{(d)}$ random dimensions. Note that the $\boldsymbol{\eta}^{(d)}$ will be generally not indepedent so it could be possible to reduce the global number of random variables, but the approach would eventually remain at least as costly as for a direct parametrization of κ as in (30). We thus consider a different approach in the following, avoiding to seek a PC expansion of the solution.

3.2.2 Local PC expansion of the condensed problem

Although computational complexity reduction using direct local expansions of the solution cannot be achieved, we propose to take advantage of the low dimensionality of the local parametrization of κ to accelerate the Monte-Carlo sampling of the stochastic solution discussed in the previous section. The key idea supporting the proposed approach comes from the following observation. Contrary to the solution over a subdomain, the contribution to the condensed problem of the subdomain can be approximated solely in terms of its local random variables $\boldsymbol{\eta}^{(d)}$. Specifically, we can write

$$\widehat{[\mathbf{A}]}(\theta) = \sum_{d=1}^{\mathrm{D}} \widehat{[\mathbf{A}]}^{(d)}(\theta) \approx \sum_{d=1}^{\mathrm{D}} \widehat{[\mathbf{A}]}^{(d)}(\boldsymbol{\eta}^{(d)}(\theta)), \tag{34}$$

with similar expressions of the right-hand-side $\widehat{\mathbf{b}}(\theta)$. It suffices to remind that, in the deterministic case, the local condensed operator $\widehat{[\mathbf{A}]}^{(d)}$ and right-hand-side $\widehat{\mathbf{b}}^{(d)}$ can be determined solving local elliptic problems over $\Omega^{(d)}$ with selected boundary conditions. Our objective is therefore to construct local PC approximation as

$$\widehat{[\mathbf{A}]}^{(d)}(\boldsymbol{\eta}^{(d)}(\theta)) \approx \widetilde{[\mathbf{A}]}^{(d)}(\boldsymbol{\eta}^{(d)}(\theta)) \doteq \sum_{\boldsymbol{\alpha}\in\mathcal{A}^{(d)}} \widehat{[\mathbf{A}]}^{(d)}_{\boldsymbol{\alpha}} \Psi_{\boldsymbol{\alpha}}(\boldsymbol{\eta}^{(d)}). \tag{35}$$

To this end, we rely on the decomposition of κ in (33) and we first consider the stochastic problems which are the counterpart of (20), namely for $\mathrm{n} \in \mathcal{N}^{(d)}$ we solve

$$\left[\mathbf{A}^{(d)}_{\mathrm{in,in}}\right](\boldsymbol{\eta}^{(d)})\mathbf{u}^{(d)}_{\mathrm{in,n}}(\boldsymbol{\eta}^{(d)}) = -\left[\mathbf{A}^{(d)}_{\mathrm{in},\boldsymbol{\Gamma}}\right](\boldsymbol{\eta}^{(d)})\mathbf{e}^{(d)}_{\mathrm{n}}. \tag{36}$$

The stochastic matrices $\left[\mathbf{A}^{(d)}_{\mathrm{in,in}}\right](\boldsymbol{\eta}^{(d)})$ and $\left[\mathbf{A}^{(d)}_{\mathrm{in},\boldsymbol{\Gamma}}\right](\boldsymbol{\eta}^{(d)})$ appearing in these problems now have entries of the form

$$\int_{\Omega^{(d)}} \hat{\kappa}^{(d)}(\boldsymbol{x},\boldsymbol{\eta}^{(d)})\boldsymbol{\nabla}\Phi_{\mathrm{n}}(\boldsymbol{x}) \cdot \boldsymbol{\nabla}\Phi_{\mathrm{n}'}(\boldsymbol{x})d\boldsymbol{x}.$$

Further, the solutions of the elementary problems (36) can be approximated on a *local* PC basis through

$$\mathbf{u}^{(d)}_{\mathrm{in,n}}(\boldsymbol{\eta}^{(d)}) \approx \tilde{\mathbf{u}}^{(d)}_{\mathrm{in,n}}(\boldsymbol{\eta}^{(d)}) \doteq \sum_{\boldsymbol{\alpha}\in\mathcal{A}^{(d)}} \left(\mathbf{u}^{(d)}_{\mathrm{in,n}}\right)_{\boldsymbol{\alpha}} \Psi_{\boldsymbol{\alpha}}(\boldsymbol{\eta}^{(d)}).$$

The local basis defined by the local multi-index set $\mathcal{A}^{(d)}$ may be based on different truncation strategies. In this work, we shall restrict ourselves to the simplest case of total order truncation using a fixed polynomial order $\mathrm{N_o} \geqslant 1$ for all the subdomains; the local basis cardinality $\mathrm{P}^{(d)}$ is then function of the number $\mathrm{N}^{(d)}_{\kappa}$ and given by (32). We stress that, as we expect $\mathrm{N}^{(d)}_{\kappa} \ll \mathrm{N}_{\kappa}$, $\mathrm{P}^{(d)}$ is much reduced because of its exponential dependence on the number of random variables ($\mathrm{N}^{(d)}_{\kappa}$).

For the computation of the expansion coefficients $\left(\mathbf{u}^{(d)}_{\mathrm{in,n}}\right)_{\boldsymbol{\alpha}}$ we shall rely on the Galerkin approximation of (36). Specifically, we solve

$$\sum_{\boldsymbol{\alpha}\in\mathcal{A}^{(d)}} \mathbb{E}\left[\left[\mathbf{A}^{(d)}_{\mathrm{in,in}}\right]\Psi_{\boldsymbol{\alpha}}\Psi_{\boldsymbol{\beta}}\right]\left(\mathbf{u}^{(d)}_{\mathrm{in,n}}\right)_{\boldsymbol{\alpha}} = -\mathbb{E}\left[\left[\mathbf{A}^{(d)}_{\mathrm{in},\boldsymbol{\Gamma}}\right]\Psi_{\boldsymbol{\beta}}\right]\mathbf{e}^{(d)}_{\mathrm{n}}, \quad \forall\boldsymbol{\beta}\in\mathcal{A}^{(d)}. \tag{37}$$

Note that the size of this linear problem is $\mathrm{N}^{(d)}_{\mathrm{in}} \times \mathrm{P}^{(d)}$, stressing the importance of achieving low-dimensional local parameterization. We also remark that only the right-hand side of this system is changing for different $\mathrm{n} \in \mathcal{N}^{(d)}_{\Gamma}$. This can be exploited to efficiently compute the set of local solutions, for instance by pre-factorizing the linear system or employing an iterative solver designed to handle multiple right-hand sides. Further, these sequences of problems are independent from a subdomain to another, and so they can be carried out in parallel. Finally, from the PC expansion of $\mathbf{u}^{(d)}_{\mathrm{in,n}}(\boldsymbol{\eta}^{(d)})$ we derive the PC expansion of the columns $\boldsymbol{I}^{(d)}_{n}(\boldsymbol{\eta}^{(d)})$ for the subdomain contribution to the stochastic condensed operator (see (23)),

$$\boldsymbol{I}^{(d)}_{\mathrm{n}}(\boldsymbol{\eta}^{(d)}) \approx \tilde{\boldsymbol{I}}^{(d)}_{\mathrm{n}}(\boldsymbol{\eta}^{(d)}) = \sum_{\boldsymbol{\alpha}\in\mathcal{A}^{(d)}} \left(\boldsymbol{I}^{(d)}_{\mathrm{n}}\right)_{\boldsymbol{\alpha}} \Psi_{\boldsymbol{\alpha}}(\boldsymbol{\eta}^{(d)}),$$

using the Galerkin interpretation of the matrix-vector product:

$$\left(\boldsymbol{I}^{(d)}_{\mathrm{n}}\right)_{\boldsymbol{\alpha}} \doteq \sum_{\boldsymbol{\beta}\in\mathcal{A}^{(d)}} \mathbb{E}\left[\left[\mathbf{A}^{(d)}_{\boldsymbol{\Gamma},\mathrm{in}}\right]\Psi_{\boldsymbol{\alpha}}\Psi_{\boldsymbol{\beta}}\right]\left(\mathbf{u}^{(d)}_{\mathrm{in,n}}\right)_{\boldsymbol{\beta}} + \mathbb{E}\left[\left[\mathbf{A}^{(d)}_{\boldsymbol{\Gamma},\boldsymbol{\Gamma}}\right]\Psi_{\boldsymbol{\alpha}}\right]\mathbf{e}^{(d)}_{n}. \tag{38}$$

A similar procedure is employed to derive the PC approximations of the stochastic subdomain contributions to the condensed problem right-hand side, namely

$$\widehat{\mathbf{b}}^{(d)}(\boldsymbol{\eta}^{d}) \approx \widetilde{\mathbf{b}}^{(d)}(\boldsymbol{\eta}^{d}) = \sum_{\boldsymbol{\alpha}\in\mathcal{A}^{(d)}} \widehat{\mathbf{b}}^{(d)}_{\boldsymbol{\alpha}} \Psi_{\boldsymbol{\alpha}}(\boldsymbol{\eta}^{(d)}). \tag{39}$$

3.3 Sampling the stochastic condensed problem

At this point, we have described a strategy to compute a composite PC expansion of the condensed problem. These approximations can be used to generate approximate samples of the solution *à la* Monte Carlo. This task amounts to sampling jointly the local random variables $\boldsymbol{\eta}^{(d)}$ of the subdomains as illustrated in the following example section. We shall denote $\boldsymbol{\eta}_i^{(d)} = \boldsymbol{\eta}^{(d)}(\theta_i)$ a sample of the local random variables; the corresponding sample of the condensed problem solution $\mathbf{u}_\Gamma(\theta_i)$ is defined through

$$\widetilde{[\mathbf{A}]}(\theta_i)\mathbf{u}_\Gamma(\theta_i) = \tilde{\mathbf{b}}(\theta_i), \tag{40}$$

where

$$\widetilde{[\mathbf{A}]}(\theta_i) = \sum_{d=1}^{\mathrm{D}} \sum_{\boldsymbol{\alpha}\in\mathcal{A}^{(d)}} \widehat{[\mathbf{A}]}_{\boldsymbol{\alpha}}^{(d)} \Psi_{\boldsymbol{\alpha}}(\boldsymbol{\eta}_i^{(d)}) \text{ and } \tilde{\mathbf{b}}(\theta_i) = \sum_{d=1}^{\mathrm{D}} \sum_{\boldsymbol{\alpha}\in\mathcal{A}^{(d)}} \hat{\mathbf{b}}_{\boldsymbol{\alpha}}^{(d)} \Psi_{\boldsymbol{\alpha}}(\boldsymbol{\eta}_i^{(d)}). \tag{41}$$

The key advantage of the proposed approach is the substitution of the exact condensed operator with its composite PC approximation. As a result, applying $\widetilde{[\mathbf{A}]}(\theta_i)$ to a given vector in an iterative solution method for (40) is much less costly than having to solve local problems in the classical method. Indeed, forming the reduced problem essentially amounts to evaluating polynomial expansions for the subdomains contribution, which can be made in parallel. Obviously, this comes at the cost of having first to compute the PC approximation $\widehat{[\mathbf{A}]}$ of $\widehat{[\mathbf{A}]}$ in the preprocessing stage; however this overhead is factorized over the number of samples subsequently generated.

Note that when the sample $\mathbf{u}_\Gamma(\theta_i)$ solving (40) is obtained, the local problems can be independently solved for (and only for) the subdomains where the solution is sought. Specifically, once $\mathbf{u}_\Gamma(\theta_i)$ is computed, one can solve (independently)

$$\left[\mathbf{A}_{\text{in,in}}^{(d)}\right](\boldsymbol{\eta}_i^{(d)})\, \mathbf{u}_{\text{in}}^{(d)}(\theta_i) = \mathbf{b}_{\text{in}}^{(d)} - \left[\mathbf{A}_{\text{in},\boldsymbol{\Gamma}}^{(d)}\right](\boldsymbol{\eta}_i^{(d)})\, \mathbf{u}_\Gamma^{(d)}(\theta_i), \tag{42}$$

to get the finite element approximation of $u(\boldsymbol{x},\theta_i)$ for $\boldsymbol{x} \in \Omega^{(d)}$.

3.4 Monte Carlo algorithm and implementation

The proposed method thus involves two distinct steps as summarized in Algorithm 1: a preprocessing stage where the PC approximations of the condensed problem are constructed and the Monte Carlo sampling of the approximate solution.

Given a partition of Ω into D subdomains and associated local random variables for the parametrization of κ, the preprocessing stage is dedicated to the construction of the local approximations for the condensed problem. The treatments of different subdomains are fully independent and can be trivially carried out in parallel (loop starting at line 2). For each subdomain, the main computational effort is the solution of a local stochastic elliptic problem (with multiple right-hand-sides), whose size is made reasonable by considering sufficiently many subdomains so $\mathrm{N}_{\text{in}}^{(d)}$ and $\mathrm{N}_\kappa^{(d)}$ are sufficiently small. The memory requirement to store the local PC expansions $\widetilde{[\mathbf{A}]}^{(d)}(\boldsymbol{\eta}^{(d)})$ and $\tilde{\mathbf{b}}^{(d)}(\boldsymbol{\eta}^{(d)})$ is proportional to $\mathrm{N}_\Gamma^{(d)} \times \mathrm{N}_\Gamma^{(d)} \times \mathrm{P}^{(d)}$ and $\mathrm{N}_\Gamma^{(d)} \times \mathrm{P}^{(d)}$ respectively.

In the sampling stage, starting at line 11, one generates joint samples $\boldsymbol{\eta}^{(d)}(\theta_i)$ and evaluates the subdomain contributions to the sample condensed problems. This involves polynomial evaluations which can be carried out in parallel over distinct subdomains (loop starting at line 13). The resulting sampled problem (40) can be solved for instance by means of an iterative method, without having to resort to any local problem solve. When the sample $\mathbf{u}_\Gamma(\theta_i)$ is computed, see line 16, one can eventually recompute classically the solution over subdomains of interest (loop starting at line 17). Again these final solves over different subdomains can be carried out in parallel.

The solver for the approximate condensed problem (40) can eventually be implemented in parallel, and another advantage of the proposed approach is the possibility of relying on a preconditioned iterative method. For instance, we show in the example section how to take advantage of the manageable dimension and explicit representation of the condensed operator to determine an effective preconditioner for the sampled problems. This preconditioner is a carefully selected realization of $\widetilde{[\mathbf{A}]}$, whose LU decomposition is computed at the preprocessing stage and subsequently employed in the sampling stage to further accelerate the convergence of the iterative solves.

Algorithm 1: Proposed method.

```
Data: Partitioning of the domain, local parametrization of $\kappa$, polynomial order $\mathrm{N_o}$
Result: Produce $M$ samples of the stochastic solution
1  Preprocessing stage: approximation of the condensed problem
2  for subdomain with index $d = 1, \dots, \mathrm{D}$ do
3      Set local PC basis
4      Compute PC expansion $\widetilde{\mathbf{b}}^{(d)}(\boldsymbol{\eta}^{(d)})$
5      for boundary node $\mathrm{n} \in \mathcal{N}_\Gamma^{(d)}$ do
6          Solve local stochastic problem (37)
7          Set PC expansion of n-th column of $\widetilde{[\mathbf{A}]}^{(d)}(\boldsymbol{\eta}^{(d)})$ using (38)
8      end for
9  end for
10 Monte-Carlo Sampling Stage: Generate approximate samples of solution
11 for sample index $i = 1, \dots, M$ do
12     Generate a random sample of $\boldsymbol{\eta}_i = (\boldsymbol{\eta}_i^{(1)} \dots \boldsymbol{\eta}_i^{(\mathrm{D})})$
13     for subdomain with index $d = 1, \dots, \mathrm{D}$ do
14         Compute $\widetilde{[\mathbf{A}]}^{(d)}(\boldsymbol{\eta}_i^{(d)})$ and $\widetilde{\mathbf{b}}^{(d)}(\boldsymbol{\eta}_i^{(d)})$ using (41)
15     end for
16     Solve sampled condensed problem (40) for $\mathbf{u}_\Gamma(\theta_i)$
17     for subdomain with index $d = 1, \dots, \mathrm{D}$ do
18         Solve local problem (42) for the inner unknowns $\mathbf{u}_{\mathrm{in}}^{(d)}$
19     end for
20 end for
```

4 Example of Stochastic Elliptic Problem

In the following sections we illustrate the application of the proposed methods to an elliptic equation with log-normal coefficient field. The problem settings are detailed in Section 4.1. Next, we provide various convergence studies in Section 4.2 to investigate the behavior of the method with respect to its principal numerical parameters, namely the number of subdomains, D, and the PC order, $\mathrm{N_o}$, of the PC expansions of operators $\widetilde{[\mathbf{A}]}^{(d)}$ and right-hand-side $\widetilde{\mathbf{b}}^{(d)}$. In Section 4.3 we focus on the case of random coefficient κ with high variability to highlight the main mechanism driving the error in the method in extreme problems. Finally, the efficiency and parallel implementation of the method are discussed in Section 5.

4.1 Test problem

We consider the elliptic problem (24) over a two-dimensional domain consisting of the unit square, $\Omega = (0,1)^2$. We set $f(\boldsymbol{x}) = 1$ and adopt homogeneous boundary conditions as follows:

$$u(\boldsymbol{x}) = 0 \text{ for } \boldsymbol{x} \in \partial\Omega_D, \text{ and } \boldsymbol{\nabla} u \cdot \hat{\boldsymbol{n}} = 0 \text{ for } \boldsymbol{x} \in \partial\Omega_N, \tag{43}$$

where $\partial\Omega_D$ corresponds to the West, South, and East sides of the domain; $\partial\Omega_N$ corresponds to the North side of the domain; and $\hat{\boldsymbol{n}}$ is the unit normal to the boundary $\partial\Omega_N$.

For the random field κ, we assume that $\kappa - \kappa_{\min}$ is a stationary log-normal process, such that

$$G(\boldsymbol{x}, \theta) \doteq \log(\kappa(\boldsymbol{x}, \theta) - \kappa_{\min}) \sim N(\mu_G(\boldsymbol{x}), C(\boldsymbol{x}, \boldsymbol{x}')).$$

Here, we have denoted $N(\mu_G, C)$ the Gaussian process with mean μ_G and covariance function C, whereas $\kappa_{\min}$ is a small positive constant ensuring the well-posedness of the problem (in

L_2-sense). We shall classically a covariance function having a square exponential decay,

$$C(\boldsymbol{x}, \boldsymbol{x}') = \sigma^2 \exp\left(-\|\boldsymbol{x} - \boldsymbol{x}'\|_2^2 / L^2\right), \tag{44}$$

where L is the correlation length and σ^2 the variance. In the following, we use $L = 0.1$, unless otherwise indicated.

For the local parametrization of the process, we consider the *local* Karhunen-Loeve expansion of G over each of the subdomain. Denoting $G^{(d)}$ the restriction of G over $\Omega^{(d)}$, we have

$$G^{(d)}(\boldsymbol{x}, \theta) = \mu_G(\boldsymbol{x}) + \sum_{k>1} \sqrt{\lambda_k^{(d)}} g_k^{(d)}(\boldsymbol{x}) \eta_k^{(d)}(\theta), \tag{45}$$

where the $\lambda_k^{(d)}$ and $g_k^{(d)}$ are the (dominant) eigenvalues and normalized eigenfunctions of the covariance satisfying

$$\int_{\Omega^{(d)}} C(\boldsymbol{x}, \boldsymbol{x}') g_k^{(d)}(\boldsymbol{x}') d\boldsymbol{x}' = \lambda_k^{(d)} g_k^{(d)}(\boldsymbol{x}). \tag{46}$$

It is a standard result that the random variables in the KL expansion above are independent standard Gaussian random variables, that is $\eta_k^{(d)} \sim N(0,1)$. Obviously, the KL expansion must be truncated; we shall truncate (45) to the $\mathrm{N}_\kappa^{(d)}$ first dominant (largest eigenvalues); accordingly, we have:

$$G^{(d)}(\boldsymbol{x}, \theta) \approx \hat{G}^{(d)}(\boldsymbol{x}, \boldsymbol{\eta}^{(d)}(\theta)) = \mu_G(\boldsymbol{x}) + \sum_{k=1}^{\mathrm{N}_\kappa^{(d)}} \sqrt{\lambda_k^{(d)}} g_k^{(d)}(\boldsymbol{x}) \eta_k^{(d)}(\theta) \tag{47}$$

and where $\mathrm{N}_\kappa^{(d)}$ is selected from the following criteria (see [10]):

$$\sum_{k=1}^{\mathrm{N}_\kappa^{(d)}} \lambda_k^{(d)} \geqslant (1 - \epsilon_G) \sigma^2 \frac{|\Omega^{(d)}|}{|\Omega|}. \tag{48}$$

Here, $\epsilon_G < 1$ is a small positive constant measuring the approximation error in the L_2-sense and $|\Omega|$ (resp. $|\Omega|^{(d)}$) is the volume of the domain Ω (resp. $\Omega^{(d)}$). Extending to zero the eigenfunctions outside of their respective supports Ω^d, we end up with

$$\kappa \approx \hat{\kappa} = \kappa_{\min} + \sum_{d=1}^{\mathrm{D}} \mathbb{1}_{\Omega^{(d)}}(\boldsymbol{x}) \hat{\kappa}^{(d)}(\boldsymbol{x}, \boldsymbol{\eta}^{(d)}), \quad \hat{\kappa}^{(d)}(\boldsymbol{x}, \boldsymbol{\eta}^{(d)}) = \exp\left[\hat{G}^{(d)}(\boldsymbol{x}, \boldsymbol{\eta}^{(d)})\right], \tag{49}$$

which has a structure similar to (33).

For the Monte Carlo sampling of the problem, we will have to sample jointly the $\boldsymbol{\eta}^{(d)}$. Since we know that the random variable are Gaussian and centered, we must provide the correlation structure between the random variables $\eta_k^{(d)}$. In [10], we have shown that the correlation structure is given by

$$\sqrt{\lambda_l^{(d)} \lambda_{l'}^{(d')}} \mathbb{E}\left[\eta_l^{(d)} \eta_{l'}^{(d')}\right] = \iint\limits_{\Omega^{(d)} \times \Omega^{(d')}} g_l^{(d)}(\boldsymbol{x}) g_{l'}^{(d')}(\boldsymbol{x}') C(\boldsymbol{x}, \boldsymbol{x}') d\boldsymbol{x} d\boldsymbol{x}'.$$

In particular, one observes that by construction the $\eta_l^{(d)}$ of a subdomain are uncorrelated as expected. Sampling the whole set of $\boldsymbol{\eta} = \{\boldsymbol{\eta}^{(d)}, d = 1, \ldots, \mathrm{D}\}$ can be achieved by standard techniques, e.g., decomposing the covariance matrix.

As an illustration of the parametrization of the random field κ, we provide in the top row of Figure 1 three realizations for $\mu_G = 0$ and an increasing value of σ^2 of G from left to right. In these examples, the number of subdomains is set to $\mathrm{D} = 480$ and the boundaries of the subdomains are outlined in the plot. With $L = 0.1$ and $\epsilon_G = 0.01$, a total of $\mathrm{N}_\kappa = 178$ would be necessary in a global construction. Instead, one only needs $\mathrm{N}_\kappa^{(d)} = 3$ random variables per subdomain for the same level of approximation error, ϵ_G. The plots show the effect of varying σ^2 with its direct impact on the range of variability for κ, which is roughly 10 times lager for $\sigma^2 = 0.5$ compared to the case with $\sigma^2 = 0.05$. The plots also illustrate the spatial structure of the fields with multiple local minima and maxima, due to the small correlation length, and the exponentiation effects that emphasizes the maxima and stiffen the gradients.

4.2 Validation of the method

Unless specified otherwise, the computation of this section uses $L = 0.1$ and a finite element mesh having $\mathrm{N_e}$ = 16,441 triangular quadratic elements and $\mathrm{N_n}$ = 32,747 unknowns.

4.2.1 Solution samples

We first verify that the proposed method with PC approximation of the condensed problem approximates the MC samples obtained with the original approach described in Section 2.2.2. To this extent, we refer to our approach as the DD-PC method and denote $\hat{u}(\boldsymbol{\eta}_i)$ a corresponding finite element solution sample, while $u(\boldsymbol{\eta}_i)$ is a finite element solution sample for the direct-sampling method. For fairness, when comparing two solution samples of $\hat{u}$ and u we use the same approximation of the random field $\hat{\kappa}(\boldsymbol{\eta}_i)$, so their difference $u - \hat{u}$ is solely due to the PC approximation error of the condensed problem.

First, we look at three different realizations of $\kappa(\boldsymbol{\eta}_i)$ corresponding to different variances for the underlying Gaussian process G. The realizations of κ are shown in the top row of Figure 1. The second row of Figure 1 shows the difference between $u(\boldsymbol{\eta}_i)$ and the corresponding mean, $\mathbb{E}[u]$, which enables us to highlight the complexity and length-scales in the solution samples. (The means, $\mathbb{E}[u]$, are depicted in the top row of Figure 2). Finally, the third row of Figure 1 depicts the differences between the realizations computed with the DD-PC and the direct method. Here, the DD-PC solutions are computed using PC approximations with order $\mathrm{N_o} = 2$ for all the 3 variances of G.

Focusing on the case with lowest variance, $\sigma^2 = 0.05$ (left column), the realization $u(\theta_i)$ is seen to be rather smooth, with differences less than 6×10^{-4} between u and $\hat{u}$. As σ^2 is increased to 0.20 (center column) the realization has now steeper gradients whereas the error level is now as high as 2×10^{-3}, roughly 1% of the maximum of $\mathbb{E}[u]$. For the largest variance $\sigma^2 = 0.50$, the solution presents even steeper gradients and the peak error is as high as 10% of the maximum of $\mathbb{E}[u]$. These observations are expected, because with increasing σ^2 a higher PC order $\mathrm{N_o}$ would be needed to achieve a certain relative accuracy in the local condensed problem. This is verified in the following.

4.2.2 Convergence with PC order

We now analyze the behavior of the DD-PC method, starting from the Monte Carlo error in the estimation of the mean of the finite element solution, namely $\mathbb{E}[\hat{u}] - \mathbb{E}[u]$. These errors are reported in Figure 2; shown are the mean fields ($\mathbb{E}[u]$, top row) and error fields for two expansion orders ($\mathrm{N_o} = 2$ in the middle row and $\mathrm{N_o} = 6$ in the bottom row) for the DD-PP method and the three variances σ^2 as before. Again, a total of D = 480 subdomains is used. Focusing first on the lower order case, $\mathrm{N_o} = 2$, we observe that the error increases with σ^2, with higher values in Ω where $\mathbb{E}[u]$ is larger. This indicates that the DD-PC method is biased. Note also that in the case with $\sigma^2 = 0.05$, when the error on the mean is the lowest, the field $\mathbb{E}[\hat{u}] - \mathbb{E}[u]$ appears noisy. This is due to the finite number of samples used in the Monte Carlo estimate of the expectations (M = 500,000) which induces a sampling error that is significant compared to the true ($M = \infty$) value of error on the mean solution. Increasing the PC order to $\mathrm{N_o} = 6$ is seen to reduce by several orders of magnitude the error in mean of the DD-PC method. In fact, with $\mathrm{N_o} = 6$ the error is so low that even for the largest σ^2 the MC sampling error remains significant and visible, whereas for the smallest σ^2 it is completely dominant.

To better understand the impact of $\mathrm{N_o}$ on the bias in the DD-PC method, we define the normalized L_2 error on the mean, ϵ_{mean}, according to:

$$\epsilon^2_{\text{mean}} = \frac{\|\mathbb{E}[\hat{u}] - \mathbb{E}[u]\|^2_{L_2(\Omega)}}{\|\mathbb{E}[u]\|^2_{L_2(\Omega)}}, \quad \|u\|_{L_2(\Omega)} \doteq \int_\Omega |u(\boldsymbol{x})|^2 d\boldsymbol{x}. \tag{50}$$

In practice the mean solutions $\mathbb{E}[\hat{u}]$ and $\mathbb{E}[u]$ are estimated by their empirical averages using M Monte Carlo samples. We report in Figure 3 the evolution with M of the estimate of ϵ_{mean} for the different values of $\mathrm{N_o}$ and σ^2. We observe that for small values of M the error norm ϵ_{mean} is overestimated because of the sampling error. The sampling error decreases as M increases, and

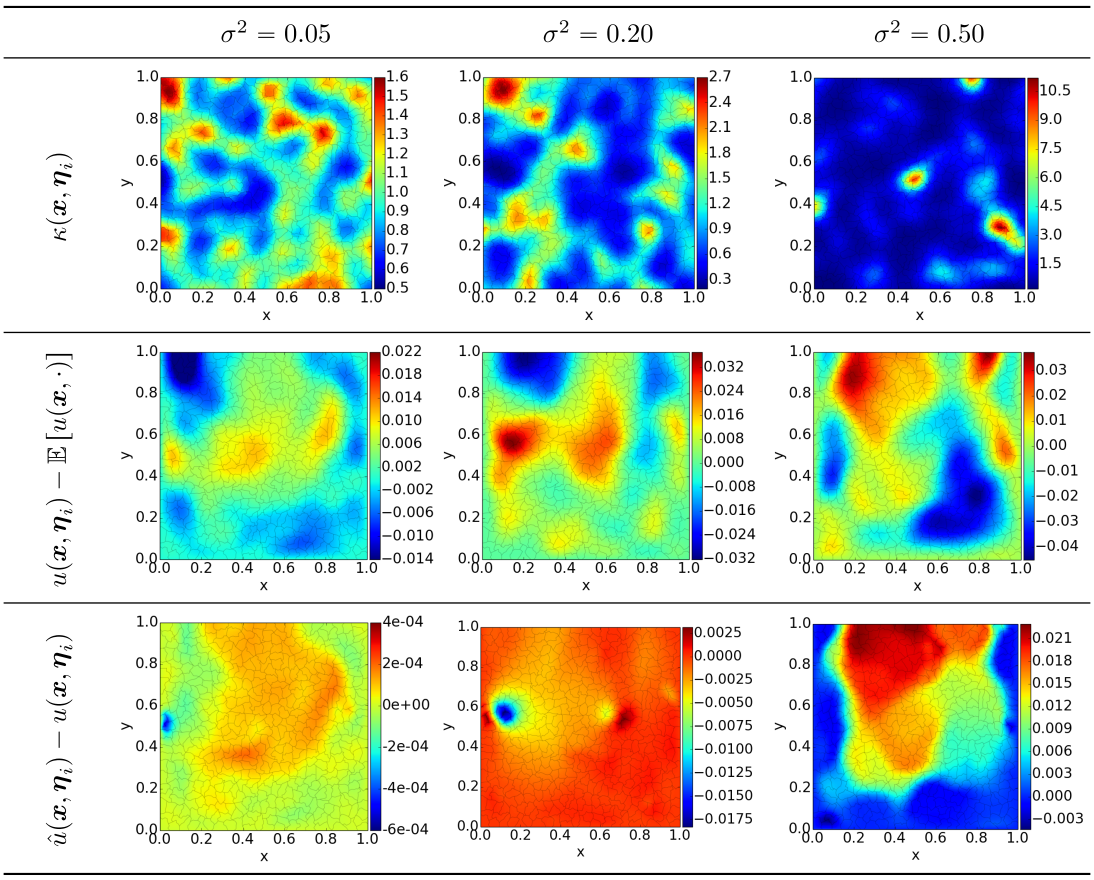


Figure 1: Realizations of κ (top row), deviation to the mean, $u - \mathbb{E}[u]$, of the direct solution (middle row) and differences between DD-PC and direct solutions (bottom row). The columns correspond to different realizations of κ drawn at random using Gaussian fields with increasing variance: $\sigma^2 = 0.05$, 0.2 and 0.5 from left to right. The DD-PC solutions use $\mathrm{N_o} = 2$, and $\mathrm{D} = 480$ (the subdomains partition is shown in all the figures).

for M large enough we see that ϵ_{mean} converges to a non-zero value, reflecting the bias in the DD-PC method. Moreover, as we saw before, the bias depends on both σ and $\mathrm{N_o}$. Specifically, higher values of σ result in higher errors on the mean, and higher values of $\mathrm{N_o}$ increase the accuracy of the PC expansion and reduce the bias. An important remark is that for high polynomial orders the sampling error will be dominant unless a large number of samples is used in estimating any desired statistic. Thus, there is not point in using a large polynomial order for a small sample size.

The convergence with $\mathrm{N_o}$ of the DD-PC method is not restricted to the mean solution but can be expected for other quantities of interest derived from u, albeit possibly with different rates. For instance, we report in Figure 4 the convergence of the error in the standard deviation of u, namely $\mathrm{Std}[\hat{u}(\boldsymbol{x})] - \mathrm{Std}[u(\boldsymbol{x})]$ for $\mathrm{N_o} = 2$ and $\mathrm{N_o} = 6$, and the 3 values of the variance σ^2. The plots show a similar trend as for the mean solution, although the spatial structure of the standard deviation error appears to depend more heavily on $\mathrm{N_o}$.

4.3 L_2-error norm

We now consider the more generic error measure as the full (or stochastic) L_2-norm of the difference $\hat{u}(\boldsymbol{x}, \theta) - u(\boldsymbol{x}, \theta)$ and define the relative stochastic error norm as

$$\epsilon_u^2 = \frac{\mathbb{E}\left[\|\hat{u} - u\|^2_{L_2(\Omega)}\right]}{\mathbb{E}\left[\|u\|^2_{L_2(\Omega)}\right]}. \tag{51}$$

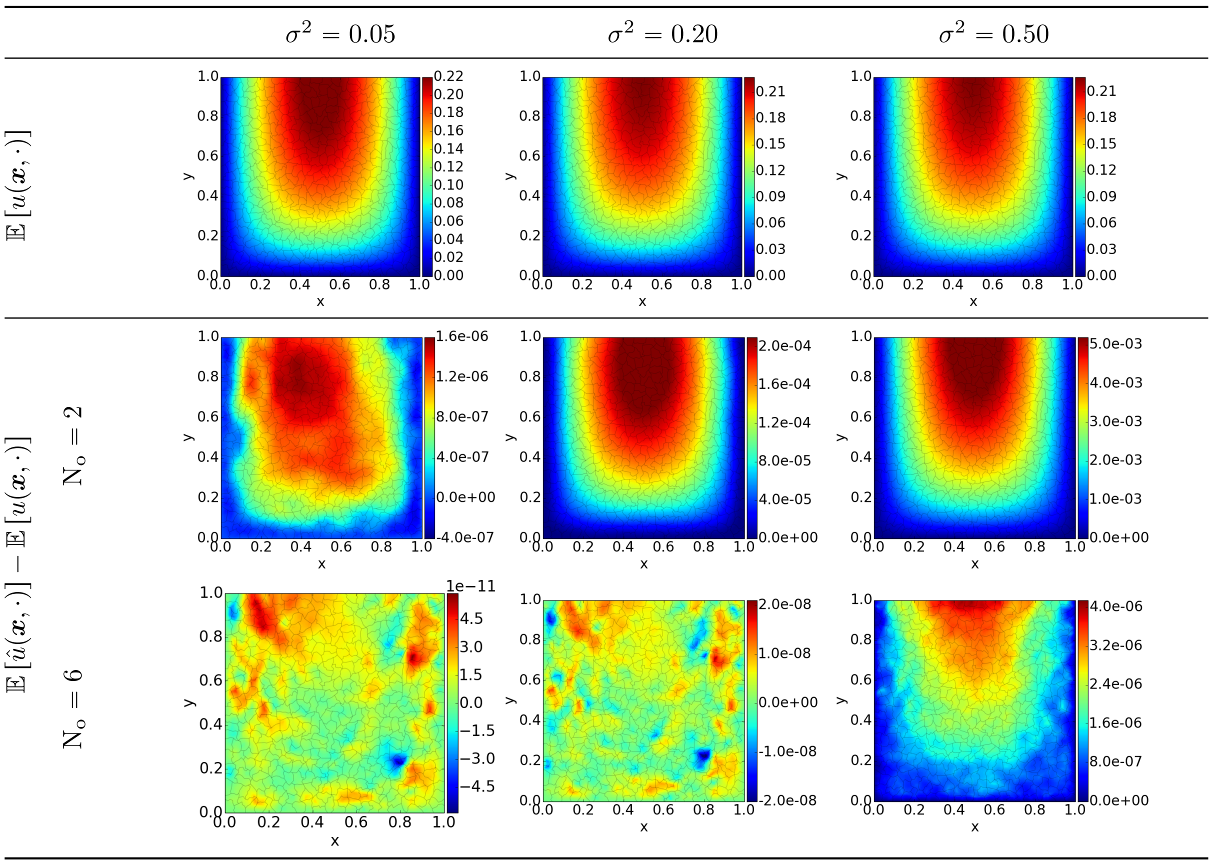
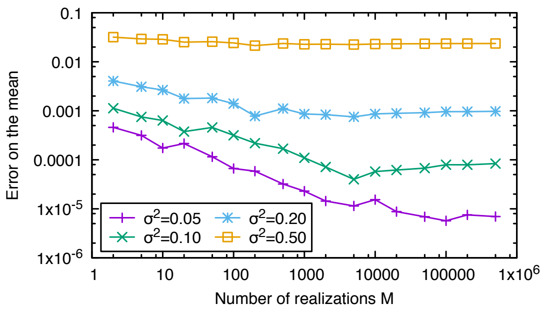
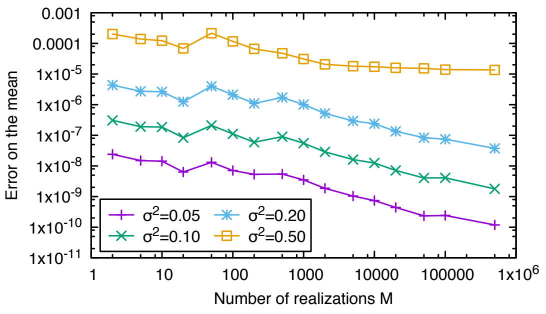


Figure 2: Mean fieds $\mathbb{E}[u(\boldsymbol{x},\cdot)]$ (top row), and DD-PC error on the mean $\mathbb{E}[\hat{u}(\boldsymbol{x},\cdot)] - \mathbb{E}[u(\boldsymbol{x},\cdot)]$ for $N_o = 3$ (middle row) and $N_o = 3$ (bottom row), and three values of σ^2, as indicated, from left to right. The computations use $M = 500{,}000$ Monte Carlo samples to estimate the expectations and $D = 480$ subdomains.



(a) Error on the mean, $N_o = 2$



(b) Error on the mean, $N_o = 6$

Figure 3: Monte Carlo estimates of the norm of the error on the mean ϵ_{mean} as a function of the number of MC samples M for different σ^2 as indicated and PC order $N_o = 2$ (left) and $N_o = 6$ (right).

Figure 5 reports ϵ_u as a function of the PC order N_o. Shown are plots for different values of σ^2 and curves for different D . The relative error on the mean, ϵ_{mean}, is also shown for comparison. For $\sigma^2 = 0.05$ (left plot) we notice that the behavior of both errors is very similar, decaying monotonically with N_o, with the relative stochastic error higher than the relative error on the mean. Further, the number of subdomains D is seen to have negligible effect on the two errors. These observations are in sharp contrast with the high variability case, $\sigma^2 = 0.5$, shown in the right plot of Figure 5, where the error decay with N_o is no longer monotonic over the reported

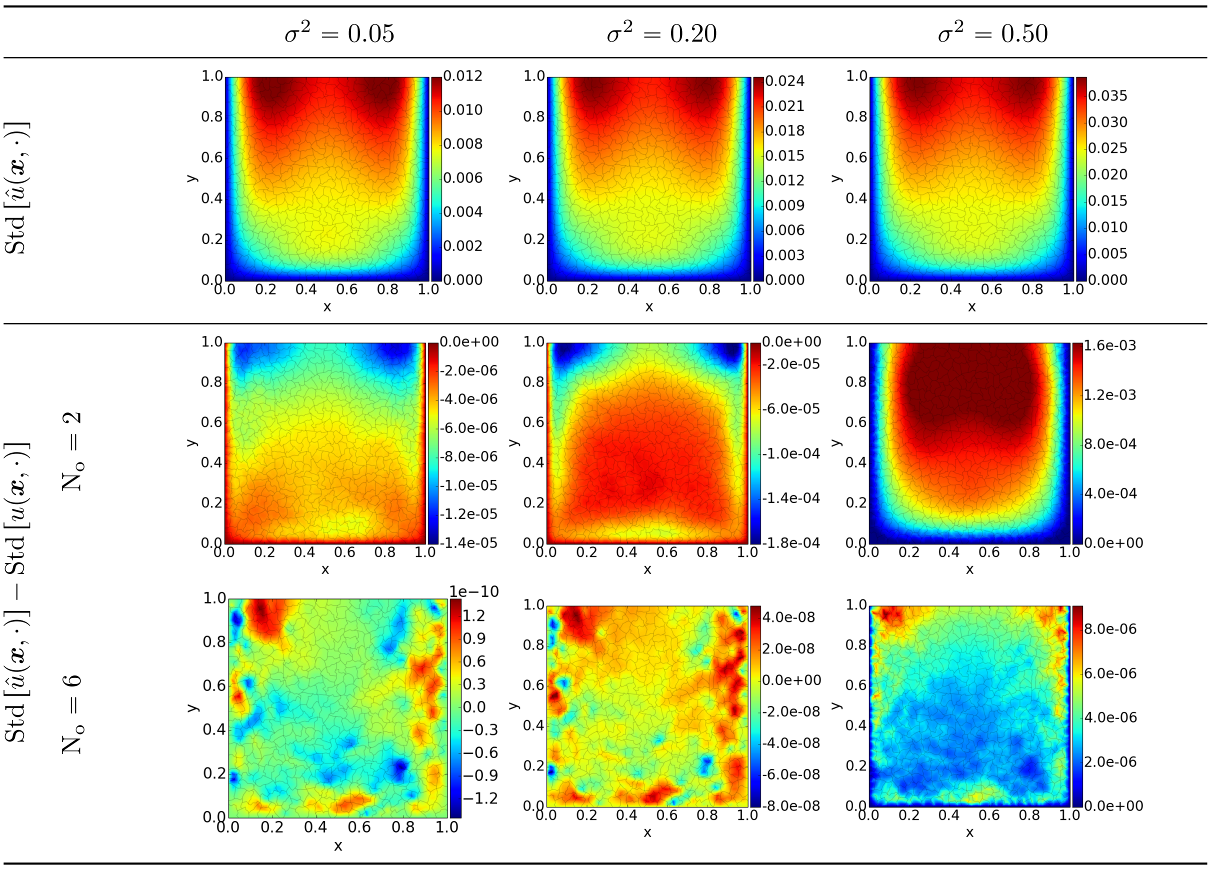
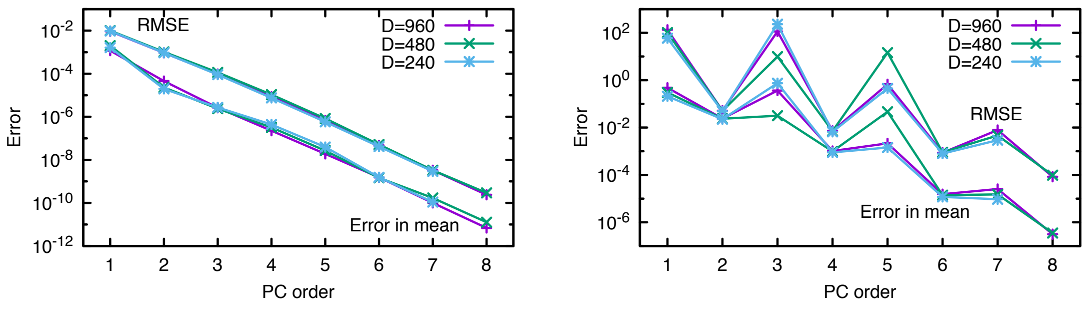


Figure 4: Standard deviation fields $\mathrm{Std}\left[u(\boldsymbol{x},\cdot)\right]$ (top row), and DD-PC error in the standard deviation $\mathrm{Std}\left[\hat{u}(\boldsymbol{x},\cdot)\right] - \mathrm{Std}\left[u(\boldsymbol{x},\cdot)\right]$ for $\mathrm{N_o} = 3$ (middle row) and $\mathrm{N_o} = 3$ (bottom row), and three values of σ^2, as indicated, from left to right. The computations use $M = 500{,}000$ Monte Carlo samples to estimate the expectations and $\mathrm{D} = 480$ subdomains.

range. In fact, the convergence curves highlight an even-odd effect with a smaller error for even order $\mathrm{N_o} = 2n$ than for the next odd order $\mathrm{N_o} = 2n + 1$. In addition, the relative stochastic error reaches dramatically large levels for odd values of $\mathrm{N_o}$ and has much more severe and non trivial dependences on D. We should remark that the case with $\sigma^2 = 0.5$ leads to a very high variability in κ and can be considered as an extreme case. In the following, we proceed to analyze the stochastic error in this large variability case.



(a) $\sigma^2 = 0.05$ (b) $\sigma^2 = 0.5$

Figure 5: Relative stochastic error ϵ_u and relative error on the mean ϵ_m as functions of the PC order $\mathrm{N_o}$, for different values of D as indicated and $\sigma^2 = 0.05$ (left plot) and 0.5 (right plot).

To better understand the error mechanism, we first reduce the computational cost of this analysis, namely by increasing the correlation length of G to $L = 1$ but keeping $\sigma^2 = 0.5$. The increased L allows to consider a coarser finite element mesh (with $\mathrm{N_e} = 1{,}630$ elements and $\mathrm{N_n} = 3{,}204$ unknowns), owing to the increased length scales in the solution u. However, this change does not affect the odd-even order effects just discussed, as shown by the convergence curves reported in Figure 6 which are similar to the previous case (Figure 5, left plot). Note that due to the coarser nature of the mesh, we also considered different values for the number of subdomains; $D = 120$, 240, and 480, instead of $D = 240$, 480, and 960.

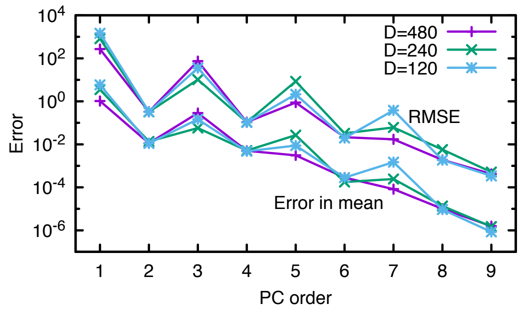


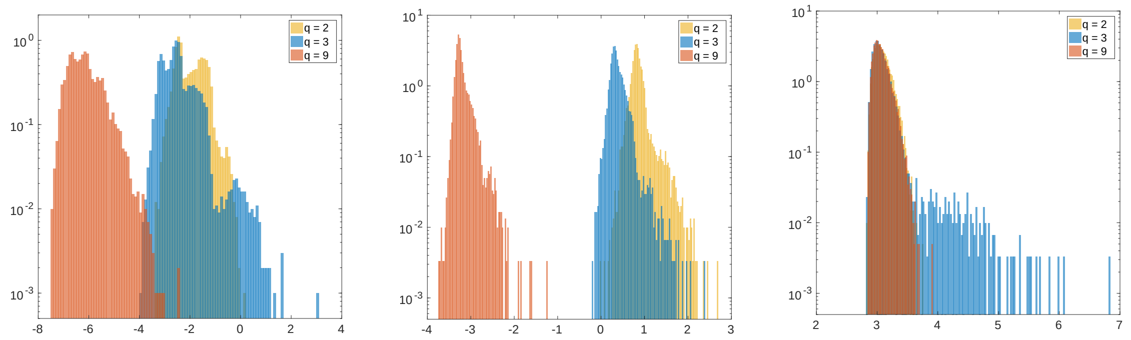
Figure 6: Relative stochastic error ϵ_u and relative norm of error on the mean ϵ_m as functions of the PC order $\mathrm{N_o}$, for different values of D as indicated. The random field G uses $\sigma^2 = 0.5$ with $L = 1$.

For the purpose of the analysis, we compute 100,000 samples of the solutions $\hat{u}(\boldsymbol{\eta}_i)$ and $u(\boldsymbol{\eta}_i)$, using the two approaches, and retrieve the corresponding samples of error norm, $\|\hat{u} - u\|_{L_2(\Omega)}$, norm of the DD-PC samples solution, $\|\hat{u}(\boldsymbol{\eta}_i)\|_{L_2(\Omega)}$, Frobenius norm of the error on the condensed problem operator, $\|\widetilde{[\mathbf{A}]}(\boldsymbol{\eta}_i) - \widehat{[\mathbf{A}]}(\boldsymbol{\eta}_i)\|_F$, and finally condition number of its PC approximation, $\mathrm{cond}\,\widetilde{[\mathbf{A}]}(\boldsymbol{\eta}_i)$. These samples are used to estimate the statistics of these quantities, which are summarized in Figure 7 using histograms in log-log scale, contrasting the cases of $\mathrm{N_o} = 2$, 3 and 9 for the PC approximation of the condensed problem.

First, the statistics of the error norms $\|\hat{u} - u\|$, depicted in Figure 7(a), are seen to be more spread for $\mathrm{N_o} = 3$ than for $\mathrm{N_o} = 2$ with a much longer tail towards the high error side: extreme samples for $\mathrm{N_o} = 3$ are standing more than 3 orders of magnitude away from the extreme samples for $\mathrm{N_o} = 2$. The presence of very large error samples induces the average-error behavior shown in Figure 6, even though the mode of the histogram for $\mathrm{N_o} = 3$ is at a lower error level compared to the mode for $\mathrm{N_o} = 2$. On the contrary, the error distribution for even order $\mathrm{N_o} = 2$ does not exhibit a long tail towards higher error values. In addition, increasing the PC order to $\mathrm{N_o} = 9$ results in a distribution of the error that remains quite broad (in the logscale) with broad right-tail but shifted to the low error values compared to $\mathrm{N_o} = 3$. Overall the error samples for $\mathrm{N_o} = 9$ remain lower than for $\mathrm{N_o} = 2$. One can conclude that the large stochastic error for odd orders is caused by a fraction of samples having abnormally very high error compared to their median error, but with a probability that decreases with increasing order.

Comparing the statistics of the error on the condensed operator $\|\widetilde{[\mathbf{A}]} - \widehat{[\mathbf{A}]}\|_F$ at different orders, shown in Figure 7(b), we observe a monotonic shift of the histograms when $\mathrm{N_o}$ increases with similar tails for both odd and even orders. This is expected as one globally improves the PC approximation with increasing $\mathrm{N_o}$. This distribution of the operator error must be contrasted with the statistics of the condensed operator condition number $\mathrm{cond}\,\widehat{[\mathbf{A}]}$ reported in Figure 7(c): the histogram for $\mathrm{N_o} = 3$ is seen to exceed by several orders of magnitude the highest values for $\mathrm{N_o} = 2$ and $\mathrm{N_o} = 9$. In fact, the histogram for $\mathrm{N_o} = 3$ reveals samples with poorly conditioned systems. Since the error on the operator itself behaves well, one can suspect the PC approximation of $\widetilde{[\mathbf{A}]}$ to induce error on the lowest part of the spectrum, that is the smallest eigenvalues and eigenfunctions of $\widehat{[\mathbf{A}]}$ (recall that $\widehat{[\mathbf{A}]}$ is symmetric positive definite).

To evidence the role of the condition number and error on the lowest eigen values of $\widetilde{[\mathbf{A}]}$ on the error, we present in Figure 8 samples of the condition number of $\widetilde{[\mathbf{A}]}(\boldsymbol{\eta}_i)$ as a function of the corresponding samples of the error $\|u - \hat{u}\|_{L_2(\Omega)}$ for different PC orders $\mathrm{N_o}$. The sample points have also been colored by the sign of the smallest eigenvalue of $\widetilde{[\mathbf{A}]}(\boldsymbol{\eta}_i)$: in blue for a



(a) $\log_{10} \|u - \hat{u}\|_{L_2(\Omega)}$ (b) $\log_{10} \|\widehat{[\mathbf{A}]} - \widetilde{[\mathbf{A}]}\|_F$ (c) $\log_{10} \operatorname{cond}\left(\widetilde{[\mathbf{A}]}\right)$

Figure 7: Log-Histograms of the error norm $\|u - \hat{u}\|_{L2(\Omega)}$ (left), of the approximation error on condensed operator $\|\widehat{[\mathbf{A}]} - \widetilde{[\mathbf{A}]}\|_F$ (center), and of the condition number of the approximate system $\operatorname{cond}\left(\widetilde{[\mathbf{A}]}\right)$ (right) for PC orders $\mathrm{N_o} = 2, 3, 9$. Case of G with $\sigma^2 = 0.5$ and $L = 1$.

positive value and in green for a negative value. Focusing first in the case $\mathrm{N_o} = 2$ reported in Figure 8(a), we observe that the error tends to be correlated with the condition number of the system. In particular, the minimal error increases when $\operatorname{cond}\widetilde{[\mathbf{A}]}$ increases. The case of $\mathrm{N_o} = 3$ in Figure 8(b) appears to have an even more pronounced correlation of the error with the condition number with additional events associated to large condition number and high error level. The color clearly highlights the fact that the highest errors and condition number events are associated with a loss of positivity in $\widetilde{[\mathbf{A}]}$. In fact, the error distribution is somehow bimodal, with one or the other mode depending highly $\widetilde{[\mathbf{A}]}$ having negative eigenvalues. On the contrary, we report no sample with negative eigenvalues in our experiments for $\mathrm{N_o} = 2$ (and also for $\mathrm{N_o} = 4$, 6, and 8; not shown for brevity). Further, increasing the order to $\mathrm{N_o} = 5$ and $\mathrm{N_o} = 9$ in Figure 8(c) and 8(d) we observe the reduction of the probability of loss of positivity events (which is not at all observed in the whole sample set with $\mathrm{N_o} = 9$), and correspondingly a reduction of the resulting solution error. As a closing remark, detection of the loss of positivity in the samples of $\widetilde{[\mathbf{A}]}$ would be a good indicator of an insufficient PC order in the approximation. In our experiments, we found that checking for the positivity of the diagonal elements of $\widetilde{[\mathbf{A}]}(\boldsymbol{\eta}_i)$, a necessary condition for the positivity of the sample, was sufficient for this purpose.

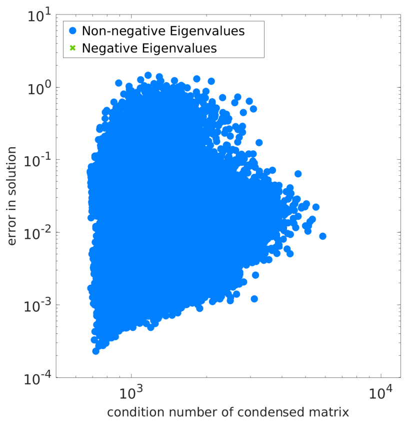
5 Performance Analysis

In Section 5.1 we provide a brief analysis of the computational complexity and memory requirements of the DD-PC method. A few alternative parallel implementations are discussed in Section 5.2, and subsequently compared in Section 5.3.

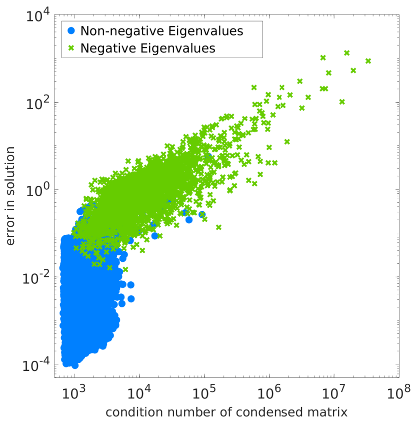
5.1 Complexity analysis

As highlighted in Algorithm 1, the proposed method has two distinct stages: a preprocessing stage during which the PC approximation of the condensed problem is computed, and a sampling stage where approximate samples of the solution are computed.

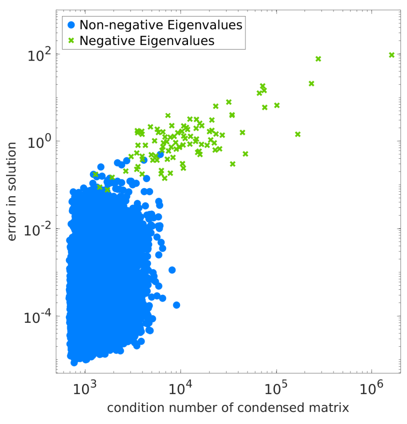
For the first stage, one has to solve on each subdomain a stochastic problem for a set of $\mathrm{N}_\Gamma^{(d)}$ distinct boundary conditions; this discretized stochastic problem has $\mathrm{N}_{\mathrm{in}}^{(d)}$ unknowns expanded on a $\mathrm{P}^{(d)}$ dimensional PC basis. Eventually, the storage of the PC approximation for the subdomain contributions $\widetilde{[\mathbf{A}]}^{(d)}$ and $\tilde{\mathbf{b}}^{(d)}$ has a memory requirement of $\mathrm{N}_\Gamma^{(d)} \times (\mathrm{N}_\Gamma^{(d)} + 1) \times \mathrm{P}^{(d)}$. Clearly, $\mathrm{N}_{\mathrm{in}}^{(d)}$, $\mathrm{N}_\Gamma^{(d)}$ and $\mathrm{P}^{(d)}$ are the parameters driving of the computational complexity of the preprocessing stage on a subdomain, and we illustrate their evolutions when one considers an increasing number D of subdomains to partition a fixed mesh ($\mathrm{N_e} = 163{,}272$) on the previous problem with $L = 0.1$, $\sigma^2 = 0.2$ and $\epsilon_G = 0.01$. Note that the underlying unstructured mesh is essentially isotropic with



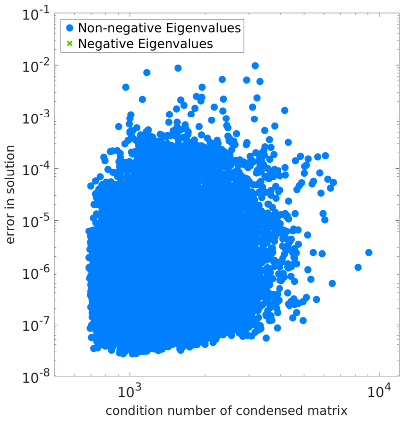
(a) $N_o = 2$



(b) $N_o = 3$



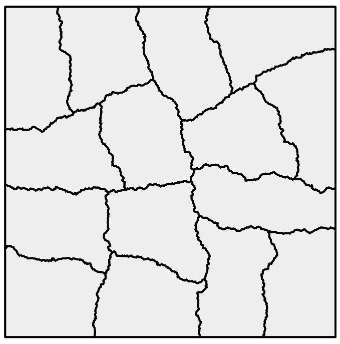
(c) $N_o = 5$



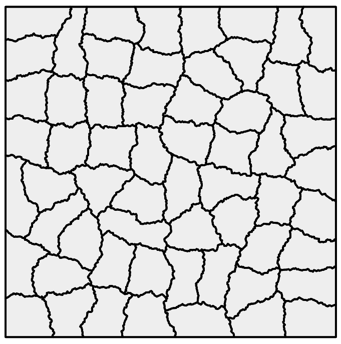
(d) $N_o = 9$

Figure 8: Samples of the error in the solution $\|u - \hat{u}\|_{L2(\Omega)}$ as a function of the condition number cond $[\widetilde{\mathbf{A}}]$. The samples are colored according to the sign of the smallest eigenvalue of $[\widetilde{\mathbf{A}}]$. Different PC orders as indicated.

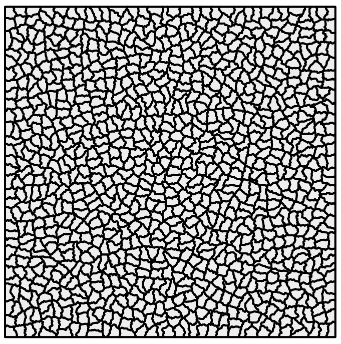
uniform refinement. The Metis software [18] is employed here to partition the domain; several examples are shown in Figure 9.



(a) D = 15.



(b) D = 60.



(c) D = 960.

Figure 9: Partitions of the computational mesh into different numbers of subdomains D as indicated.

The results are reported in Table 1. The second column shows the evolution with D of the condensed problem dimension N_Γ. The third and fourth columns report the corresponding values of $\mathrm{N}_\Gamma^{(d)}$ and $\mathrm{N}_{\mathrm{in}}^{(d)}$ (rounded averages over the set of subdomains, with ± RMS values). It is seen that while $\mathrm{N}_{\mathrm{in}}^{(d)} \sim 1/\mathrm{D}$, the decay of $\mathrm{N}_\Gamma^{(d)}$ is slower denoting the number of interfaces increasing with D (see Figure 9). Similarly, the number of local random variables $\mathrm{N}_\kappa^{(d)}$ decreases at a sub-linear rate with respect to 1/D and would tend asymptotically to 1 for $\mathrm{D} \to \mathrm{N}_\mathrm{e}$ (see the discussion in [10]). The decay behavior of $\mathrm{N}_\kappa^{(d)}$ induces an extremely fast decay rate of the local polynomial basis dimensions $\mathrm{P}^{(d)}$ with D as reported in the last two rows of Table 1, corresponding to PC degrees $\mathrm{N}_\mathrm{o} = 2$ and 6 respectively. For instance, when $\mathrm{N}_\mathrm{o} = 6$ the local PC basis dimension is 10,000 times smaller for D = 480 than for D = 8. However, when D becomes too large, $\mathrm{N}_\kappa^{(d)}$ levels off and so does the dimension of the local PC bases.

D	N_Γ	$\mathrm{N}_\Gamma^{(d)}$	$\mathrm{N}_{\mathrm{in}}^{(d)}$	$\mathrm{N}_\kappa^{(d)}$	$\mathrm{P}^{(d)}$	
					$\mathrm{N}_\mathrm{o} = 2$	$\mathrm{N}_\mathrm{o} = 6$
8	2,233	752 ± 93	$40{,}477 \pm 91$	28.0 ± 0.0	$(4.35 \pm 0.00) \times 10^2$	$(1.34 \pm 0.00) \times 10^6$
15	3,337	549 ± 66	$21{,}514 \pm 62$	17.0 ± 0.0	$(1.71 \pm 0.00) \times 10^2$	$(1.01 \pm 0.00) \times 10^5$
30	5,258	404 ± 48	$10{,}693 \pm 41$	10.7 ± 0.4	$(7.48 \pm 0.53) \times 10^1$	$(1.12 \pm 0.19) \times 10^4$
60	7,582	280 ± 26	$5{,}308 \pm 23$	7.0 ± 0.2	$(3.57 \pm 0.14) \times 10^1$	$(1.70 \pm 0.14) \times 10^3$
120	11,205	201 ± 17	$2{,}624 \pm 14$	5.0 ± 0.0	$(2.10 \pm 0.00) \times 10^1$	$(4.62 \pm 0.00) \times 10^2$
240	15,921	141 ± 11	$1{,}292 \pm 9$	3.2 ± 0.4	$(1.11 \pm 0.21) \times 10^1$	$(1.11 \pm 0.52) \times 10^2$
480	22,726	100 ± 8	632 ± 6	3.0 ± 0.0	$(1.00 \pm 0.00) \times 10^1$	$(8.40 \pm 0.00) \times 10^1$
960	32,618	72 ± 6	306 ± 4	2.8 ± 0.4	$(9.23 \pm 1.58) \times 10^0$	$(7.32 \pm 2.21) \times 10^1$
1920	46,047	51 ± 5	146 ± 3	2.0 ± 0.0	$(6.00 \pm 0.09) \times 10^0$	$(2.80 \pm 0.13) \times 10^1$

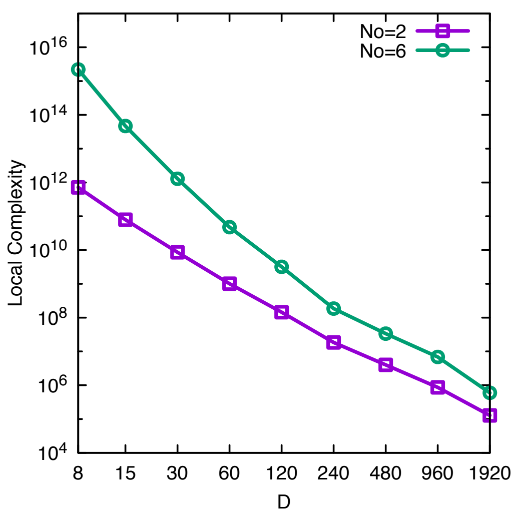
Table 1: Evolutions with the number of subdomains D of the dimension of the condensed problem (N_Γ), (averaged) numbers of local unknowns ($\mathrm{N}_\Gamma^{(d)}$ and $\mathrm{N}_{\mathrm{in}}^{(d)}$), local random variables ($\mathrm{N}_\kappa^{(d)}$) and local PC basis dimension $\mathrm{P}^{(d)}$ for $\mathrm{N}_\mathrm{o} = 2$ and 6.

The results in Table 1 enable us to quantify the reduction in the local stochastic problem complexity and memory requirements to store $\widetilde{[\mathbf{A}]}$ and $\widetilde{\mathbf{b}}$. This is illustrated in Figure 10 which shows the evolution of the local complexity measured by the (averaged) value of $(\mathrm{N}_{\mathrm{in}}^{(d)})^2 \times \mathrm{P}^{(d)}$ is reported for $\mathrm{N}_\mathrm{o} = 2$ and 6 (left plot). We do not report here the consolidated computational complexity, or the sum of local complexities, as the solves at the preprocessing stage are fully independent over the subdomains and can be carried out in parallel. Instead, we remark that in the case of $\mathrm{N}_\mathrm{o} = 6$, small values of D yield too many local variables $\mathrm{N}_\kappa^{(d)}$ with large local PC bases and prohibitive complexities: increasing D makes the local solves tractable. Similarly, increasing D reduces the memory requirements for storing each local contribution to the condensed operator, as depicted in the right plot of Figure 10. The plot shows both the local memory requirement, measured by (averaged) $(\mathrm{N}_\Gamma^{(d)})^2 \times \mathrm{P}^{(d)}$, and the global memory requirement defined as the sum of the local ones. It is seen that the local requirements have essentially the same evolution with D as the complexity. However, the reduction in the global requirement tends to level off as D becomes large, as it could be expected from the behavior of $\mathrm{N}_\Gamma^{(d)}$ and $\mathrm{N}_\kappa^{(d)}$ shown in Table 1.

These findings support the use of the largest possible number of subdomains to reduce the computational complexity and memory requirements of the preprocessing stage. However, we might not want to make D as large as possible because N_Γ, the size of the condensed problem, increases as D increases (see the second column of Table 1). The cost of solving the reduced problem at the sampling stage, therefore, increases as the number of subdomains increases. Thus, as it is typically the case for methods involving domain decomposition, the best value for D will depend on the specific problem at hand and the available computational resources.

5.2 Implementation details

In this section, we discuss choices for the design and implementation of sampling stage algorithms. As described in Section 3.4 and shown in Algorithm 1, the computation of a sample with index i, in the loop starting at line 11 (Algorithm 1), involves four main steps. For a given realization, *i.e.* for one particular index i in these steps can be summarized as follows. First, generate



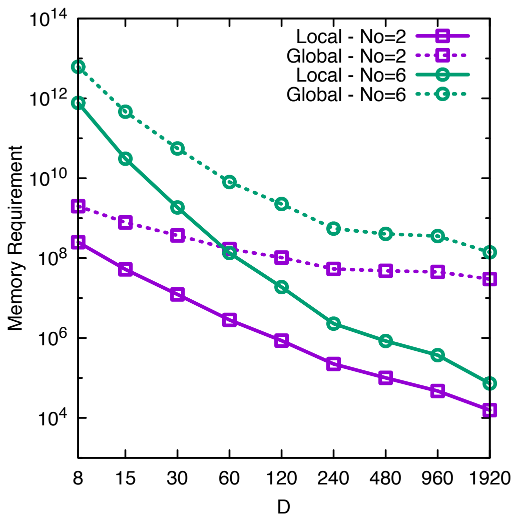


Figure 10: Local complexity (left plot) and local and global memory requirements (right plot), as a function of the number of subdomains D and for two PC degree $N_o = 2$ and $N_o = 6$. Note that both plots use a log-log scale.

a joint random sample of the local random variables (line 12); second, evaluate the subdomain contributions to the condensed problem (40) (line 13), which amounts to evaluating polynomials; third, solve the sample domain decomposition problem (40) (line 16); and finally, if desired, recompute the solution inside selected subdomains (line 17). The parallelization of the first and second steps is trivial, as well as the solution of the local problems in the fourth stage when the boundary data are known; see (42). For the latter step, our PC approach can even bypass the final local solves if the local PC approximations $\tilde{\mathbf{u}}_{\mathrm{in,n}}^{(d)}$ of the elementary solutions $\mathbf{u}_{\mathrm{in,n}}^{(d)}$ can be stored (see Section 3.2.2).

In contrast, different strategies can be envisioned to solve the sampled condensed problem, as further discussed below.

5.2.1 Strategies for solving the condensed problem

Our PC-based sampling approach aims at accelerating direct MC sampling. As discussed earlier in Sections 2.1.1 and 2.2.2, such direct MC methods are usually based on matrix-free iterative solvers where the realizations of the condensed operator and corresponding right-hand side in (28) are never explicitly assembled. Instead, using a CG algorithm, the application of the condensed operator to successive conjugate vectors is implicitly performed in a matrix-free manner by computing residuals from local PDE solutions at the subdomain level. This approach will be referred to as `Dir-loc-CG` and will serve as a reference. It will be compared with different PC-based strategies, also relying on the CG method to solve the condensed problems, and using the same stopping criterion in order to ensure the fairness of the comparisons.

In our PC-based approach, we investigate two main strategies for solving of the condensed problem (40). The first strategy mimics the reference `Dir-loc-CG` above, in that it never assembles the full condensed problem (40). Inside the CG iterations, the subdomain contribution to the residual of the successive conjugate vectors is computed locally, by matrix multiplication with the sample value of $\widetilde{[\mathbf{A}]}^{(d)}$ instead of solving a local PDE problem. This approach will be referred to as `PC-loc-(P)CG`, where the optional `P` indicates whether or not a preconditioner is involved in the CG method (see Section 5.2.3 below). The second strategy, on the contrary, is based on assembling for each sample the corresponding full condensed operator and right-hand side. The condensed problem (40) is still solved using CG, leading to the approach referred to as `PC-glo-(P)CG` in the following. Note that `PC-loc-(P)CG` and `PC-glo-(P)CG` are equivalent, as they solve the same problem, but are expected to have different parallel efficiencies as they will have different communication patterns as discussed in the following.

5.2.2 Parallelism

For `PC-loc-(P)CG`, the realizations are processed sequentially, as in the reference method. For each sample, the solution of the condensed problem is performed in parallel, in a fashion following closely the `Dir-loc-CG` strategy. Specifically, each MPI process is in charge of computing the local contributions to the residual of the set of subdomains handled by the process. We will refer to this strategy as parallelism across subdomains because the workload is distributed among the MPI processes according to the spatial domain decomposition. An overview of this parallel implementation is given in the schematic Algorithm 2.

Algorithm 2: Schematic algorithm illustrating the parallelism across subdomains for strategy `PC-loc-(P)CG`.

```
1  for sample index $i = 1, \dots, M$ do // [SEQUENTIAL LOOP]
2      Generate a random sample of $\boldsymbol{\eta}_i = (\boldsymbol{\eta}_i^{(1)} \dots \boldsymbol{\eta}_i^{(\mathrm{D})})$
3      for subdomain with index $d = 1, \dots, \mathrm{D}$ do // [PARALLEL LOOP]
4          Compute $\widetilde{[\mathbf{A}]}^{(d)}(\boldsymbol{\eta}_i^{(d)})$ and $\tilde{\mathbf{b}}^{(d)}(\boldsymbol{\eta}_i^{(d)})$ using (41)
5      end for
       // PARALLEL solve (except preconditioning)
6      Solve sampled condensed problem (40) for $\mathbf{u}_\Gamma(\theta_i)$ using (local) CG iterations
7      for subdomain with index $d = 1, \dots, \mathrm{D}$ do // [PARALLEL LOOP]
8          Solve local problem (42) for the inner unknowns $\mathbf{u}_{\mathrm{in}}^{(d)}$
9      end for
10 end for
```

Regarding `PC-glo-(P)CG`, a parallelism across samples is more appropriate because the global condensed problem is explicitly assembled. In this strategy, the full condensed operator and right-hand side are assembled in batches of samples, each batch being processed in parallel. For the sake of simplicity, and without loss of generality, we assume that a batch has as many samples as the number of MPI processes, $\mathrm{N}_{\mathrm{MPI}}$. In a given batch, the first and second steps are performed sequentially for the $\mathrm{N}_{\mathrm{MPI}}$ samples, parallelizing the tasks across the subdomains for each sample element of the batch. To each of the $\mathrm{N}_{\mathrm{MPI}}$ samples of the batch corresponds a sample of the condensed problem, which is globally assembled, through collective communications, on its dedicated MPI process. Once all the samples of the batch have been processed this way, each MPI process owns one particular sample of the condensed problem, and can then proceed with its solution. This amounts to a parallelism across samples, in the sense that the current batch of $\mathrm{N}_{\mathrm{MPI}}$ samples has been distributed among the $\mathrm{N}_{\mathrm{MPI}}$ MPI processes and are solved independently. An overview of this parallel implementation is given in the schematic Algorithm 3. Optionally, as for the other strategies, the full solutions (step 4) may be retrieved by final local solves using the solutions of the condensed problem, returning to a parallelization across subdomains.

5.2.3 Preconditioning

One advantage of having an expression of the condensed operator is the possibility to propose a preconditioner for the CG solver. Classical domain decomposition methods can be preconditioned, in particular using two-levels strategies [13, 33]. Here, we rely on an alternative preconditioner based on the condensed operator's expectation, $\mathbb{E}\left[\widehat{[\mathbf{A}]}\right]$, defined as:

$$\overline{[\mathbf{A}]} \doteq \mathbb{E}\left[\widehat{[\mathbf{A}]}\right] \approx \sum_{d=1}^{\mathrm{D}} \mathbb{E}\left[\widetilde{[\mathbf{A}]}^{(d)}\right] = \sum_{d=1}^{\mathrm{D}} \widetilde{[\mathbf{A}]}_{\mathbf{0}}^{(d)}, \tag{52}$$

where $\mathbf{0} \in \mathbb{N}^{\mathrm{N}_\kappa^{(d)}}$ is the multi-index of the constant polynomial. Hereafter, $\overline{[\mathbf{A}]}$ will be referred to as the mean condensed operator. It is expected that $\overline{[\mathbf{A}]}^{-1}\widetilde{[\mathbf{A}]}$ remains close to the identity for all samples so the mean operator can be used as a preconditioner to the full residual iterate appearing in the CG algorithms. In practice, the LU decomposition of $\overline{[\mathbf{A}]}$ is once precomputed

Algorithm 3: Schematic algorithm showing the mixed subdomains and samples parallel processing for the strategy `PC-glo-(P)CG`.

```
1  $i \leftarrow 0$
   // While the desired number of samples has not been reached
2  while $i < M$ do // [SEQUENTIAL LOOP]
       // Start a new batch
3      for process index $\mathrm{p} = 1, \ldots, \mathrm{N_{MPI}}$ do // [SEQUENTIAL LOOP]
4          $i \leftarrow i + 1$
5          Generate a random sample of $\boldsymbol{\eta}_i = (\boldsymbol{\eta}_i^{(1)} \ldots \boldsymbol{\eta}_i^{(\mathrm{D})})$
6          for subdomain with index $d = 1, \ldots, \mathrm{D}$ do // [PARALLEL LOOP]
               // Each process handles $\mathrm{D_p} \approx \mathrm{D}/\mathrm{N_{MPI}}$ subdomains
7              Compute $\widetilde{[\mathbf{A}]}^{(d)}(\boldsymbol{\eta}_i^{(d)})$ and $\tilde{\mathbf{b}}^{(d)}(\boldsymbol{\eta}_i^{(d)})$ using (41)
8          end for
9          Assemble and store the global $\widetilde{[\mathbf{A}]}(\theta_i)$ and $\tilde{\mathbf{b}}(\theta_i)$ on process p
10     end for
       // Each process now owns one global realization of the condensed problem
11     for process index $\mathrm{p} = 1, \ldots, \mathrm{N_{MPI}}$ do // [PARALLEL LOOP]
           // Each process handles 1 realization
12         Solve sampled condensed problem (40) using (global) CG
13     end for
       // Optional
14     for subdomain with index $d = 1, \ldots, \mathrm{D}$ do // [PARALLEL LOOP]
15         Solve local problem (42) for the inner unknowns $\mathbf{u}_{\mathrm{in}}^{(d)}$
16     end for
17 end while
```

prior to the sampling stage, and subsequently used to precondition the CG iterations when solving the condensed problem for different samples. Note that for the (`PC-loc-PCG`) strategy, where the full operator is not assembled, further gain may be obtained by parallelizing the application of the preconditioner, although this direction is not further investigated here.

5.3 Computational behavior

The analysis of the computational behavior of the method is broken down into two parts. First, we investigate the scalability of the preprocessing stage. Second, we discuss the computational behavior of the sampling stage, for the different solving strategies described above, and compare it with the behavior of the classical matrix-free MC sampling approach. Unless specified otherwise, the computations of this section use $\sigma^2 = 0.2$, $L = 0.1$, $\mathrm{D} = 512$ and $\mathrm{N_o} = 2$.

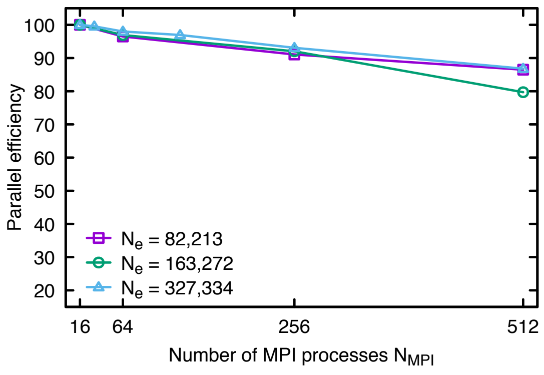
5.3.1 Preprocessing stage

We characterize the scalability with the number $\mathrm{N_{MPI}}$ of MPI processes of the preprocessing stage by the parallel efficiency E, expressed as a percentage:

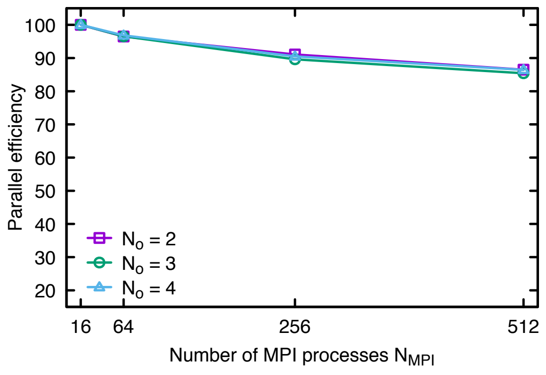
$$E(\mathrm{N_{MPI}}) \doteq 100 \, \frac{T_{\mathrm{ref}}}{\mathrm{N_{MPI}} \, T(\mathrm{N_{MPI}})}, \tag{53}$$

where T_{ref} and $T(\mathrm{N_{MPI}})$ are the measured CPU times of the tasks execution for a reference case and the execution using $\mathrm{N_{MPI}}$ processes. For the reference, we take the smallest number of processes tested, $\mathrm{N_{MPI}} = 16$, and use $T_{\mathrm{ref}} = 16T(16)$, assuming a perfect parallel efficiency from 1 to 16 processes.

Figure 11 shows the parallel efficiency of the preprocessing stage for 3 meshes of increasing size. In Fig. 11(a), we observe that the parallel efficiency slightly decreases with $\mathrm{N_{MPI}}$, but remains above 80% on 512 processes for all three meshes. It shows that the preprocessing stage is scaling decently, even using a naive, static, a priori load balancing strategy. The moderate loss of efficiency can be explained by processes waiting for each other to get to a certain point, caused by load imbalance. Although the preprocessing stage involves no communication (either point-to-point or collective) between processes, the preprocessing stage ends when *all* processes have terminated leading to a worst case idle-time scenario. In addition, each process p handles a certain number of subdomains whose indices d are collected in $\mathcal{I}_{\mathrm{p}}$. Note that in the present tests we used numbers of processes such that $\mathrm{D} = 512$ is always a multiple of $\mathrm{N}_{\mathrm{MPI}}$, and that the processes handle exactly the same number of subdomains, namely $\mathrm{D}/\mathrm{N}_{\mathrm{MPI}}$. However, the subdomains support FE meshes having different sizes, as well as possibly different numbers of local random variables $\mathrm{N}_{\kappa}^{(d)}$, see Table 1. As a consequence, the number and size of the local Galerkin problems that a process p has to solve may change slightly from one process to another. As $\mathrm{N}_{\mathrm{MPI}}$ increases, fewer subdomains are handled by a process, down to the case of 512 processes each handling one single subdomain, tending to increase the load imbalance between processes with a degradation of the parallel efficiency. It is clear than more advanced partitioning and load balancing techniques can be employed to improve the scaling properties of this stage. As a side note, we point out that using carefully designed regular structured meshes should theoretically lead to quasi-ideal scaling. Finally, we observe in Fig. 11(b) that the parallel efficiency is not affected by the PC degree N_{o}.



(a) $\mathrm{N}_{\mathrm{o}} = 2$.



(b) $\mathrm{N}_{\mathrm{e}} = 82{,}213$.

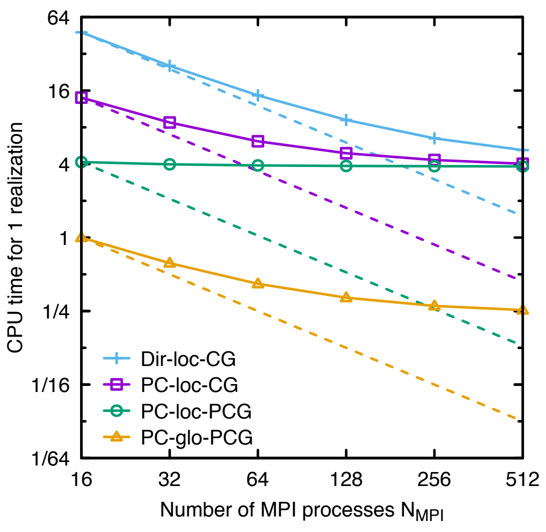
Figure 11: Parallel efficiency $E(\mathrm{N}_{\mathrm{MPI}})$ of the preprocessing stage, see (53), for different meshes (Fig. 11(a)) and different PC degrees (Fig. 11(b)).

5.3.2 Sampling stage

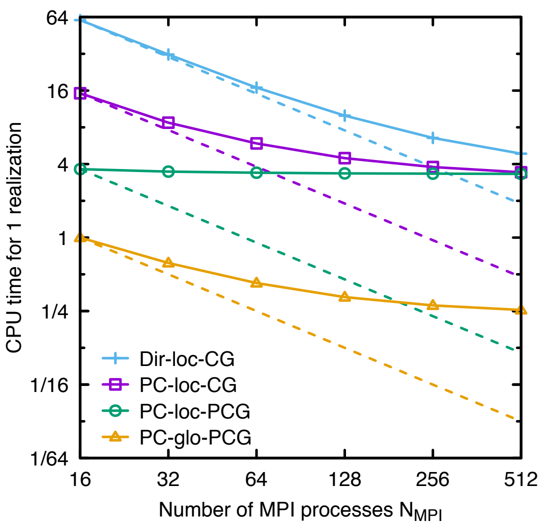
We now investigate the computational behavior of the sampling stage. In particular, we compare the different strategies discussed in Section 5.2.

Figure 12 reports the CPU times needed to generate a single sample of the condensed problem solution, as a function of $\mathrm{N}_{\mathrm{MPI}}$. These measurements only include the first three steps of the sampling procedure, leaving aside the final calculation of the full solution over the subdomains. Moreover, for `PC-glo-PCG` and a parallelization over samples, we consider a batch of size $M = \mathrm{D} = 512$ and report the average computational time (divided by M) for a fair comparison. In addition, the CPU times are scaled so that the reported time using `PC-glo-PCG` on 16 processes equals 1.

In Fig. 12(a), corresponding to a spatial mesh with $\mathrm{N}_{\mathrm{e}} = 163{,}272$ quadratic finite elements, we observe that the strategy `PC-loc-CG` outperforms the reference approach `Dir-loc-CG` with an acceleration factor of about 3.5 on 16 processes. As the number of processes increases, the two approaches lose parallel efficiency and seem to converge to the same CPU time. This trend can be explained by the collective data communication which needs to be performed at each CG iteration. This communication time does not decrease as $\mathrm{N}_{\mathrm{MPI}}$ increases, while on the contrary, the workload of the processes for solving local problems (in `Dir-loc-CG`) or performing local matrix-vector products (in `PC-loc-CG`) decreases, due to a good parallel scaling of these computations. Eventually, the communication cost becomes comparable to the computational



(a) $N_e = 163{,}272$.



(b) $N_e = 327{,}334$.

Figure 12: Scaled CPU times, to generate one sample, as a function of the number of MPI processes N_{MPI}. The dashed lines represent ideal parallel scaling.

cost of the rest of the CG algorithm (*e.g.* dot products), which scale poorly, and consequently the overall sampling cost converges to this flat cost. Concerning `PC-loc-PCG`, the effect of the mean preconditioner can be appreciated comparing its computational time with the `PC-loc-CG` strategy: for 16 processes, the CPU time is reduced by another factor of about 3.5. This reduction is due to the improved convergence of the iterative solver, allowing to save many CG iterations. However, the overall cost of `PC-loc-PCG` is quickly dominated by the application of the preconditioner, which is not performed in parallel in the present implementation (see Section 5.2.3), with a very poor parallel scaling of `PC-loc-PCG` as a result. Eventually, the savings of the preconditioner are lost and the overall CPU time converges to that of the non-preconditioned version.

Finally, the strategy `PC-glo-PCG`, based on the full assembly of the global condensed system (40) and using the mean operator as a preconditioner, has a parallel efficiency behavior similar to that of `PC-loc-CG`, but with a computational cost up to 16 times less. `PC-glo-PCG` outperforms the reference strategy `Dir-loc-CG` by a factor of about 48 for $N_{MPI} = 16$ and remains asymptotically 20 times faster despite its efficiency drop. Again, the drop in efficiency for `PC-glo-PCG` is caused by the collective communication needed to assemble the global condensed system from its local contributions. For the present example, this communication step involves the exchange of about 4.5 million double precision values between all the N_{MPI} processes. In addition to being much more efficient than the other strategies, this last approach lends itself to a task-based parallel framework, where data locality would be preserved and collective communication would be avoided. Although outside the scope of this paper, it is important to point out that adopting such a parallel processing paradigm could potentially improve significantly the parallel scaling of this approach. In any case, having a different treatment of the PC evaluation (parallelized across subdomains) and of the condensed system solve (parallelized across samples) clearly allows for more flexibility.

To conclude this analysis, let us mention that the reported trends in the parallel efficiency for the different methods does not significantly depend on the FE discretization of the problem. This can be seen comparing the similar evolution of the CPU times in Fig. 12(a) and Fig. 12(b), the latter corresponding to an FE mesh with twice as many elements as before $N_e = 327{,}334$ elements (leading to computational times roughly twice as long). We also note that for this refined mesh and 16 processes, `PC-glo-PCG` is roughly 64 times faster than for the reference `Dir-loc-CG`.

6 Conclusions

We have presented an acceleration strategy for a Monte Carlo sampling-based Stochastic Elliptic PDE solver. The method employs a domain decomposition technique to partition the computational domain into smaller non-overlapping subdomains. In a first stage, an approximation of the local boundary-to-residual map is constructed, independently over each subdomain. This approximation uses a PC expansion to represent the dependencies of the map on the stochastic coefficient of the elliptic equation. The cost of computing this local PC approximation is reduced owing to the possibly low dimensional representation of the stochastic coefficient over the considered subdomain, compared to its global representation. These local PC expansions can be combined together to obtain an approximation of the (global) condensed problem relating the stochastic solution at the interface of the subdomains. The local PC-based representations of the condensed problem can be sampled with a low computational cost, amounting to simple polynomial evaluations. This feature is exploited in a second stage to generate, at a reduced computational cost, realizations of the stochastic solution via MC sampling.

We validated the accuracy of the proposed approach on a numerical example that also served to analyze convergence with the polynomial degree of the PC expansion. An important finding is that, as desired, the domain decomposition allows for significant computational time saving while having a negligible effect on the approximation error which is essentially driven by the polynomial degree of the expansion. We also analyzed the performance of different parallel implementations of the approach. Specifically, we showed that the cost of the preprocessing stage can be conveniently distributed over multiple processors with a close to ideal parallel efficiency (higher than 80% in our experiments). Given that we used a naive, static, a priori load balancing strategy, the scaling properties of the first stage could even be improved by employing more advanced partitioning and load balancing techniques. Concerning the sampling stage, all the parallel strategies involving the PC approximation of the condensed problem perform better than the reference approach. A noticeable degradation in the parallel efficiency is however reported when the number of MPI processes is increased. Despite this efficiency drop, the best sampling strategy is found to remain at least 20 times faster than the reference for the largest number of processes tested (512) when it is up to 60 times faster when only 16 MPI processes are used. The collective communications involved in the assembly of the condensed problem, from the local contributions, are responsible for the efficiency drop. A possible way to mitigate this issue would be to rely on a task-based parallel framework, where data locality would be preserved and collective communication would be avoided.

In addition to improving parallel efficiency, future works should focus on improved partitioning strategies and the determination of the optimal number of subdomains yielding the lowest computational cost. The latter aspect is non-obvious, but involves several trade-offs between different steps of the method, and clearly depends on the computational architecture and the resources available. Another potential route to develop the proposed approach is exploring the potential interest of considering a hierarchy of FE meshes. This hierarchy could be used to accelerate the resolution of the condensed problem (as in two-level domain decomposition methods [13, 33, 26]), on the one hand, and to optimize the computational complexity of the MC method (as in multilevel MC methods [8, 2]), on the other hand.

Acknowledgments

This work was supported by the US Department of Energy (DOE), Office of Science, Office of Advanced Scientific Computing Research, under Award Number DE-SC0010540. Sandia National Laboratories is a multimission laboratory managed and operated by National Technology and Engineering Solutions of Sandia, LLC., a wholly owned subsidiary of Honeywell International, Inc., for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-NA-0003525. Support of the King Abdullah University of Science and Technology is also acknowledged. The parallel experiments were carried out on the IBM Blue Gene/Q machine of IDRIS (CNRS, Orsay), and the authors would like to thank J. Chergui (LIMSI-CNRS, Orsay) for helpful discussions and his valuable assistance in carrying out the parallel experiments.

References

[1] 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(3):1005–1034, 2007.

[2] A. Barth, C. Schwab, and N. Zollinger. Multi-level monte carlo finite element method for elliptic pdes with stochastic coefficients. *Numerische Mathematik*, 119(1):123–161, 2011.

[3] J. Beck, F. Nobile, L. Tamellini, and R. Tempone. Convergence of quasi-optimal stochastic galerkin methods for a class of {PDES} with random coefficients. *Computers & Mathematics with Applications*, 67(4):732 – 751, 2014. High-order Finite Element Approximation for Partial Differential Equations.

[4] J. Beck, R. Tempone, F. Nobile, and L. Tamellini. On the optimal polynomial approximation of stochastic pdes by galerkin and collocation methods. *Mathematical Models and Methods in Applied Sciences*, 22(09):1250023, 2012.

[5] R. E. Caflisch. Monte carlo and quasi-monte carlo methods. *Acta numerica*, 7:1–49, 1998.

[6] Y. Chen, J. Jakeman, C. Gittelson, and D. Xiu. Local polynomial chaos expansion for linear differential equations with high dimensional random inputs. *SIAM Journal on Scientific Computing*, 37(1):A79–A102, 2015.

[7] A. Chkifa, A. Cohen, and C. Schwab. High-dimensional adaptive sparse polynomial interpolation and applications to parametric pdes. *Foundations of Computational Mathematics*, 14(4):601–633, 2014.

[8] K. A. Cliffe, M. B. Giles, R. Scheichl, and A. L. Teckentrup. Multilevel monte carlo methods and applications to elliptic pdes with random coefficients. *Computing and Visualization in Science*, 14(1):3, 2011.

[9] A. Cohen, R. DeVore, and C. Schwab. Convergence rates of best n-term galerkin approximations for a class of elliptic spdes. *Foundations of Computational Mathematics*, 10(6):615–646, 2010.

[10] A. A. Contreras, P. Mycek, O. P. Le Maître, F. Rizzi, B. Debusschere, and O. M. Knio. Parallel domain decomposition strategies for stochastic elliptic equations – part a: Local kl representations. *Under review*, 2016.

[11] M. K. Deb, I. M. Babuška, and J. T. Oden. Solution of stochastic partial differential equations using galerkin finite element techniques. *Computer Methods in Applied Mechanics and Engineering*, 190(48):6359–6372, 2001.

[12] R. T. F. Nobile and C. Webster. A sparse grid stochastic collocation method for partial differential equations with random input data. *SIAM J. Numer. Anal.*, 46(5):2309–2345, 2008.

[13] C. Farhat, A. Macedo, and M. Lesoinne. A two-level domain decomposition method for the iterative solution of high frequency exterior helmholtz problems. *Numerische Mathematik*, 85(2):283–308, 2000.

[14] P. Frauenfelder, C. Schwab, and R. A. Todor. Finite elements for elliptic problems with stochastic coefficients. *Computer methods in applied mechanics and engineering*, 194(2):205–228, 2005.

[15] R. G. Ghanem and P. D. Spanos. *Stochastic finite elements: a spectral approach.* Courier Corporation, 2003.

[16] M. Kac and A. Siegert. An explicit representation of a stationary gaussian process. *Ann. Math. Stat.*, 18:438–442, 1947.

[17] K. Karhunen. Uber lineare methoden in der wahrscheinlichkeitsrechnung. *Amer. Acad. Sci., Fennicade, Ser. A, I*, 37:3–79, 1947.

[18] G. Karypis and V. Kumar. A fast and high quality multilevel scheme for partitioning irregular graphs. *SIAM Journal on scientific Computing*, 20(1):359–392, 1998.

[19] O. Knio and O. Le Maître. Uncertainty propagation in CFD using polynomial chaos decomposition. *Fluid Dyn. Res.*, 38:616–640, 2006.

[20] V. G. Korneev and U. Langer. *Dirichlet–Dirichlet Domain Decomposition Methods for Elliptic Problems: h and hp Finite Element Discretizations.* World Scientific, 2015.

[21] O. P. Le Maître and O. M. Knio. *Spectral methods for uncertainty quantification: with applications to computational fluid dynamics.* Scientific Computation. Springer, 2010.

[22] O. Le Maître, O. Knio, B. Debusschere, H. Najm, and R. Ghanem. A multigrid solver for two-dimensional stochastic diffusion equations. *Computer Methods in Applied Mechanics and Engineering*, 192(41–42):4723 – 4744, 2003.

[23] O. P. Le Maître, M. T. Reagan, H. N. Najm, R. G. Ghanem, and O. M. Knio. A stochastic projection method for fluid flow: Ii. random process. *Journal of Computational Physics*, 181(1):9 – 44, 2002.

[24] M. Loève. Fonctions aléatoires du second ordre. In P. Lévy, editor, *Processus Stochastique et mouvement Brownien.* Gauthier Villars, 1948.

[25] H. Najm. Uncertainty quantification and polynomial chaos techniques in computational fluid dynamics. *Ann. Rev. Fluid Mech.*, 41:35–52, 2009.

[26] F. Nataf, H. Xiang, and V. Dolean. A two level domain decomposition preconditioner based on local dirichlet-to-neumann maps. *Comptes Rendus Mathematique*, 348(21):1163 – 1167, 2010.

[27] A. Nouy. A generalized spectral decomposition technique to solve a class of linear stochastic partial differential equations. *Comput. Methods Appl. Mech. Engrg.*, 196(45-48):4521–4537, 2007.

[28] A. Nouy. Generalized spectral decomposition method for solving stochastic finite element equations: Invariant subspace problem and dedicated algorithms. *Computer Methods in Applied Mechanics and Engineering*, 197(51–52):4718 – 4736, 2008.

[29] A. Nouy and O. Le Maître. Generalized spectral decomposition for stochastic nonlinear problems. *J. Comput. Phys.*, 228:202–235, 2009.

[30] M. Papadrakakis and V. Papadopoulos. Robust and efficient methods for stochastic finite element analysis using monte carlo simulation. *Computer Methods in Applied Mechanics and Engineering*, 134(3-4):325–340, 1996.

[31] S. Pranesh and D. Ghosh. Addressing the curse of dimensionality in ssfem using the dependence of eigenvalues in kl expansion on domain size. *Computer Methods in Applied Mechanics and Engineering*, 311:457–475, 2016.

[32] A. Quarteroni and A. Valli. *Domain Decomposition Methods for Partial Differential Equations.* Numerical mathematics and scientific computation. Clarendon Press, 1999.

[33] B. Smith, P. Bjorstad, and W. Gropp. *Domain Decomposition: parallel multilevel methods for elliptic partial differential equations.* Cambridge University Press, 2004.

[34] B. F. Smith. Domain decomposition methods for partial differential equations. In *Proceedings of ICASE/LaRC Workshop on Parallel Numerical Algorithms.* Univ. Press, 1995.

[35] L. Tamellini, O. Le Maître, and A.Nouy. Model reduction based on proper generalized decomposition for stochastic steady incompressible navier stokes equations. *SIAM J. Scientific Computing*, 36(3):1089–1117, 2014.

[36] A. Toselli and O. Widlund. *Domain Decomposition Methods - Algorithms and Theory*. Springer Series in Computational Mathematics. Springer, 2005.

[37] D. Xiu and G. E. Karniadakis. The wiener–askey polynomial chaos for stochastic differential equations. *SIAM journal on scientific computing*, 24(2):619–644, 2002.