A stochastic finite element scheme for solving partial differential equations defined on random domains

Zhibao Zheng[a,d,*], Marcos Valdebenito[c], Michael Beer[b,e,f], Udo Nackenhorst[a,d]

[a]*Institute of Mechanics and Computational Mechanics, Leibniz Universität Hannover, Appelstraße 9a, 30167, Hannover, Germany*
[b]*Institute for Risk and Reliability, Leibniz Universität Hannover, Callinstraße 34, 30167 Hannover, Germany*
[c]*Chair for Reliability Engineering, TU Dortmund University, Leonhard-Euler-Straße 5, 44227 Dortmund, Germany*
[d]*International Research Training Group 2657, Leibniz Universität Hannover, Appelstraße 11/11a, 30167, Hannover, Germany*
[e]*Institute for Risk and Uncertainty and School of Engineering, University of Liverpool, Peach Street, Liverpool L69 7ZF, UK*
[f]*International Joint Research Center for Resilient Infrastructure & International Joint Research Center for Engineering Reliability and Stochastic Mechanics, Tongji University, 1239 Siping Road, Shanghai 200092, China*

Abstract

This paper proposes a novel stochastic finite element scheme to solve partial differential equations defined on random domains. A geometric mapping algorithm first transforms the random domain into a reference domain. By combining the mesh topology (i.e. the node numbering and the element numbering) of the reference domain and random nodal coordinates of the random domain, random meshes of the original problem are obtained by only one mesh of the reference domain. In this way, the original problem is still discretized and solved on the random domain instead of the reference domain. A random isoparametric mapping of random meshes is then proposed to generate the stochastic finite element equation of the original problem. We adopt a weak-intrusive method to solve the obtained stochastic finite element equation. In this method, the unknown stochastic solution is decoupled into a sum of the products of random variables and deterministic vectors. Deterministic vectors are computed by solving deterministic finite element equations, and corresponding random variables are solved by a proposed sampling method. The computational effort of the proposed method does not increase dramatically as the stochastic dimension increases and it can solve high-dimensional stochastic problems with low computational effort, thus the proposed method avoids the curse of dimensionality successfully. Four numerical examples are given to demonstrate the good performance of the proposed method.

Keywords: Random domains; Mesh transformation; Random isoparametric mapping; Stochastic finite element method; Random interfaces;

1. Introduction

As predicting uncertainty propagation of physical models has become an important part of the analysis of many engineering problems, it is necessary to develop efficient computational schemes for dealing with a variety of uncertainties, e.g. random material properties, random external forces, random geometries and their couplings [1, 2]. The focus of this work is on the geometric uncertainty, which may arise from many kinds of problems, e.g. topology optimization, random multi-phased materials, etc [3, 4, 5, 6, 7].

In this paper, we pay attention on solving deterministic/stochastic partial differential equations (PDEs) defined on random/parameter-dependent domains. In the last decades, several methods have been proposed for this purpose. Following the approaches to deal with random geometries, they can be divided into the remeshing method [8], the geometric mapping method [9], the fictitious method [10, 11] and the extended stochastic finite element method [5, 6], etc. A straightforward method to deal with random domains is the Monte Carlo simulation (MCS) [12] and other non-intrusive techniques, e.g. multilevel MCS, response surface method, stochastic collocation method, etc [8, 13, 14]. This kind of method remeshes the geometry associated with each random sample and classical finite element solvers are then used to solve the solution on each sampled mesh. It is easy to implement and existing codes can be adopted without difficulties. The method is independent or weakly dependent on stochastic dimensions, thus it can be applied to the geometries related to a large number of random variables. However, numerous deterministic simulations are necessary in order to obtain a high-precision stochastic solution, which leads to expensive computational costs. Also, since the samples of the stochastic solution are obtained on different domains and meshes, extra attention is required for the postprocessing of the stochastic solution. In order to avoid remeshing the domain for each sample realization, the geometric mapping

*Corresponding author
Email address: zhibao.zheng@ibnm.uni-hannover.de (Zhibao Zheng)

method is proposed in [9, 15, 16]. The method maps the random domain into a reference domain and then transforms the PDE defined on the random domain into a stochastic PDE defined on the reference domain. In this method, a boundary-conforming coordinate system is used to map the random geometry, which can be considered as a stochastic extension of deterministic mesh mapping methods [17]. For the obtained stochastic PDE, classical stochastic finite element solvers are used to solve the stochastic finite element equation (SFEE), e.g. MCS, stochastic collocation method, spectral stochastic finite element method, etc [9, 18, 19, 20]. However, extra complexities are introduced when tackling nonlinear PDEs since the differential operator is coupled with the geometric mapping.

The fictitious domain method, as a stochastic extension of the deterministic fictitious domain method [21], is proposed to solve PDEs defined on random domains [10, 11]. The fictitious domain is a deterministic domain including all possible sample realizations of the random domain and it usually has simple geometry. This kind of method can be applied to complex geometries, but effective methods are required to tackle the random boundaries of the random domain in the fictitious domain. In [10], the original PDE is transformed into a saddle-point problem defined on the fictitious domain and the original boundary conditions are enforced by Lagrange multipliers. In [11], the original PDE is reformulated on the fictitious domain. The proper generalized decomposition (PGD) method is used to approximate the stochastic solution and a stochastic indicator function is proposed to capture the random domain in the fictitious domain. Furthermore, the extended stochastic finite element method is proposed in [5, 6, 22, 23]. This method still reformulates the PDE on a fictitious domain and extends the deterministic extended finite element method [24] to handle the discontinuities of the stochastic solution on the fictitious domain. The spectral stochastic finite element method [18, 25] is then used to solve the solution on the mesh of the fictitious domain. Other methods have also been developed, e.g. the computational frameworks and error estimations for Neumann and Dirichlet boundary value problems defined on random domains [26, 27], the perturbation-based methods for problems with small geometric variability [28, 29, 30] and the PGD-based descriptions for geometric parameters [31, 32].

In this paper, we develop an efficient numerical scheme for solving linear deterministic/stochastic partial differential equations (PDEs) defined on random/parameter-dependent domains. To avoid

remeshing the random domain, a geometric mapping algorithm is adopted to transform the random domain into a reference domain. As the unknown stochastic solution of the transformation equation, random nodal coordinates of the random domain are obtained by solving a Laplace equation with random boundary conditions. The random mesh of the random domain can be generated by combining the mesh topology (i.e. the node numbering and the element numbering) of the reference domain and random nodal coordinates. A random isoparametric mapping of the random mesh is developed to assemble the stochastic stiffness matrix and the stochastic force vector. The assembly procedure is the same as the classical FEM assembly and is completely non-intrusive. Existing FEM codes can be embedded without any modification.

We mention that a similar random mesh approach is also used in [16] as a discretized version of the method proposed in [9, 15] and conditions are given to ensure the well-posedness of the geometric transformation. However, the method relies on the polynomial chaos approximation of the random coordinates and still assembles stochastic matrices and vectors in the reference domain, which leads to the coupling of the differential operator of the PDE and the geometric uncertainty. Much attention is needed to tackle the coupling when nonlinear PDEs defined on random domains are solved. Our proposed method still discretizes and solves the original PDE on the random domain instead of the reference domain. Hence it decouples the differential operator of the PDE and the geometric uncertainty, which is consistent with the fact that the differential operators of PDEs are not sensitive to geometric domains when using the finite element method. We also mention that a random isoparametric mapping is used in [28], which embeds the random isoparametric mapping of the random boundary into the spectral SFEM frame. This method relies on the polynomial chaos-based approximations of random boundaries, mapping of differential operators and corresponding Jacobian matrices, which is considered as a kind of intrusive approach. Additional computational complexity is thus required. Also, high-dimensional stochastic problems remain challenging for this method. The proposed random isoparametric mapping in this paper has the same computational complexity as the deterministic isoparametric mapping and it is easy to implement by existing FEM assembly codes.

After assembling stochastic matrices and vectors, we adopt a weak-intrusive method to solve the obtained stochastic finite element equation. In this method, the unknown stochastic solution

is decoupled into a sum of the products of a set of random variables and deterministic vectors. Deterministic vectors are obtained by solving deterministic finite element equations that are generated by applying the stochastic Galerkin method to the original SFEE. Corresponding random variables are the solutions of one-dimensional stochastic algebraic equations that are solved efficiently by use of a proposed sampling method. Since the computational effort of the proposed method does not increase dramatically as the stochastic dimension increases and it can solve high-dimensional stochastic problems with low computational effort, the proposed method avoids the curse of dimensionality successfully.

The paper is organized as follows: Section 2 gives the basic setting of deterministic/stochastic PDEs on random domains and a geometric transformation from the random domain to a reference domain is presented. Section 3 introduces a random isoparametric mapping method to assemble SFEE. A weak-intrusive SFEM is then used to solve the obtained SFEE in Section 4. The algorithm implementation of the proposed method is elaborated in Section 5. Following that, four numerical examples, including an elastic equation defined on a random domain, a stochastic elliptic PDE with a random interface and a case from orthodontics with random material properties and random geometry, are given to demonstrate the performance of the proposed method in Section 6, and conclusions and discussions follow in Section 7.

2. Geometric transformation of the random domain

2.1. PDEs defined on random domains

Let $(\Theta, \Xi, \mathcal{P})$ be a suitable probability space , where Θ denotes the space of elementary events, Ξ is a σ-algebra defined on Θ and $\mathcal{P}$ is a probability measure. In this paper, we consider the deterministic/stochastic linear PDE defined on a random domain $\mathcal{D}(\theta) \subset \mathbb{R}^d$ with the physical dimension $d = 1, 2, 3$ and the random boundary $\partial\mathcal{D}(\theta)$. We solve the following stochastic problem: find a stochastic solution $\mathbf{u}(\mathbf{x}, \theta) : \mathcal{D}(\theta) \to \mathbb{R}$ such that the following PDE holds for all $\theta \in \Theta$,

$$\mathcal{L}(\mathbf{u}(\mathbf{x}, \theta), \mathbf{x}) = 0 \quad \text{in} \quad \mathcal{D}(\theta), \tag{1}$$

subjected to boundary conditions on $\partial\mathcal{D}(\theta)$, where $\mathcal{L}(\cdot)$ is a linear differential operator, $\mathbf{x} = (x_1, \cdots, x_d) \in \mathbb{R}^d$ is the Cartesian coordinate. We assume that the random boundary $\partial\mathcal{D}(\theta)$ is

sufficiently regular and that Eq. (1) is well-posed. The randomness of the domain $\mathcal{D}(\theta)$ is only described by the random boundary $\partial\mathcal{D}(\theta)$ that is related to random variables/fields (the parameters for describing unfixed domains can also be considered as special random variables). Although we only consider linear PDEs in this paper, nonlinear PDEs defined on random domains are readily solved in the computational frame proposed in this paper.

2.2. Geometric transformation from random domain to reference domain

To avoid remeshing the random domain $\mathcal{D}\left(\theta^{(i)}\right)$ for each sample $\theta^{(i)}$, $i = 1, \cdots, n_s$, where n_s is the number of random samples, we transform the random domain into a reference domain and represent the random domain by the mesh topology of the reference domain. To transform the random domain $\mathcal{D}(\theta) \subset \mathbb{R}^d$ into a reference domain $\overline{\mathcal{D}} \subset \mathbb{R}^d$, we consider that the random coordinate $\mathbf{x}(\theta) = (x_1(\theta), \cdots, x_d(\theta)) \in \mathcal{D}(\theta)$ is mapped into the deterministic coordinate $\overline{\mathbf{x}} = (\overline{x}_1, \cdots, \overline{x}_d) \in \overline{\mathcal{D}}$ by the following transformation,

$$\overline{\mathbf{x}} = \mathscr{M}^{-1}(\mathbf{x}(\theta), \theta), \tag{2}$$

and the random coordinate $\mathbf{x}(\theta)$ is represented by the deterministic coordinate $\overline{\mathbf{x}}$ and the inverse mapping of Eq. (2),

$$\mathbf{x}(\theta) = \mathscr{M}(\overline{\mathbf{x}}, \theta), \tag{3}$$

where $\mathscr{M}(\cdot, \theta)$ represents the mapping operator and $\mathscr{M}^{-1}(\cdot, \theta)$ is its inverse operator. In this way, we can use the deterministic coordinate of the reference domain to represent the random coordinate of the random domain. In other words, the random domain is fully described by the reference domain and the realization θ. The selection of $\mathscr{M}(\cdot, \theta)$ is not unique and it is dependent on the selection of the reference domain and the uncertainty involved in the problem under consideration. We can adopt an arbitrary reference domain as long as the ease of implementation and well-posedness of the transformation can be ensured [9, 17]. In practice, the mean of the random domain is usually chosen as the reference domain.

2.3. Geometric mapping based on the Laplace equation

We execute the transformation Eq. (3) on the mesh of the reference domain. Correspondingly, the random mesh of the random domain is obtained by the mesh transformation of the reference

domain, which can be considered a mesh-based random geometry description. Several differential system-based mapping methods are initially developed for the mesh transformation, e.g. the elliptic equations, hyperbolic equations, etc [17], which are then extended to deal with random geometries in [9, 15]. In this paper we adopt the Laplace equation for the transformation $\mathscr{M}(\cdot,\theta)$ in Eq. (3), which corresponds to

$$\Delta x_i(\theta) = 0 \quad \text{in} \quad \overline{\mathcal{D}},\ i = 1,\cdots,d \tag{4}$$

with the geometric boundary constraints

$$x_i(\theta)|_{\Gamma_j} = \mathscr{M}_{i,j}\left(\overline{\mathbf{x}}\big|_{\Gamma_j},\theta\right),\ j = 1,\cdots,b, \tag{5}$$

where $\Delta = \sum_{j=1}^{d} \frac{\partial^2}{\partial \overline{x}_j^2}$ represents the Laplace operator, $\overline{\mathbf{x}}\big|_{\Gamma_j} = \left[\overline{x}_1|_{\Gamma_j},\cdots,\overline{x}_d|_{\Gamma_j}\right] \in \mathbb{R}^d$ and $\mathbf{x}(\theta)|_{\Gamma_j} = \left[x_1(\theta)|_{\Gamma_j},\cdots,x_d(\theta)|_{\Gamma_j}\right] \in \mathbb{R}^d$ represent the deterministic and random coordinates on the boundary Γ_j, $\mathscr{M}_{i,j}(\cdot,\theta)$ represents the mapping of the i-th coordinate on the boundary Γ_j, b is the total number of geometry boundaries under consideration. In this paper, we assume that the obtained random mesh has a good quality and can be used to solve the PDE, e.g., there should be no element flips in the random mesh. Conditions to ensure the well-posedness of the transformation can be found in [16, 17]. Further, several advanced mesh deformation methods [33, 34] can be used for more general cases that the Laplace equation-based mesh transformation does not work well, e.g., the case of too large mesh deformations caused by large uncertainties, which is beyond the scope of this paper and will be presented in follow-up studies.

2.4. Finite element solution of the transformation equation

To solve Eq. (4), we discretize the reference domain by use of the finite element method [35]. Let $\overline{\mathbf{X}}_i \in \mathbb{R}^n$, $i = 1,\cdots,d$ be the discretized nodal coordinates of the coordinate $\overline{x}_i$ obtained by the finite element discretization, where n is the number of nodes, and $\mathbf{X}_i(\theta) \in \mathbb{R}^n$ be the corresponding stochastic solution on the discretized nodes, which is also the discretization of the random coordinate $x_i(\theta)$. The finite element equation of Eq. (4) is thus given by

$$\begin{cases} \mathbf{K}_{\mathcal{G}}\mathbf{X}_i(\theta) = 0 \\ \mathbf{X}_i(\theta)|_{\Gamma_j} = \mathscr{M}_{i,j}\left(\overline{\mathbf{X}}\big|_{\Gamma_j},\theta\right),\ j = 1,\cdots,b \end{cases},\ i = 1,\cdots d, \tag{6}$$

which can be rewritten as

$$\begin{bmatrix} \widetilde{\mathbf{K}}_{\mathcal{G},0,0} & \widetilde{\mathbf{K}}_{\mathcal{G},0,1} & \cdots & \widetilde{\mathbf{K}}_{\mathcal{G},0,b} \\ \widetilde{\mathbf{K}}_{\mathcal{G},1,0} & \widetilde{\mathbf{K}}_{\mathcal{G},1,1} & \cdots & \widetilde{\mathbf{K}}_{\mathcal{G},1,b} \\ \vdots & \vdots & \ddots & \vdots \\ \widetilde{\mathbf{K}}_{\mathcal{G},b,0} & \widetilde{\mathbf{K}}_{\mathcal{G},b,1} & \cdots & \widetilde{\mathbf{K}}_{\mathcal{G},b,b} \end{bmatrix} \begin{bmatrix} \mathbf{X}_i(\theta)|_{\mathcal{D}(\theta)} \\ \mathscr{M}_{i,1}\left(\overline{\mathbf{X}}|_{\Gamma_1}, \theta\right) \\ \vdots \\ \mathscr{M}_{i,b}\left(\overline{\mathbf{X}}|_{\Gamma_b}, \theta\right) \end{bmatrix} = 0,\ i = 1, \cdots d, \tag{7}$$

where $\mathbf{X}_i(\theta)|_{\mathcal{D}(\theta)} \in \mathbb{R}^{n_0}$ represents the unknown stochastic solution of the nodes not on the constraint boundaries, n_0 is the number of the nodes not on the constraint boundaries, $\mathbf{X}_i(\theta)|_{\Gamma_j(\theta)} = \mathscr{M}_{i,j}\left(\overline{\mathbf{X}}|_{\Gamma_j}, \theta\right) \in \mathbb{R}^{n_j}$, $j = 1, \cdots, b$ are the stochastic solutions of the nodes on the boundary Γ_j and they can be calculated via Eq. (5), $\widetilde{\mathbf{K}}_{\mathcal{G},i,j} \in \mathbb{R}^{n_i \times n_j}$ are corresponding submatrices, $n_1, \cdots, n_b$ are the number of nodes on the boundary Γ_j, $j = 1, \cdots, b$. The matrix $\mathbf{K}_{\mathcal{G}}$ is also not unique and depends on the selection of the reference domain. For a given reference domain and its mesh discretization, the matrix $\mathbf{K}_{\mathcal{G}}$ is unique and assembled by using the classical finite element method [35]. Eq. (7) is considered as a deterministic finite element equation with stochastic Dirichlet boundary conditions. To calculate the submatrix $\widetilde{\mathbf{K}}_{\mathcal{G},i,j}$, $i, j = 0, 1, \cdots, b$, we let $\{\mathcal{I}_{i,1}, \cdots, \mathcal{I}_{i,n_i}\}$ be the numbering of nodes on the i-th mapped boundary (the case $i = 0$ corresponds to the nodes not on mapped boundaries). The index matrix $\mathcal{II}_{ij}$ is given by

$$\mathcal{II}_{ij} = \begin{bmatrix} \mathcal{I}_{i,1} \\ \vdots \\ \mathcal{I}_{i,n_i} \end{bmatrix} \otimes \left[\mathcal{I}_{j,1}, \cdots, \mathcal{I}_{j,n_j}\right] = \begin{bmatrix} \left(\mathcal{I}_{i,1}, \mathcal{I}_{j,1}\right) & \cdots & \left(\mathcal{I}_{i,1}, \mathcal{I}_{j,n_j}\right) \\ \vdots & \ddots & \vdots \\ \left(\mathcal{I}_{i,n_i}, \mathcal{I}_{j,1}\right) & \cdots & \left(\mathcal{I}_{i,n_i}, \mathcal{I}_{j,n_j}\right) \end{bmatrix} \in \mathbb{R}^{n_i \times n_j}, \tag{8}$$

where $\otimes$ represents the outer product operator. The submatrix $\widetilde{\mathbf{K}}_{\mathcal{G},i,j}$ is thus calculated by

$$\widetilde{\mathbf{K}}_{\mathcal{G},i,j} = \mathbf{K}_{\mathcal{G}}\left[\mathcal{II}_{ij}\right]. \tag{9}$$

Based on the first row of Eq. (7), the stochastic solution $\mathbf{X}_i(\theta)|_{\mathcal{D}(\theta)}$ is solved by an explicit form

$$\mathbf{X}_i(\theta)|_{\mathcal{D}(\theta)} = \sum_{j=1}^{b} \underline{\mathbf{K}}_{\mathcal{G},j} \mathscr{M}_{i,j}\left(\overline{\mathbf{X}}|_{\Gamma j}, \theta\right),\ i = 1, \cdots, d, \tag{10}$$

where the matrices $\underline{\mathbf{K}}_{\mathcal{G},j} = -\widetilde{\mathbf{K}}^{-1}_{\mathcal{G},0,0}\widetilde{\mathbf{K}}_{\mathcal{G},0,j} \in \mathbb{R}^{n_0 \times n_j}$.

The above transformation is summarized in Algorithm 1. In step 1, we choose a reference domain and generate its mesh topology $\left[\mathcal{K}\{\text{Node}, \text{Ele}\}, \left\{\overline{\mathbf{X}}_i\right\}_{i=1}^{d}\right]$, where $\mathcal{K}\{\text{Node}, \text{Ele}\}$ represents the topology of the node numbering and the element numbering. The matrix $\mathbf{K}_{\mathcal{G}}$ in Eq. (6) (or

Algorithm 1 Algorithm for mapping random domains to reference domains

1: Choose a reference domain and generate its mesh topology $\left[\mathcal{K}\left\{\text{Node}, \text{Ele}\right\}, \left\{\overline{\mathbf{X}}_i\right\}_{i=1}^{d}\right]$

2: Assemble the deterministic matrix $\mathbf{K}_{\mathcal{G}}$ in Eq. (6) (or Eq. (7))

3: **for** the geometric boundary $j = 1, \cdots, b$ **do**

4: Calculate the matrices $\underline{\mathbf{K}}_{\mathcal{G},j} = -\widetilde{\mathbf{K}}_{\mathcal{G},0,0}^{-1}\widetilde{\mathbf{K}}_{\mathcal{G},0,j} \in \mathbb{R}^{n_0 \times n_j}$

5: **end**

6: **for** the geometric boundary $j = 1, \cdots, b$ **do**

7: **for** the coordinate dimension $i = 1, \cdots, d$ **do**

8: The coordinate transformation $\mathscr{M}_{i,j}\left(\overline{\mathbf{X}}\big|_{\Gamma_j}, \theta\right)$ on the j-th boundary

9: **end**

10: **end**

11: **for** the coordinate dimension $i = 1, \cdots, d$ **do**

12: Solve the random coordinate $\mathbf{X}_i(\theta)|_{\mathcal{D}(\theta)}$ via Eq. (10)

13: **end**

Eq. (7)) is assembled in step 2, the computation of which is fully deterministic for a given reference domain. From step 3 to 5, the matrices $\widehat{\mathbf{K}}_{\mathcal{G},j}$ are solved by a set of systems of linear equations,

$$\widetilde{\mathbf{K}}_{\mathcal{G},0,0}\underline{\mathbf{K}}_{\mathcal{G},j} = \widetilde{\mathbf{K}}_{\mathcal{G},0,j}, \ j = 1, \cdots, b, \tag{11}$$

which can be solved efficiently by use of existing FEM solvers [35] and can be executed in parallel for each boundary j. After that, the random boundaries are computed by step 6 to 10, which only depends on the choice of the reference domain and is almost independent of the finite element discretization (weakly depends on the number of the discretized nodes on the boundaries). We adopt a non-intrusive way to execute the transformation $\mathscr{M}_{i,j}\left(\overline{\mathbf{X}}\big|_{\Gamma_j}, \boldsymbol{\theta}\right) \in \mathbb{R}^{n_j \times n_s}$ of the i-th coordinate on the j-th boundary, where $\boldsymbol{\theta} = \left\{\theta^{(i)}\right\}_{i=1}^{n_s}$ are n_s sample realizations. In this way, changes in the geometric shape can be accurately captured compared to the approximation method [9, 16]. Also, the procedure from step 6 to 10 is independent of the matrix assembly from step 3 to 5 and they can be executed in parallel. Taking into account $\widehat{\mathbf{K}}_{\mathcal{G},j}$ and $\mathscr{M}_{i,j}\left(\overline{\mathbf{X}}\big|_{\Gamma_j}, \boldsymbol{\theta}\right)$ as calculated in the previous steps, the stochastic solutions $\mathbf{X}_i(\theta)|_{\mathcal{D}(\theta)}$ are computed in step 12.

3. Discretization and assembly on random domains

Based on the above geometric mapping, we transform the random domain into a reference domain. A popular method is to transform the PDE defined on the random domain into a stochastic PDE defined on the reference domain [9, 15]. In this kind of method, the dependency of the PDE on the geometric randomness is transformed into the dependence on stochastic coefficients over the reference domain by means of a random Jacobian matrix. The differential operator of the PDE and the randomness are coupled in this way. In this section, we propose a random isoparametric mapping to deal with the PDEs defined on the random domain, which decouples the differential operator of the PDE and the randomness and has the same computational complexity as the classical isoparametric mapping.

3.1. Random isoparametric mapping

We recall the mesh topology $\left[\mathcal{K}\left\{\text{Node, Ele}\right\}, \overline{\mathbf{X}}\right]$ of the reference domain generated in step 1 in Algorithm 1, where $\overline{\mathbf{X}} = \left[\overline{\mathbf{X}}_1, \cdots, \overline{\mathbf{X}}_d\right] \in \mathbb{R}^{n\times d}$ is the set of node coordinates of the reference domain. We construct a random mesh of the random domain in the form $[\mathcal{K}\left\{\text{Node, Ele}\right\}, \mathbf{X}(\theta)]$, where $\mathbf{X}(\theta) = [\mathbf{X}_1(\theta), \cdots, \mathbf{X}_d(\theta)] \in \mathbb{R}^{n\times d}$ is the stochastic solutions of Eq. (6). In other words, the random mesh is obtained by combining the topology $\mathcal{K}\left\{\text{Node, Ele}\right\}$ of the reference domain and the random coordinates $\mathbf{X}(\theta)$ of the random domain. To illustrate the construction of the random mesh, we consider two sample realizations $\mathcal{D}\left(\theta^{(1)}\right)$ and $\mathcal{D}\left(\theta^{(2)}\right)$ of the random domain shown in Fig. 1a and both of them are transformed into the reference domain $\overline{\mathcal{D}}$. The mesh topology $\mathcal{K}\left\{\text{Node, Ele}\right\}$ is generated on the reference domain by one meshing and the random coordinates $\mathbf{X}(\theta)$ of the random domain is solved by Eq. (6). The realizations of two random meshes $\left[\mathcal{K}\left\{\text{Node, Ele}\right\}, \mathbf{X}\left(\theta^{(1)}\right)\right]$ and $\left[\mathcal{K}\left\{\text{Node, Ele}\right\}, \mathbf{X}\left(\theta^{(2)}\right)\right]$ are thus obtained. They have the same mesh topology $\mathcal{K}\left\{\text{Node, Ele}\right\}$ and only one meshing is involved. Further, we consider the element analysis of the random elements $\mathcal{E}_i\left(\theta^{(1)}\right)$ and $\mathcal{E}_i\left(\theta^{(2)}\right)$ in random domains $\mathcal{D}\left(\theta^{(1)}\right)$ and $\mathcal{D}\left(\theta^{(2)}\right)$, which are mapped from the element $\mathcal{E}_i$ in the reference domain $\overline{\mathcal{D}}$. As shown in Fig. 1b, the elements $\mathcal{E}_i$, $\mathcal{E}_i\left(\theta^{(1)}\right)$ and $\mathcal{E}_i\left(\theta^{(2)}\right)$ have the same element number (i.e. i) and consist of the same node group (i.e. $\left\{n_i^{(1)}, n_i^{(2)}, n_i^{(3)}, n_i^{(4)}\right\}$). The only difference between them is the coordinates of

the node group $\left\{n_i^{(1)}, n_i^{(2)}, n_i^{(3)}, n_i^{(4)}\right\}$. We extend the deterministic isoparametric mapping to random cases and map elements $\mathcal{E}_i\left(\theta^{(1)}\right)$ and $\mathcal{E}_i\left(\theta^{(2)}\right)$ into the isoparametric element $\mathcal{E}_{\text{iso}}$.

Let us simply recall the deterministic isoparametric mapping [35, 36]. For complex geometric domains, irregular elements are used to fit irregular curves (or surfaces), which increases computational complexity of the element analysis. Isoparametric mapping enables meshing geometric domains with irregular elements but performing the element analysis using regular elements. Isoparametric formulations are used to map the irregular elements into regular elements and the mapping functions are usually the same as shape functions used for the solutions. Based on the mapping, the numerical integration on the irregular elements is transformed into that on the regular isoparametric elements. For the explanation of this point, we consider a two-dimensional irregular rectangular element $\mathcal{E}$ with four nodes (x_i, y_i), $i = 1, \cdots, 4$ as the physical element, and a two-dimensional regular rectangular element $\mathcal{E}_{\text{iso}}$ with four nodes (η_i, ζ_i), $i = 1, \cdots, 4$ as the reference element. The basic idea of the deterministic isoparametric mapping is to use the shape functions $N_i(\eta, \zeta)$, $i = 1, \cdots, 4$ on the reference element $\mathcal{E}_{\text{iso}}$ to describe the shape of $\mathcal{E}$ through the following equations

$$x(\eta, \zeta) = \sum_{i=1}^{4} x_i N_i(\eta, \zeta), \; y(\eta, \zeta) = \sum_{i=1}^{4} y_i N_i(\eta, \zeta). \tag{12}$$

The solution $u(\eta, \zeta)$ is approximated in a similar way

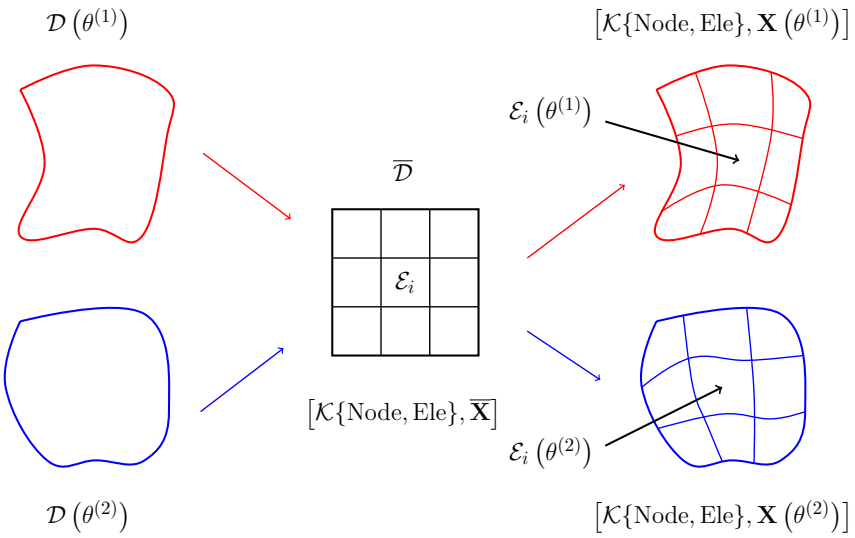
$$u(\eta, \zeta) = \sum_{i=1}^{4} u_i N_i(\eta, \zeta). \tag{13}$$

After the above mapping, we can transform the irregular elements and the solution into those on the reference elements. The element analysis is then performed on the reference elements, such as calculating numerical integration and assembling element matrices.

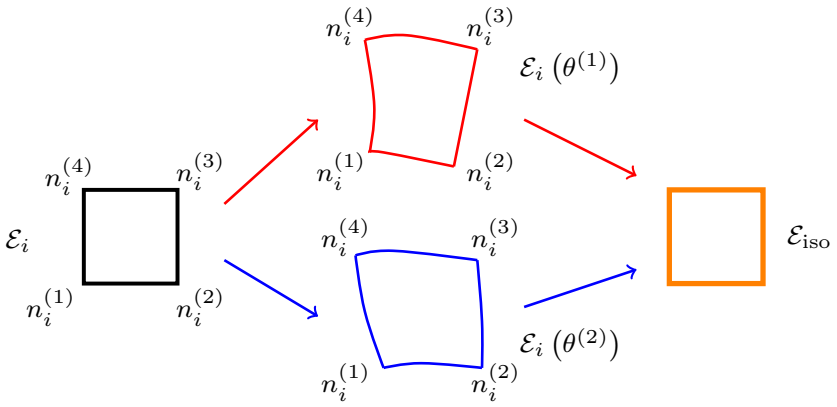
We extend the above idea of deterministic isoparametric mapping to random cases. Considering the weak form of Eq. (1) discretized by finite element method, the stochastic solution $\mathbf{u}^{(i)}(\theta)$ of the i-th element is thus represented by shape functions of the isoparametric element in a way similar to the deterministic isoparametric mapping Eq. (13)

$$\mathbf{u}^{(i)}(\theta) = \sum_{j=1}^{n_I} N_j(\boldsymbol{\eta})\, \mathbf{u}^{(i,j)}(\theta) \tag{14}$$

and the random coordinates are approximated in a similar way,



(a) Random domain transformation and the mesh topology reuse.



(b) Random isoparametric mapping.

Figure 1: Domain transformation and random isoparametric mapping: (a) domain transformation and random mesh obtained from the reference domain, (b) random isoparametric mapping.

$$\mathbf{x}_k^{(i)}(\theta) = \sum_{j=1}^{n_I} N_j(\boldsymbol{\eta})\, \mathbf{x}_k^{(i,j)}(\theta),\ k = 1, \cdots, d, \tag{15}$$

where $N_j(\boldsymbol{\eta})$ is the shape function of the isoparametric element, $\boldsymbol{\eta} \in \mathbb{R}^d$ represents the local coordinate of the isoparametric element, $\mathbf{u}^{(i,j)}(\theta)$ is the value of the stochastic solution $\mathbf{u}(\theta)$ on the j-th node of the i-th element, $\mathbf{x}_k^{(i,j)}(\theta)$ is the value of the random coordinate $\mathbf{x}_k(\theta)$ on the j-th node of the i-th element, n_I is the number of nodes of the i-th element. It is noted that the

random isoparametric mapping in this section is applicable to any type of element although only the rectangular element is shown in Fig. 1. Also, different $N_j(\boldsymbol{\eta})$ and n_I should be adopted if elements of different types and orders are used for the finite element discretization. Thus Eq. (14) and Eq. (15) can represent the isoparametric mapping analyses of general cases. Also, from now on we discard the reference domain $\overline{\mathcal{D}}$ and all analyses are performed on the random domain and the corresponding random mesh.

Inspired by the deterministic isoparametric analysis, the random transformation between the global coordinate $\mathbf{x}^{(i)}(\theta)$ and the local coordinate $\boldsymbol{\eta}$ is given by

$$\frac{\partial N_j(\boldsymbol{\eta})}{\partial \boldsymbol{\eta}} = \mathbf{J}\left(\mathbf{x}^{(i)}(\theta)\right) \frac{\partial N_j(\boldsymbol{\eta})}{\partial \mathbf{x}^{(i)}(\theta)},\ j = 1, \cdots, n_I, \tag{16}$$

where the vectors $\frac{\partial N_j(\boldsymbol{\eta})}{\partial \boldsymbol{\eta}} \in \mathbb{R}^d$ and $\frac{\partial N_j(\boldsymbol{\eta})}{\partial \mathbf{x}^{(i)}(\theta)} \in \mathbb{R}^d$ are defined as

$$\frac{\partial N_j(\boldsymbol{\eta})}{\partial \boldsymbol{\eta}} = \left[\frac{\partial N_j(\boldsymbol{\eta})}{\partial \eta_1}, \cdots, \frac{\partial N_j(\boldsymbol{\eta})}{\partial \eta_d}\right]^T \in \mathbb{R}^d,\ \frac{\partial N_j(\boldsymbol{\eta})}{\partial \mathbf{x}^{(i)}(\theta)} = \left[\frac{\partial N_j(\boldsymbol{\eta})}{\partial x_1^{(i)}(\theta)}, \cdots, \frac{\partial N_j(\boldsymbol{\eta})}{\partial x_d^{(i)}(\theta)}\right]^T \in \mathbb{R}^d \tag{17}$$

and $\mathbf{J}\left(\mathbf{x}^{(i)}(\theta)\right)$ is the random Jacobian matrix given by

$$\mathbf{J}\left(\mathbf{x}^{(i)}(\theta)\right) = \begin{bmatrix} \frac{\partial x_1^{(i)}(\theta)}{\partial \eta_1} & \cdots & \frac{\partial x_d^{(i)}(\theta)}{\partial \eta_1} \\ \vdots & \ddots & \vdots \\ \frac{\partial x_1^{(i)}(\theta)}{\partial \eta_d} & \cdots & \frac{\partial x_d^{(i)}(\theta)}{\partial \eta_d} \end{bmatrix} \in \mathbb{R}^{d\times d}. \tag{18}$$

The inverse transformation of Eq. (16) is

$$\frac{\partial N_j(\boldsymbol{\eta})}{\partial \mathbf{x}^{(i)}(\theta)} = \mathbf{J}^{-1}\left(\mathbf{x}^{(i)}(\theta)\right) \frac{\partial N_j(\boldsymbol{\eta})}{\partial \boldsymbol{\eta}}, \tag{19}$$

where $\mathbf{J}^{-1}\left(\mathbf{x}^{(i)}(\theta)\right) \in \mathbb{R}^{d\times d}$ is the inverse matrix of the Jacobian matrix $\mathbf{J}\left(\mathbf{x}^{(i)}(\theta)\right)$. In this way, we transform the random global coordinates into the deterministic local coordinates. The above random isoparametric mapping is very similar to the classical isoparametric analysis but involves the random coordinates instead of the deterministic coordinates. We will extend the classical isoparametric mapping for solving PDEs to a random case in the next section.

3.2. Elastic equation defined on the random domain

We illustrate the proposed random isoparametric analysis using an elastic equation defined on a random domain, but the proposed method can be applied to more general PDEs, as will be shown in the example 6.2. Consider a two-dimensional linear elastic equation written as

$$\operatorname{div}\left(\sigma\left(\mathbf{x}\left(\theta\right)\right)\right)+f\left(\mathbf{x}\left(\theta\right)\right)=0 \quad \text{in} \quad \mathcal{D}\left(\theta\right), \tag{20}$$

where $\operatorname{div}(\cdot)$ represents the divergence operator, $\sigma\left(\mathbf{x}\left(\theta\right)\right)$ is the stochastic stress tensor and $f\left(\mathbf{x}\left(\theta\right)\right)$ is the stochastic external force. Based on the finite element discretization of Eq. (20), the stochastic strain $\varepsilon^{(i)}\left(\theta\right)$ of the i-th element is given by

$$\varepsilon^{(i)}\left(\theta\right)=\left[\frac{\partial u_1^{(i)}\left(\mathbf{x}^{(i)}\left(\theta\right)\right)}{\partial x_1^{(i)}\left(\theta\right)},\frac{\partial u_2^{(i)}\left(\mathbf{x}^{(i)}\left(\theta\right)\right)}{\partial x_2^{(i)}\left(\theta\right)},\frac{\partial u_1^{(i)}\left(\mathbf{x}^{(i)}\left(\theta\right)\right)}{\partial x_2^{(i)}\left(\theta\right)}+\frac{\partial u_2^{(i)}\left(\mathbf{x}^{(i)}\left(\theta\right)\right)}{\partial x_1^{(i)}\left(\theta\right)}\right]=\left[\mathbf{B}_1\left(\theta\right),\cdots,\mathbf{B}_{n_I}\left(\theta\right)\right]\mathbf{u}^{(i)}\left(\theta\right), \tag{21}$$

where $\mathbf{u}^{(i)}\left(\theta\right)=\left[\mathbf{u}^{(i,1)}\left(\theta\right)^T,\cdots,\mathbf{u}^{(i,n_I)}\left(\theta\right)^T\right]^T$ is the stochastic solution of all nodes of the i-th element, $\mathbf{u}^{(i,j)}\left(\theta\right)=\left[u_1^{(i,j)}\left(\theta\right),u_2^{(i,j)}\left(\theta\right)\right]^T$ is the stochastic solution of the j-th node of the i-th element. The submatrix $\mathbf{B}_j\left(\theta\right)$ of the strain matrix $\mathbf{B}\left(\theta\right)=\left[\mathbf{B}_1\left(\theta\right),\cdots,\mathbf{B}_{n_I}\left(\theta\right)\right]$ is defined as

$$\mathbf{B}_j\left(\theta\right)=\begin{bmatrix}\frac{\partial N_j(\boldsymbol{\eta})}{\partial x_1^{(i)}(\theta)} & 0\\ 0 & \frac{\partial N_j(\boldsymbol{\eta})}{\partial x_2^{(i)}(\theta)}\\ \frac{\partial N_j(\boldsymbol{\eta})}{\partial x_2^{(i)}(\theta)} & \frac{\partial N_j(\boldsymbol{\eta})}{\partial x_1^{(i)}(\theta)}\end{bmatrix}\in\mathbb{R}^{3\times 2},\ j=1,\cdots,n_I. \tag{22}$$

Further, the element stochastic stiffness matrix $\mathbf{k}^{(i)}\left(\theta\right)$ can be calculated via the isoparametric element [35, 36]

$$\mathbf{k}^{(i)}\left(\theta\right)=\int_{\mathcal{D}^{(i)}(\theta)}\mathbf{B}^T\left(\theta\right)\mathbf{C}\mathbf{B}\left(\theta\right)\mathrm{d}\mathbf{x}\left(\theta\right)=\int_{\mathcal{E}_{\mathrm{iso}}}\mathbf{B}^T\left(\theta\right)\mathbf{C}\mathbf{B}\left(\theta\right)\left|\mathbf{J}\left(\mathbf{x}\left(\theta\right)\right)\right|\mathrm{d}\boldsymbol{\eta}\in\mathbb{R}^{2n_I\times 2n_I}, \tag{23}$$

where $i=1,\cdots,n_e$ is the element numbering, $\mathcal{D}^{(i)}\left(\theta\right)$ is the i-th random element, $\left|\mathbf{J}\left(\mathbf{x}\left(\theta\right)\right)\right|$ represents determinant of the matrix $\mathbf{J}\left(\mathbf{x}\left(\theta\right)\right)$ and the material matrix $\mathbf{C}$ is given by

$$\mathbf{C}=\frac{E}{1-\mu^2}\begin{bmatrix}1 & \mu & 0\\ \mu & 1 & 0\\ 0 & 0 & \frac{1-\mu}{2}\end{bmatrix}. \tag{24}$$

where E and μ are the Young's modulus and the Poisson's ratio, respectively. We can compute the global stochastic stiffness matrix by assembling the element stochastic stiffness matrix,

$$\mathbf{K}\left(\theta\right)=\bigcup_{i=1}^{n_e}\left(\mathbf{k}^{(i)}\left(\theta\right)\right)\in\mathbb{R}^{n\times n}, \tag{25}$$

where $\bigcup_{i=1}^{n_e}$ is the assembly operation. The global stochastic force vector $\mathbf{F}\left(\theta\right)=\bigcup_{e=1}^{n_e}\left(\mathbf{f}^{(e)}\left(\theta\right)\right)\in\mathbb{R}^n$ can be obtained in a similar way, where $\mathbf{f}^{(e)}\left(\theta\right)\in\mathbb{R}^{2n_I}$ is the element stochastic force vector.

The inverse Jacobian matrix $\mathbf{J}^{-1}\left(\mathbf{x}^{(i)}(\theta)\right)$ in Eq. (19) has the following form for 2D finite elements,

$$\mathbf{J}^{-1}\left(\mathbf{x}^{(i)}(\theta)\right)=\frac{1}{\left|\mathbf{J}\left(\mathbf{x}^{(i)}(\theta)\right)\right|}\begin{bmatrix}\frac{\partial x_2^{(i)}(\theta)}{\partial \eta_2} & -\frac{\partial x_2^{(i)}(\theta)}{\partial \eta_1}\\ -\frac{\partial x_1^{(i)}(\theta)}{\partial \eta_2} & \frac{\partial x_1^{(i)}(\theta)}{\partial \eta_1}\end{bmatrix}\in\mathbb{R}^{2\times 2}, \tag{26}$$

where the component is given by

$$\frac{\partial x_k^{(i)}(\theta)}{\partial \eta_j}=\sum_{l=1}^{n_l}\frac{\partial N_l(\boldsymbol{\eta})}{\partial \eta_j}x_k^{(i,l)}(\theta),\ k,j=1,2 \tag{27}$$

and its determinant is

$$\left|\mathbf{J}\left(\mathbf{x}^{(i)}(\theta)\right)\right|=\frac{\partial x_1^{(i)}(\theta)}{\partial \eta_1}\frac{\partial x_2^{(i)}(\theta)}{\partial \eta_2}-\frac{\partial x_1^{(i)}(\theta)}{\partial \eta_2}\frac{\partial x_2^{(i)}(\theta)}{\partial \eta_1}. \tag{28}$$

Combining Eq. (19) and Eq. (26) we rewrite Eq. (22) as

$$\mathbf{B}_j(\theta)=\begin{bmatrix}1&0\\0&0\\0&1\end{bmatrix}\mathbf{J}^{-1}\left(\mathbf{x}^{(i)}(\theta)\right)\left[\frac{\partial N_j(\boldsymbol{\eta})}{\partial \boldsymbol{\eta}},\mathbf{0}_{2\times 1}\right]+\begin{bmatrix}0&0\\0&1\\1&0\end{bmatrix}\mathbf{J}^{-1}\left(\mathbf{x}^{(i)}(\theta)\right)\left[\mathbf{0}_{2\times 1},\frac{\partial N_j(\boldsymbol{\eta})}{\partial \boldsymbol{\eta}}\right], \tag{29}$$

which indicates that the random coordinate $\mathbf{x}(\theta)$ is only related to the random inverse Jacobian matrix $\mathbf{J}^{-1}\left(\mathbf{x}^{(i)}(\theta)\right)$. We compute the random coordinate $\mathbf{x}(\theta)$ by the non-intrusive solution of Eq. (6) in practice. Following that, we still adopt the non-intrusive way to compute the matrix $\mathbf{B}_j(\theta)$ in Eq. (29) and assemble the element stochastic stiffness matrix in Eq. (23), which can be performed by using the deterministic FEM procedure to loop random samples $\mathbf{x}\left(\theta^{(i)}\right), i=1,\cdots,n_s$. It is noted that Eq. (23) and Eq. (29) are only applicable to the elastic equation considered in this section, and other assembly formats should be adopted for more general problems.

It is seen from Eq. (23) that if the material parameters are not spatially dependent, the material matrix $\mathbf{C}$ is independent of the random coordinate $\mathbf{x}(\theta)$. If the random material matrix $\mathbf{C}(\theta)$ is involved in the problem, the same computational framework as described above can be employed to assemble the stiffness matrix. In the numerical example 6.2, we will show that the proposed method can be applied to the stochastic PDE defined on the random geometry, the random coefficients of which are simulated as random variables. However, since simulating random fields defined on random domains is still an open problem, PDEs that couple random geometries and random coefficient fields require further study. Also, although we only consider linear PDEs in

this paper, the proposed method is readily extended to nonlinear PDEs. For instance, we can describe inelastic behavior by adopting a nonlinear material matrix $\mathbf{C}$ in Eq. (23), which is still independent of the random coordinate and can be assembled in the same way as deterministic FEM. The applicability of the proposed method to nonlinear PDEs on random domains will be verified with the aid of a nonlinear heat equation on a random domain in Section 6.4.

4. Solution algorithm of stochastic finite element equations

Based on the above assembly of the stochastic matrix and the stochastic vector, we can obtain a stochastic finite element equation (SFEE)

$$\mathbf{K}(\theta)\,\mathbf{u}(\theta) = \mathbf{F}(\theta). \tag{30}$$

It is noted that the stochastic matrix $\mathbf{K}(\theta)$ and the stochastic vector $\mathbf{F}(\theta)$ are assembled in a non-intrusive way. By repeating the realizations $\mathbf{K}\left(\theta^{(i)}\right) \in \mathbb{R}^{n\times n}$ and $\mathbf{F}\left(\theta^{(i)}\right) \in \mathbb{R}^{n}$ for n_s different samples, they usually have the forms $\mathbf{K}(\boldsymbol{\theta}) \in \mathbb{R}^{n\times n\times n_s}$ and $\mathbf{F}(\boldsymbol{\theta}) \in \mathbb{R}^{n\times n_s}$. Much memory is needed to store $\mathbf{K}(\theta)$ and $\mathbf{F}(\theta)$ if the sample size n_s is large. However, if a small size n_s is adopted, the accuracy of the stochastic solution will be low. To avoid the difficulties, we adopt a weak-intrusive SFEM [37, 38] to solve Eq. (30) in this paper.

4.1. A weak-intrusive SFEM

To solve Eq. (30), we approximate the stochastic solution $\mathbf{u}(\theta)$ in the form,

$$\mathbf{u}(\theta) \approx \sum_{i=1}^{k} \lambda_i(\theta)\,\mathbf{d}_i = \mathbf{D}\boldsymbol{\Lambda}(\theta), \tag{31}$$

where $\lambda_i(\theta) \in \mathbb{R}$ denotes a scalar random variables, $\boldsymbol{\Lambda}(\theta) = [\lambda_1(\theta), \cdots, \lambda_k(\theta)]^{\mathrm{T}} \in \mathbb{R}^k$ is a random variable vector, $\mathbf{d}_i \in \mathbb{R}^n$ denotes a deterministic vector and $\mathbf{D} = [\mathbf{d}_1, \cdots, \mathbf{d}_k] \in \mathbb{R}^{n\times k}$ is a matrix. All of these terms are not known a priori and need to be solved. As it is not easy to compute them at once, we adopt a sequential way to solve the couple $\{\lambda_i(\theta), \mathbf{d}_i\}_i$ one by one. For this purpose, we assume that the first $k-1$ couples $\{\lambda_i(\theta), \mathbf{d}_i\}_{i=1}^{k-1}$ have been determined and Eq. (30) is thus transformed into

$$\mathbf{K}(\theta)\,\lambda_k(\theta)\,\mathbf{d}_k = \mathbf{F}_k(\theta), \tag{32}$$

where $\mathbf{F}_k(\theta) = \mathbf{F}(\theta) - \mathbf{K}(\theta)\sum_{i=1}^{k-1}\lambda_i(\theta)\mathbf{d}_i$. In this way, only $\lambda_k(\theta)$ and $\mathbf{d}_k$ are unknown. They are solved via an alternating iterative algorithm as follows.

For a known random variable $\lambda_k(\theta)$ (or a given initial value, the details of which can be found in Section 5), a deterministic finite element equation is obtained by use of the stochastic Galerkin method [18, 25],

$$\mathbb{E}\left\{\lambda_k^2(\theta)\mathbf{K}(\theta)\right\}\mathbf{d}_k = \mathbb{E}\left\{\lambda_k(\theta)\mathbf{F}_k(\theta)\right\}, \tag{33}$$

which can be solved efficiently via existing FEM solvers [35]. To speed up the convergence, we let the vector $\mathbf{d}_k$ orthogonal to the obtained vectors $\{\mathbf{d}_i\}_{i=1}^{k-1}$. The Gram-Schmidt orthogonalization is adopted,

$$\mathbf{d}_k = \mathbf{d}_k - \sum_{i=1}^{k-1}\frac{\mathbf{d}_k^{\mathrm{T}}\mathbf{d}_i}{\mathbf{d}_i^T\mathbf{d}_i}\mathbf{d}_i, \tag{34}$$

where $\{\mathbf{d}_i\}_{i=1}^{k-1}$ are normalized orthogonal vectors that meet $\mathbf{d}_i^{\mathrm{T}}\mathbf{d}_j = \delta_{ij}$, where δ_{ij} is the Kronecker delta. Based on the solution $\mathbf{d}_k$ of Eq. (33), the random variable $\lambda_k(\theta)$ is updated via the Galerkin approach,

$$\left[\mathbf{d}_k^{\mathrm{T}}\mathbf{K}(\theta)\mathbf{d}_k\right]\lambda_k(\theta) = \mathbf{d}_k^{\mathrm{T}}\mathbf{F}_k(\theta). \tag{35}$$

In this paper we assume that the matrix $\mathbf{K}(\theta)$ is positive (or negative) definite, i.e. $\mathbf{x}^{\mathrm{T}}\mathbf{K}(\theta)\mathbf{x} > 0\,(<0)$ holds for the nonzero vector $\mathbf{x}\in\mathbb{R}^n$, which holds true in many problems. For the indefinite matrix $\mathbf{K}(\theta)$, Eq. (35) is insolvable when the realization $\mathbf{d}_k^{\mathrm{T}}\mathbf{K}\left(\theta^{(i)}\right)\mathbf{d}_k = 0$. The residual minimization can be used to build a numerically stable equation $\left[\mathbf{d}_k^{\mathrm{T}}\mathbf{K}(\theta)^{\mathrm{T}}\mathbf{K}(\theta)\mathbf{d}_k\right]\lambda_k(\theta) = \mathbf{d}_k^{\mathrm{T}}\mathbf{K}(\theta)^{\mathrm{T}}\mathbf{F}_k(\theta)$, the details of which will not be discussed in this paper. In order to solve the stochastic algebraic equation (35) efficiently, we adopt a non-intrusive way [37, 38], which is easily implemented and can be applied to high-dimensional stochastic problems. Specifically, if the matrix $\mathbf{K}(\theta)$ is not an indefinite matrix, Eq. (35) is solved by

$$\lambda_k(\boldsymbol{\theta}) = \frac{\mathbf{d}_k^{\mathrm{T}}\mathbf{F}_k(\boldsymbol{\theta})}{\mathbf{d}_k^T\mathbf{K}(\boldsymbol{\theta})\mathbf{d}_k} \in \mathbb{R}^{n_s}, \tag{36}$$

where $\boldsymbol{\theta} = \left\{\theta^{(i)}\right\}_{i=1}^{n_s} \in \mathbb{R}^{n_s}$ represents n_s sample realizations of the random variables, $\lambda_k(\boldsymbol{\theta}) \in \mathbb{R}^{n_s}$ is the sample vector of the random variable $\lambda_k(\theta)$ and statistical methods are used to provide the probability characteristics of $\lambda_k(\theta)$. Eq. (36) is insensitive to stochastic dimensions and has low

computational costs even for very high-dimensional stochastic problems. In this way, Eq. (31) is considered a weak-intrusive approximation, which combines the high efficiency of intrusive methods and the weakly dimensional dependence of non-intrusive methods.

A set of couples $\{\lambda_i(\theta), \mathbf{d}_i\}_{i=1}^k$ can be obtained by the above iteration and a specific criterion for the selection of the size k will be discussed in Section 5. It is noted that these couples are solved in a sequential way, thus they do not exactly fulfill Eq. (30). The stochastic solution $\mathbf{u}(\theta)$ approximated by the couples $\{\lambda_i(\theta), \mathbf{d}_i\}_{i=1}^k$ may have poor accuracy in some cases. To improve the accuracy, we consider $\{\mathbf{d}_i\}_{i=1}^k$ as reduced bases and recompute the random vector $\mathbf{\Lambda}(\theta)$ in Eq. (31) by

$$\left[\mathbf{D}^{\mathrm{T}}\mathbf{K}(\theta)\mathbf{D}\right]\mathbf{\Lambda}(\theta) = \mathbf{D}^{\mathrm{T}}\mathbf{F}(\theta), \tag{37}$$

where the reduced-order matrix $\mathbf{D}^{\mathrm{T}}\mathbf{K}(\theta)\mathbf{D} \in \mathbb{R}^{k\times k}$ and the reduced-order vector $\mathbf{D}^{\mathrm{T}}\mathbf{F}(\theta) \in \mathbb{R}^k$. The computational effort is very cheap due to the small size of Eq. (37). The final stochastic solution is obtained by n_s solutions $\mathbf{\Lambda}\left(\theta^{(i)}\right)$, $i = 1, \cdots, n_s$ of Eq. (37).

We mention that although similar expansions have been widely used in the computational framework of PGD methods [39, 40], the proposed approximation Eq. (31) is more powerful than the classical PGD methods and is more suitable for solving high-dimensional and nonlinear stochastic problems due to its weak intrusiveness. With the aid of numerical examples, we will show that it is also applicable to solve Eq. (30) with a small sample size n_s.

4.2. Computational aspects

In this section we will discuss the numerical details of the implementation of the proposed method, including the implementations at the global and element levels.

4.2.1. Implementation at the global level

We recall that the stochastic matrix $\mathbf{K}(\boldsymbol{\theta}) \in \mathbb{R}^{n\times n\times n_s}$ and the stochastic vector $\mathbf{F}(\boldsymbol{\theta}) \in \mathbb{R}^{n\times n_s}$ are obtained in a non-intrusive way and the sample vector of the random variable $\lambda_k(\theta)$ are given by $\lambda_k(\boldsymbol{\theta}) \in \mathbb{R}^{n_s}$. The computations of Eq. (33) and Eq. (36) can be executed at the global level via

$$\mathbb{E}\left\{\lambda_k^2(\theta)\,\mathbf{K}(\theta)\right\} = \frac{1}{n_s}\sum_{j=1}^{n_s}\lambda_k^2\left(\theta^{(j)}\right)\mathbf{K}\left(\theta^{(j)}\right) \in \mathbb{R}^{n\times n}, \tag{38}$$

$$\mathbb{E}\left\{\lambda_k(\theta)\,\mathbf{F}(\theta)\right\} = \frac{1}{n_s}\mathbf{F}(\boldsymbol{\theta})\,\lambda_k(\boldsymbol{\theta}) \in \mathbb{R}^{n}, \tag{39}$$

$$\mathbf{d}_k^{\mathrm{T}}\mathbf{K}(\theta)\,\mathbf{d}_k = \left[\mathbf{d}_k^{\mathrm{T}}\mathbf{K}\left(\theta^{(1)}\right)\mathbf{d}_k, \cdots, \mathbf{d}_k^{\mathrm{T}}\mathbf{K}\left(\theta^{(n_s)}\right)\mathbf{d}_k\right]^{\mathrm{T}} \in \mathbb{R}^{n_s}, \tag{40}$$

$$\mathbf{d}_k^{\mathrm{T}}\mathbf{F}(\theta) = \mathbf{F}(\boldsymbol{\theta})^{\mathrm{T}}\,\mathbf{d}_k \in \mathbb{R}^{n_s}, \tag{41}$$

which provides a direct way to perform the stochastic computations. It is seen that Eq. (39) and Eq. (41) can be computed efficiently. However, Eq. (38) and Eq. (40) may need much more computational effort and storage memory. In practice, we store $\mathbf{K}(\boldsymbol{\theta}) \in \mathbb{R}^{n\times n\times n_s}$ in a sparse tensor structure and adopt efficient tensor multiplications to execute the computations.

4.2.2. Implementation at the element level

To avoid the large storage memory required for the above implementation, we implement the stochastic computations at the element level and deterministic assemblies are then executed, which corresponds

$$\mathbb{E}\left\{\lambda_k^2(\theta)\,\mathbf{K}(\theta)\right\} = \bigcup_{i=1}^{n_e}\left(\mathbb{E}\left\{\lambda_k^2(\boldsymbol{\theta})\,\mathbf{k}^{(i)}(\boldsymbol{\theta})\right\}\right) = \bigcup_{i=1}^{n_e}\left(\frac{1}{n_s}\sum_{j=1}^{n_s}\lambda_k^2\left(\theta^{(j)}\right)\mathbf{k}^{(i)}\left(\theta^{(j)}\right)\right) \in \mathbb{R}^{n\times n}, \tag{42}$$

$$\mathbb{E}\left\{\lambda_k(\theta)\,\mathbf{F}(\theta)\right\} = \bigcup_{i=1}^{n_e}\left(\mathbb{E}\left\{\lambda_k(\boldsymbol{\theta})\,\mathbf{f}^{(i)}(\boldsymbol{\theta})\right\}\right) = \bigcup_{i=1}^{n_e}\left(\frac{1}{n_s}\mathbf{f}^{(i)}(\boldsymbol{\theta})\,\lambda_k(\boldsymbol{\theta})\right) \in \mathbb{R}^{n}, \tag{43}$$

$$\mathbf{d}_k^{\mathrm{T}}\mathbf{K}(\theta)\,\mathbf{d}_k = \left[\sum_{i=1}^{n_e}\mathbf{d}_k^{(i)\mathrm{T}}\mathbf{k}^{(i)}\left(\theta^{(1)}\right)\mathbf{d}_k^{(i)}, \cdots, \sum_{i=1}^{n_e}\mathbf{d}_k^{(i)\mathrm{T}}\mathbf{k}^{(i)}\left(\theta^{(n_s)}\right)\mathbf{d}_k^{(i)}\right]^{\mathrm{T}} \in \mathbb{R}^{n_s}, \tag{44}$$

$$\mathbf{d}_k^{\mathrm{T}}\mathbf{F}(\theta) = \sum_{i=1}^{n_e}\mathbf{f}^{(i)}(\boldsymbol{\theta})^{\mathrm{T}}\,\mathbf{d}_k^{(i)} \in \mathbb{R}^{n_s}, \tag{45}$$

where the element matrix and vector are $\mathbf{k}^{(i)}(\boldsymbol{\theta}) \in \mathbb{R}^{n_I\times n_I\times n_s}$ and $\mathbf{f}^{(i)}(\boldsymbol{\theta}) \in \mathbb{R}^{n_I\times n_s}$. In this way, only the element matrix and vector are stored for each iteration, which reduces the storage memory. However, we need to reassemble the element matrix and vector for each iteration and more computational effort is needed for the assemblies compared to the implementation at the global level.

5. Algorithm implementation

The above iterative algorithm for solving PDEs defined on random domains is summarized in Algorithm 2, which consists of two-loop procedures. The inner loop is from step 7 to 24 and is used to compute the couple $\{\lambda_k(\theta), \mathbf{d}_k\}$. To execute the inner loop, random samples $\lambda_k^{(0)}(\boldsymbol{\theta}) \in \mathbb{R}^{n_s}$ are initialized in step 6. All nonzero vectors of size n_s can be used as the initial random samples and it has little influence on the computational accuracy and efficiency of the proposed method. With the initial random samples, the deterministic vector $\mathbf{d}_k^{(j)}$ is computed in step 14 by solving linear finite element equations. By using the Gram-Schmidt orthogonalization in step 15, $\mathbf{d}_k^{(j)}$ is orthogonalized and normalized along the whole iteration. With the obtained vector $\mathbf{d}_k^{(j)}$, the random variable $\lambda_k^{(j)}(\theta)$ is calculated in the form of random samples $\lambda_k^{(j)}(\boldsymbol{\theta}) \in \mathbb{R}^{n_s}$ in step 22. Two numerical strategies can be adopted to implement the above iteration, i.e. the implementation at the global level via steps 3, 9, 17 and the implementation at the local level via steps 11, 12, 19, 20. After the inner loop, the stochastic solution $\mathbf{u}_k(\theta)$ in step 25 of the outer loop is approximated recursively to meet Eq. (30). Following that, based on the known matrix $\mathbf{D}$, the random vector $\boldsymbol{\Lambda}(\theta)$ is recalculated by solving n_s k-dimensional linear stochastic finite element equations. It is noted that we can adopt different sample sizes in step 6 and step 28. In practice, a small size n_s is first used in step 6 and a large sample size is adopted to recompute the random vector $\boldsymbol{\Lambda}(\theta)$, which saves a lot of computational costs but still has good accuracy. The performance of this strategy will be illustrated in numerical examples.

Also, two iterative criteria in Algorithm 2 are used to check the convergence, i.e. $\varepsilon_{\mathbf{d},j}$ in step 23 and $\varepsilon_{\mathbf{u},k}$ in step 26. The locally iterative error $\varepsilon_{\mathbf{d},j}$ is defined as

$$\varepsilon_{\mathbf{d},j} = \frac{\left\|\mathbf{d}_k^{(j)} - \mathbf{d}_k^{(j-1)}\right\|}{\left\|\mathbf{d}_k^{(j)}\right\|} = 2 - 2\mathbf{d}_k^{(j)\mathrm{T}}\mathbf{d}_k^{(j-1)}, \tag{46}$$

where the operator $\|\square\| = \mathbb{E}\left\{\square^{\mathrm{T}}\square\right\}$. It measures the difference between the vectors $\mathbf{d}_k^{(j)}$ and $\mathbf{d}_k^{(j-1)}$ and the calculation is stopped when $\mathbf{d}_k^{(j)}$ is almost the same as $\mathbf{d}_k^{(j-1)}$. Similarly, the globally iterative error $\varepsilon_{\mathbf{u},k}$ is defined as

$$\varepsilon_{\mathbf{u},k} = \frac{\|\mathbf{u}_k(\theta) - \mathbf{u}_{k-1}(\theta)\|}{\|\mathbf{u}_k(\theta)\|} = \frac{\mathbb{E}\left\{\lambda_k^2(\theta)\right\}\mathbf{d}_k^{\mathrm{T}}\mathbf{d}_k}{\sum_{i,j=1}^{k}\mathbb{E}\left\{\lambda_i(\theta)\lambda_j(\theta)\right\}\mathbf{d}_i^{\mathrm{T}}\mathbf{d}_j} = \frac{\mathbb{E}\left\{\lambda_k^2(\theta)\right\}}{\sum_{i=1}^{k}\mathbb{E}\left\{\lambda_i^2(\theta)\right\}}, \tag{47}$$

Algorithm 2 Algorithm for solving PDEs defined on random domains

Generate the mesh topology $\mathcal{K}\{\text{Node}, \text{Ele}\}$ of the reference domain and solve the random coordinate $\mathbf{x}(\theta)$ via Algorithm 1
if *Implementation at the global level* **then**
    Assemble the tensor $\mathbf{K}(\boldsymbol{\theta}) \in \mathbb{R}^{n\times n\times n_s}$ and the matrix $\mathbf{F}(\boldsymbol{\theta}) \in \mathbb{R}^{n\times n_s}$
end
while $\varepsilon_{\mathbf{u},k} > \varepsilon_{\mathbf{u}}$ **do**
    Initialize random samples $\lambda_k^{(0)}(\boldsymbol{\theta}) = \left\{\lambda_k^{(0)}\left(\theta^{(i)}\right)\right\}_{i=1}^{n_s} \in \mathbb{R}^{n_s}$
    while $\varepsilon_{\mathbf{d},j} > \varepsilon_{\mathbf{d}}$ **do**
        if *Implementation at the global level* **then**
            Compute $\mathbb{E}\left\{\lambda_k^2(\theta)\mathbf{K}(\theta)\right\}$ and $\mathbb{E}\left\{\lambda_k(\theta)\mathbf{F}_k(\theta)\right\}$ by Eq. (38) and Eq. (39)
        else if *Implementation at the local level* **then**
            Assemble the tensor $\mathbf{k}^{(i)}(\boldsymbol{\theta}) \in \mathbb{R}^{n_I\times n_I\times n_s}$ and the matrix $\mathbf{f}^{(i)}(\boldsymbol{\theta}) \in \mathbb{R}^{n_I\times n_s}$
            Compute $\mathbb{E}\left\{\lambda_k^2(\theta)\mathbf{K}(\theta)\right\}$ and $\mathbb{E}\left\{\lambda_k(\theta)\mathbf{F}_k(\theta)\right\}$ by Eq. (42) and Eq. (43)
        end
        Compute the deterministic vector $\mathbf{d}_k^{(j)}$ via Eq. (33)
        Orthogonalize $\mathbf{d}_k^{(j)} \perp \{\mathbf{d}_i\}_{i=1}^{k-1}$ and normalize $\left\|\mathbf{d}_k^{(j)}\right\| = 1$
        if *Implementation at the global level* **then**
            Compute $\mathbf{d}_k^{\mathrm{T}}\mathbf{K}(\theta)\mathbf{d}_k$ and $\mathbf{d}_k^{\mathrm{T}}\mathbf{F}_k(\theta)$ by Eq. (40) and Eq. (41)
        else if *Implementation at the local level* **then**
            Assemble the tensor $\mathbf{k}^{(i)}(\boldsymbol{\theta}) \in \mathbb{R}^{n_I\times n_I\times n_s}$ and the matrix $\mathbf{f}^{(i)}(\boldsymbol{\theta}) \in \mathbb{R}^{n_I\times n_s}$
            Compute $\mathbf{d}_k^{\mathrm{T}}\mathbf{K}(\theta)\mathbf{d}_k$ and $\mathbf{d}_k^{\mathrm{T}}\mathbf{F}_k(\theta)$ by Eq. (44) and Eq. (45)
        end
        Update $\lambda_k^{(j)}(\boldsymbol{\theta}) \in \mathbb{R}^{n_s}$ via Eq. (36)
        Compute the locally iterative error $\varepsilon_{\mathbf{d},j}$
    end
    Update the stochastic solution $\mathbf{u}_k(\theta) = \mathbf{u}_{k-1}(\theta) + \lambda_k(\theta)\mathbf{d}_k$
    Compute the globally iterative error $\varepsilon_{\mathbf{u},k}$
end
Recompute the random vector $\boldsymbol{\Lambda}(\theta) \in \mathbb{R}^k$ via Eq. (37)

which measures the contribution of the k-th couple $\{\lambda_k(\theta), \mathbf{d}_k\}$ to the stochastic solution $\mathbf{u}_k(\theta)$. However, since the random variables $\{\lambda_i(\theta)\}$ are calculated in a sequential way and the value of $\mathbb{E}\left\{\lambda_k^2(\theta)\right\}$ does not keep decreasing in some cases [38], Eq. (47) may be not a good convergence checker in practice. To avoid this issue, we adopt a new error indicator by modifying the random variables $\{\lambda_i(\theta)\}$ in Eq. (47). To this end, we consider the autocorrelation function of the random vector $\mathbf{\Lambda}(\theta)$

$$\mathbf{C}_{\mathbf{\Lambda}\mathbf{\Lambda}} = \mathbb{E}\left\{\mathbf{\Lambda}(\theta)\,\mathbf{\Lambda}(\theta)^{\mathrm{T}}\right\}, \tag{48}$$

which is decomposed into

$$\mathbf{C}_{\mathbf{\Lambda}\mathbf{\Lambda}} = \mathbf{Q}\mathbf{Z}\mathbf{Q}^{\mathrm{T}} \tag{49}$$

by the eigendecomposition, where $\mathbf{Q} \in \mathbb{R}^{k\times k}$ is an orthonormal matrix and $\mathbf{Z}$ is a diagonal matrix. We rewrite the stochastic solution $\mathbf{u}(\theta)$ in Eq. (31) as

$$\mathbf{u}(\theta) = \mathbf{D}\mathbf{Q}\mathbf{Q}^{\mathrm{T}}\mathbf{\Lambda}(\theta) \tag{50}$$

and let a new random vector $\widetilde{\mathbf{\Lambda}}(\theta) = \mathbf{Q}^{\mathrm{T}}\mathbf{\Lambda}(\theta) = \left[\widetilde{\lambda}_1(\theta), \cdots, \widetilde{\lambda}_k(\theta)\right]^{\mathrm{T}} \in \mathbb{R}^k$, the autocorrelation function of which is given by

$$\widetilde{\mathbf{C}}_{\widetilde{\mathbf{\Lambda}}\widetilde{\mathbf{\Lambda}}} = \mathbb{E}\left\{\widetilde{\mathbf{\Lambda}}(\theta)\,\widetilde{\mathbf{\Lambda}}(\theta)^{\mathrm{T}}\right\} = \mathbf{Q}^{\mathrm{T}}\mathbb{E}\left\{\mathbf{\Lambda}(\theta)\,\mathbf{\Lambda}(\theta)^{\mathrm{T}}\right\}\mathbf{Q} = \mathbf{Z}. \tag{51}$$

To improve Eq. (47), we replace the random variables $\{\lambda_i(\theta)\}$ with the new random variables $\left\{\widetilde{\lambda}_i(\theta)\right\}$ and the iterative error $\varepsilon_{\mathbf{u},k}$ thus becomes

$$\varepsilon_{\mathbf{u},k} = \frac{\mathbb{E}\left\{\widetilde{\lambda}_k^2(\theta)\right\}}{\sum_{i=1}^{k}\mathbb{E}\left\{\widetilde{\lambda}_i^2(\theta)\right\}} = \frac{\mathbf{Z}_k}{\mathrm{Tr}(\mathbf{Z})}, \tag{52}$$

where $\mathrm{Tr}(\cdot)$ is the trace operator and $\mathbf{Z}_k$ is the element at position (k,k) of the matrix $\mathbf{Z}$. In this way, the iterative error $\varepsilon_{\mathbf{u},k}$ keeps decreasing as the retained item k increases. It is noted that Eq. (50) does not improve the accuracy of the stochastic solution and just provides a new representation.

6. Numerical examples

We test the proposed method with the aid of four numerical examples. The convergence errors are set as $\varepsilon_{\mathbf{d}} = 1\times10^{-3}$ and $\varepsilon_{\mathbf{u}} = 1\times10^{-8}$ in Algorithm 2. All tests are performed on a laptop

(dual-core, Intel Core i7, 2.40GHz). Without loss of generality, we ignore the dimension of the physical quantities and execute the dimensionless analysis in the first two examples.

6.1. Example 1: elastic equation defined on a random domain

6.1.1. Problem setting

In this example, we consider a two-dimensional elastic problem as discussed in Eq. (20), which is defined on the random domain shown in Fig. 2. The outer bound of the domain is a deterministic square of length 2. The inner boundary $\Gamma_r(\theta)$ of the domain is a hole described by a random elliptic curve that is controlled by four mutually independent random variables $x_c(\theta)$, $y_c(\theta)$, $l_x(\theta)$, $l_y(\theta)$. As depicted in Fig. 2, the location $(x_c(\theta), y_c(\theta))$ of the center point of the ellipse are two uniformly distributed random variables on $[-0.2, 0.2]$ and the major and minor (or minor and major) axes $l_x(\theta)$ and $l_y(\theta)$ are two uniformly distributed random variables on $[0.9, 1.1]$. The boundary conditions are given by the vertical force $f(\overline{x}, \overline{y}) = -1$ on the upper boundary Γ_N and the Dirichlet condition $u(\overline{x}, \overline{y}) = 0$ on the lower boundary Γ_D. The Young's modulus and the Poisson's ratio are 2.10×10^8 and 0.3, respectively.

We choose the mean value of the random boundary $\Gamma_r(\theta)$ as the inner boundary of the reference domain. As shown in Fig. 3a, the reference domain has the same outer boundary as the random domain and its inner boundary is a circle with the center $(0, 0)$ and the diameter 1. The finite ele-

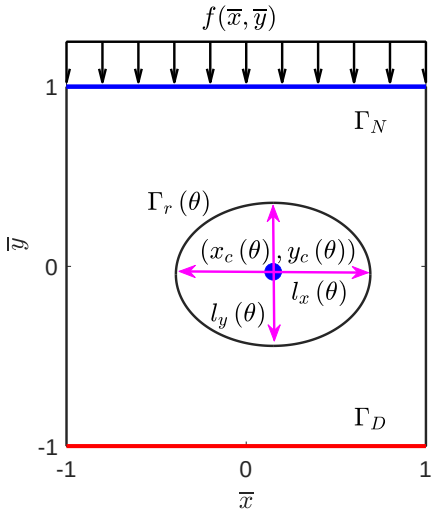
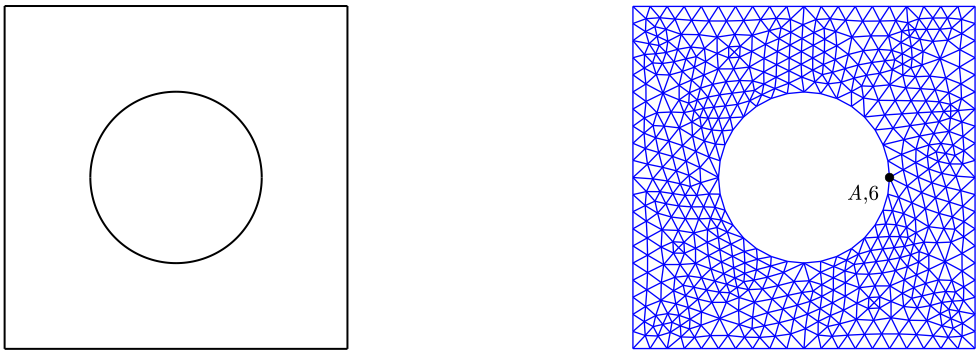


Figure 2: The domain with a random inner boundary.



(a) The reference domain. (b) Finite element mesh of the reference domain.

Figure 3: The reference domain and its finite element mesh.

ment mesh of the reference domain is shown in Fig. 3b, and 624 nodes and 1136 linear triangular elements are included. Following that, we construct the mapping between the reference and the random domains based on the mesh of the reference domain. The random boundary $\Gamma_r(\theta)$ can be represented by random scaling and shift transformations of the circle in the reference domain

$$\begin{bmatrix} x(\theta) \\ y(\theta) \end{bmatrix} = \mathbf{D}(\theta)\,\mathbf{S}(\theta) \begin{bmatrix} \overline{x} \\ \overline{y} \\ 1 \end{bmatrix} \quad \text{on} \quad \Gamma_r(\theta), \tag{53}$$

where the random shift transformation matrix $\mathbf{D}(\theta)$ and the random scaling transformation matrix $\mathbf{S}(\theta)$ are given by

$$\mathbf{D}(\theta) = \begin{bmatrix} 1 & 0 & x_c(\theta) \\ 0 & 1 & y_c(\theta) \end{bmatrix} \in \mathbb{R}^{2\times 3},\ \mathbf{S}(\theta) = \begin{bmatrix} l_x(\theta) & 0 & 0 \\ 0 & l_y(\theta) & 0 \\ 0 & 0 & 1 \end{bmatrix} \in \mathbb{R}^{3\times 3}. \tag{54}$$

6.1.2. Numerical results

By using Algorithm 1, we can solve the random nodal coordinates of the random mesh of the random domain. To solve the displacement $\mathbf{u}(\theta)$ in Eq. (20), we set the sample size as $n_{s,1} = 40$ in step 6 in Algorithm 2. After obtaining the reduced-order matrix $\mathbf{D}$, we reset the sample size as $n_{s,2} = 1 \times 10^4$ in step 28 in Algorithm 2. It is noted that the choice of the sample size in step 6 is still an open problem. In this paper we adopt the sample size $n_{s,1} = 10r$, where r is the number of

447 random variables. A large sample size is suggested if storage requirements are ignored.

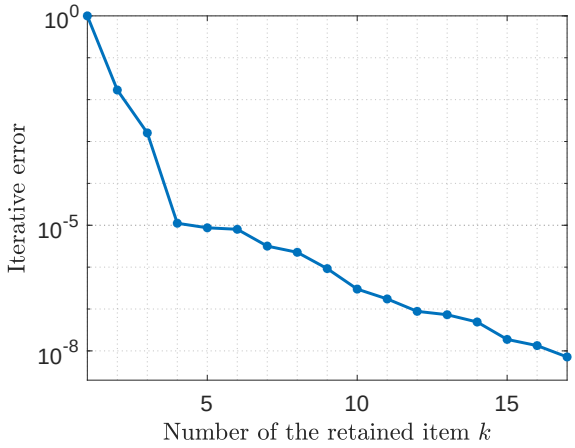


Figure 4: Iterative errors of different numbers of the retained item k calculated by Eq. (52).

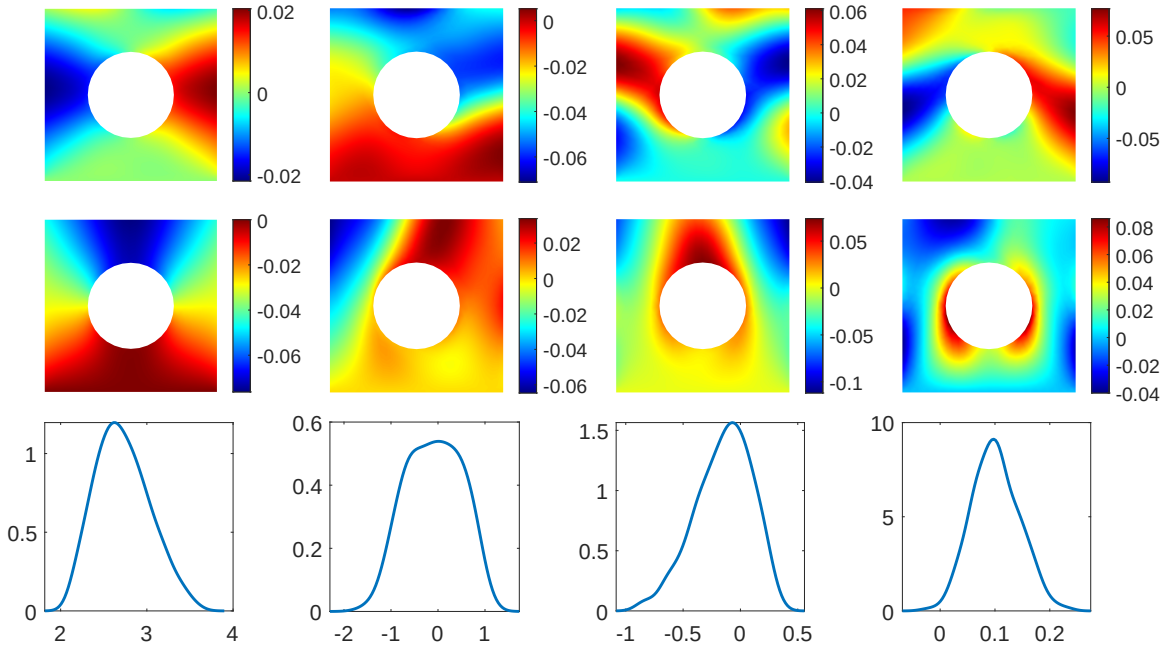


Figure 5: Solutions of the components: the first four solutions $\{\mathbf{d}_i\}_{i=1}^4$ in the $\overline{x}$ direction (the first row), the first four solutions $\{\mathbf{d}_i\}_{i=1}^4$ in the $\overline{y}$ direction (the second row) and PDFs of the first four random variables $\{\lambda_i(\theta)\}_{i=1}^4$ (the third row).

448 Iterative errors $\varepsilon_{\mathbf{u},k}$ in step 26 in Algorithm 2 calculated using Eq. (52) are shown in Fig. 4
449 and retaining 17 terms achieves the specified accuracy, which demonstrates the good convergence

of the proposed method. It is seen that the iterative errors keep decreasing as the retained items increase, which verifies the effectiveness of the proposed error indicator Eq. (52). First four components of the deterministic vectors $\{\mathbf{d}_i\}_{i=1}^{17}$ are depicted in Fig. 5, where $\{\mathbf{d}_i\}_{i=1}^{4}$ in the $\overline{x}$ direction and $\{\mathbf{d}_i\}_{i=1}^{4}$ in the $\overline{y}$ direction are seen from the first and second rows of Fig. 5, respectively. PDFs of the first four components of the recomputed random variables $\{\lambda_i(\theta)\}_{i=1}^{17}$ are shown in the third row of Fig. 5, which are obtained by 1×10^4 random samples. It is noted that the j-th value $\mathbf{d}_{i,j}$ of the vector $\mathbf{d}_i$ is the solution of the j-th node whose coordinate is given by $\left(x_j(\theta), y_j(\theta)\right)$. It is not easy to show a vector on random coordinates, thus the vector $\mathbf{d}_i$ is described in the node-index coordinate system instead of the Cartesian coordinate system in this paper. Although we show the vector $\mathbf{d}_i$ on the reference domain, they can be depicted on all possible domains.

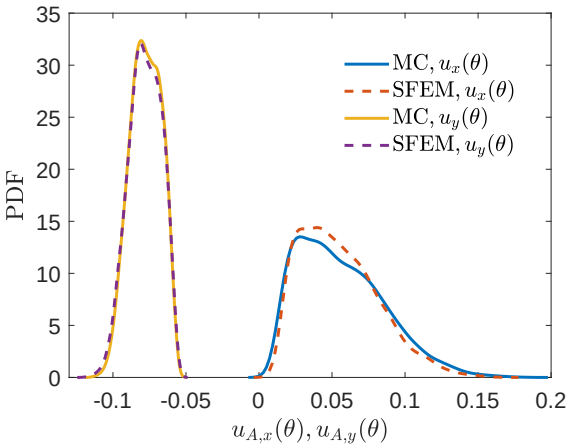


Figure 6: PDFs of stochastic displacements of the point A in the $\overline{x}$ and $\overline{y}$ directions obtained by the proposed method and 1×10^4 MCS.

To test the accuracy of the proposed method, we compare PDFs of stochastic displacements of the point A (its node number is 6 as shown in Fig. 3b) obtained by the proposed method and 1×10^4 MCS. As shown in Fig. 6, PDFs of the stochastic displacements in both $\overline{x}$ and $\overline{y}$ directions have good agreements with MCS, which indicates that the proposed method has comparable accuracy to MCS. It is seen from Fig. 6 that compared to that in the $\overline{y}$ direction, the PDF of the stochastic displacement in the $\overline{x}$ direction is slightly less accurate, but its accuracy is still acceptable in most problems. If a more accurate stochastic solution is required in some cases, we can retain more

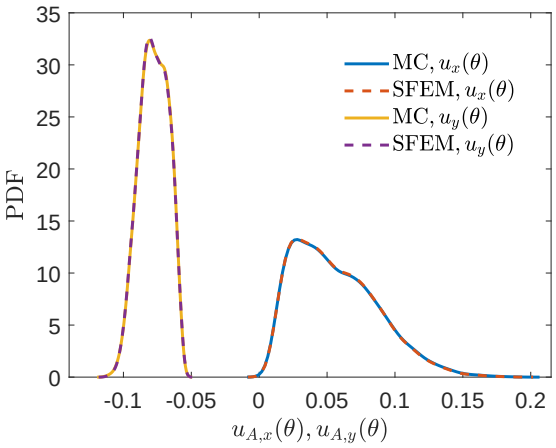


Figure 7: PDFs of stochastic displacements of the point A in the $\overline{x}$ and $\overline{y}$ directions obtained by the proposed method retaining 50 terms and 1×10^4 MCS.

terms for the purpose. PDFs of the stochastic displacements of the point A in $\overline{x}$ and $\overline{y}$ directions obtained by the proposed method retaining 50 terms are depicted in Fig. 7. They have much higher accuracy than that in Fig. 6 and reach very good agreements with MCS. Furthermore, we test the computational efficiency of the proposed method. Computational costs of different implementations are listed in Tab. 1, where GI ($n_{s,1}$, $n_{s,2}$) and LI ($n_{s,1}$, $n_{s,2}$) represent the implementation at global and local levels with sample sizes $n_{s,1}$, $n_{s,2}$ in steps 6 and 28 in Algorithm 2. The solving cost and the recomputing cost are the computational cost from step 5 to 27 and the computational cost of step 28 in Algorithm 2, respectively. Total computational times of three implementations are much less than the cost of 1×10^4 MCS, which demonstrates the high efficiency of the proposed method. It is noted that stochastic solutions obtained by implementations at global and local levels have the same accuracy and only the matrix and the vector are formed differently. Compared to GI (40, 1×10^4), LI (40, 1×10^4) needs a bit more solving costs since more assemblies are performed in steps 9 and 19 in Algorithm 2. As a comparison, we test the case LI (1×10^3, 1×10^4), i.e. $n_{s,1} = 1 \times 10^3$ samples are adopted in step 9. A solution of similar accuracy to the case LI (40, 1×10^4) is obtained, which indicates that $n_{s,1} = 40$ can reach good accuracy in this example. But the case LI (1×10^3, 1×10^4) requires more costs for sample assemblies. The recomputing costs of the three implementations are close since they have the same number of retained items and the

size of the reduced-order stochastic finite element equation (37) is fixed.

Table 1: Computational costs of different implementations.

Methods	GI $(40, 1 \times 10^4)$	LI $(40, 1 \times 10^4)$	LI $(1 \times 10^3, 1 \times 10^4)$	MCS (1×10^4)
Solving costs	324.49	361.74	427.31	
Recomputing costs	28.02	27.53	30.67	
Total costs (second)	352.51	389.27	457.98	2519.66

6.1.3. Postprocessing of the stochastic solution

As discussed above, the vector $\mathbf{d}_i$ is depicted on the node-index coordinates, which is different from the classical FEM. Thus we need to pay extra attention to the postprocessing of stochastic solutions. In practice, to perform the postprocessing of stochastic solutions, we need to combine each realization of stochastic solutions and the corresponding realization of random meshes. For an explanation of this point, let us consider the postprocessing of the stochastic solution under the sample realization $\left[x_c(\theta^*), y_c(\theta^*), l_x(\theta^*), l_y(\theta^*)\right] = [0.1215, -0.1564, 1.0876, 0.9390]$. As shown in Fig. 8 (left), the random mesh is obtained via combining the mesh topology of the reference domain and the random coordinates, where the green part is the reference domain. The realization of the stochastic solution in $\overline{x}$ and $\overline{y}$ directions are then depicted on the random mesh shown in

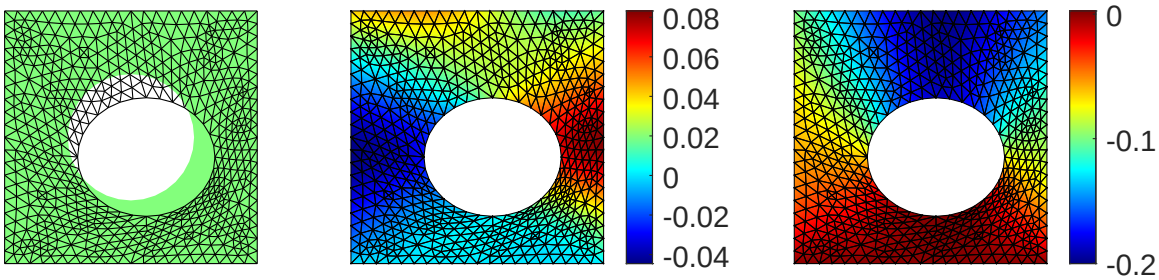


Figure 8: Postprocessing of the stochastic solution of the sample realization $\left[x_c(\theta^*), y_c(\theta^*), l_x(\theta^*), l_y(\theta^*)\right] = [0.1215, -0.1564, 1.0876, 0.9390]$: The reference domain (green part) and the random mesh (left), the solution in the $\overline{x}$ direction (mid) and the solution in the $\overline{y}$ direction (right).

Fig. 8 (mid and right). In this way, the stochastic solution is shown in the Cartesian coordinate system, the same way as the classical FEM postprocessing.

6.2. *Example 2: stochastic elliptic PDE with a random interface*

6.2.1. *Problem setting*

In this example, we consider a stochastic elliptic PDE

$$-\nabla \cdot \left(c\left(x\left(\theta\right), y\left(\theta\right), \theta\right) \nabla u\left(x\left(\theta\right), y\left(\theta\right), \theta\right)\right) = f\left(x\left(\theta\right), y\left(\theta\right)\right) \tag{55}$$

defined on the random domain $\mathcal{D}\left(\theta\right)$ shown in Fig. 9, which has been widely used in many problems with random interfaces [5, 6, 15, 41]. Boundary conditions are given by the Dirichlet condition $u\left(x\left(\theta\right), y\left(\theta\right)\right) = 0$ on Γ_D, the Neumann conditions $\left.\frac{\partial u}{\partial \vec{\mathbf{n}}}\right|_{\Gamma_{N,1}} = 1$, $\left.\frac{\partial u}{\partial \vec{\mathbf{n}}}\right|_{\Gamma_{N,2}} = 2$ and the $f\left(x\left(\theta\right), y\left(\theta\right)\right) = 1$ in $\mathcal{D}\left(\theta\right)$. We consider the discontinuously random coefficients

$$c_1\left(x\left(\theta\right), y\left(\theta\right), \theta\right) = \xi_{c,1}\left(\theta\right) + 1,\ c_2\left(x\left(\theta\right), y\left(\theta\right), \theta\right) = 2\xi_{c,2}\left(\theta\right) + 2, \tag{56}$$

where $\xi_{c,1}\left(\theta\right)$ and $\xi_{c,2}\left(\theta\right)$ are independently uniform random variables on $[0, 1]$.

The random interface $\Gamma_r\left(\theta\right)$ is considered as a Gaussian random field $\Gamma\left(\overline{x}, \theta\right)$ with the mean function $\overline{\Gamma}\left(\overline{x}\right) = 0$ and the covariance function

$$C_{\Gamma\Gamma}\left(\overline{x}_1, \overline{x}_2\right) = \sigma_{\Gamma}^2\left(\min\left(\overline{x}_1, \overline{x}_2\right) - \overline{x}_1\overline{x}_2\right), \tag{57}$$

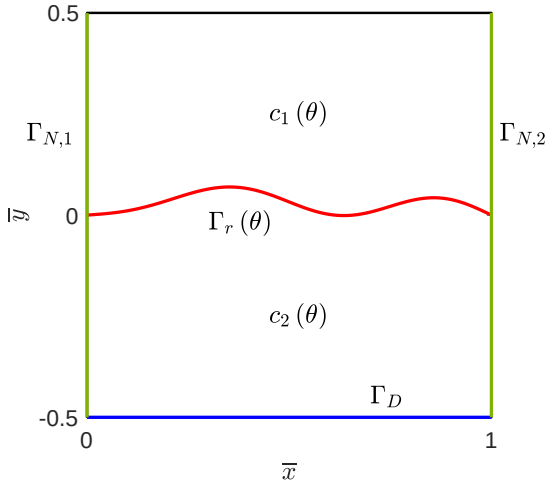
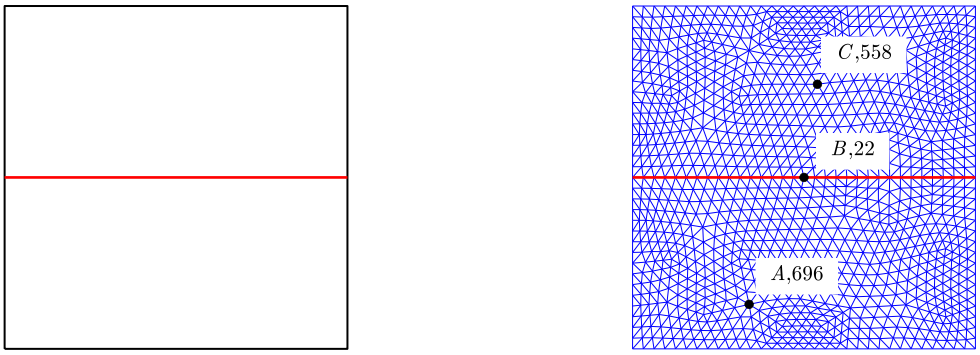


Figure 9: The domain with a random interface.



(a) The reference domain.

(b) Finite element mesh of the reference domain.

Figure 10: The reference domain and its finite element mesh.

where the standard deviation $\sigma_\Gamma = 0.1$. By use of KL expansion [18, 42, 43], the random field $\Gamma(\overline{x}, \theta)$ is approximated as

$$\Gamma(\overline{x}, \theta) = \sigma_\Gamma \sum_{i=1}^{r} \xi_i(\theta) \sqrt{\kappa_i} \Gamma_i(\overline{x}), \tag{58}$$

where $\{\kappa_i\}_{i=1}^r$ and $\{\Gamma_i(\overline{x})\}_{i=1}^r$ are eigenvalues and eigenfunctions of the covariance function $C_{\Gamma\Gamma}(\overline{x}_1, \overline{x}_2)$ and their analytical solutions are

$$\Gamma_i(\overline{x}) = \sqrt{2} \sin(i\pi\overline{x}), \ \kappa_i = (i\pi)^{-2}, \ i = 1, \cdots, r. \tag{59}$$

In the numerical implementation, we limit the sample realization $\Gamma_i(\overline{x})$ in the interval $[-0.25, 0.25]$. To reach this point, non-Gaussian bounded distributions can also be adopted to model the random interface, such as the Beta distribution and the lognormal distribution. Further, the covariance function in Eq. (57) is non-smooth and more random variables are required to achieve a high-accuracy simulation of the random field. Smooth or differentiable covariance functions can be used to reduce the number r of the truncation in Eq. (58) [44, 45].

As shown in Fig. 10a, we choose the mean value $\overline{\Gamma}(\overline{x}) = 0$ of the random interface $\Gamma_r(\theta)$ as the inner interface of the reference domain. As depicted in Fig. 10b, 1217 nodes and 2304 linear triangular elements are generated for the finite element mesh of the reference domain. Based on the finite element mesh, discretized random coordinates of points on the random interface are represented as

$$x(\theta) = \overline{x},\ y(\theta) = \Gamma(\overline{x}, \theta) \quad \text{on} \quad \Gamma_r(\theta). \tag{60}$$

6.2.2. *Numerical results*

A low-dimensional case is considered by truncating KL expansion Eq. (58) at $r = 5$ items, thus 7 (=5+2) random variables are involved in this example. The sample sizes $n_{s,1} = 70$ in step 6 and $n_{s,2} = 1 \times 10^4$ in step 28 in Algorithm 2 are adopted. The iterative errors of different numbers of the retained item k calculated by Eq. (52) are shown in Fig. 11 and 11 retained items converge to the final stochastic solution, which verifies the fast convergence of the proposed method again. First eight deterministic vectors $\{\mathbf{d}_i\}_{i=1}^{8}$ are depicted in the first and second rows of Fig. 12 and PDFs of corresponding first four recomputed random variables $\{\lambda_i(\theta)\}_{i=1}^{4}$ are depicted in the third row. It is seen from Fig. 12 that as the number of retained terms increases, the deterministic vectors concentrate near the random interface, which makes that the stochastic solution near the random interface can be approximated with good accuracy.

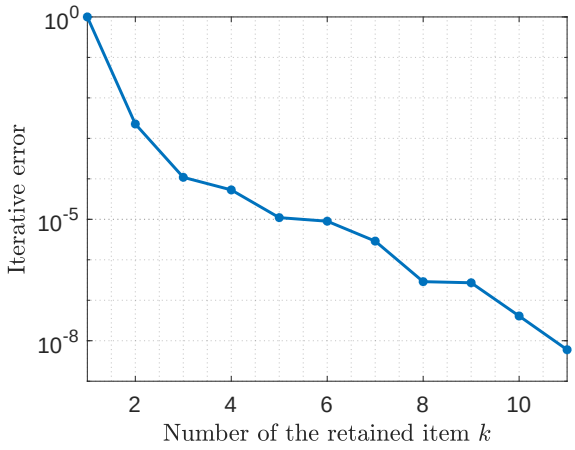


Figure 11: Iterative errors of different numbers of the retained item k calculated by Eq. (52).

As shown in Fig. 10b, we check the accuracy of the stochastic solution by using three characteristic points, i.e. the point A (its node number is 696) in the lower domain, the point B (its node number is 22) on the random interface and the point C (its node number is 558) in the upper domain. PDFs of the solutions at three points are calculated by the proposed method and 1×10^4 MCS and their comparisons are found in Fig. 13. For all three points, PDFs obtained by

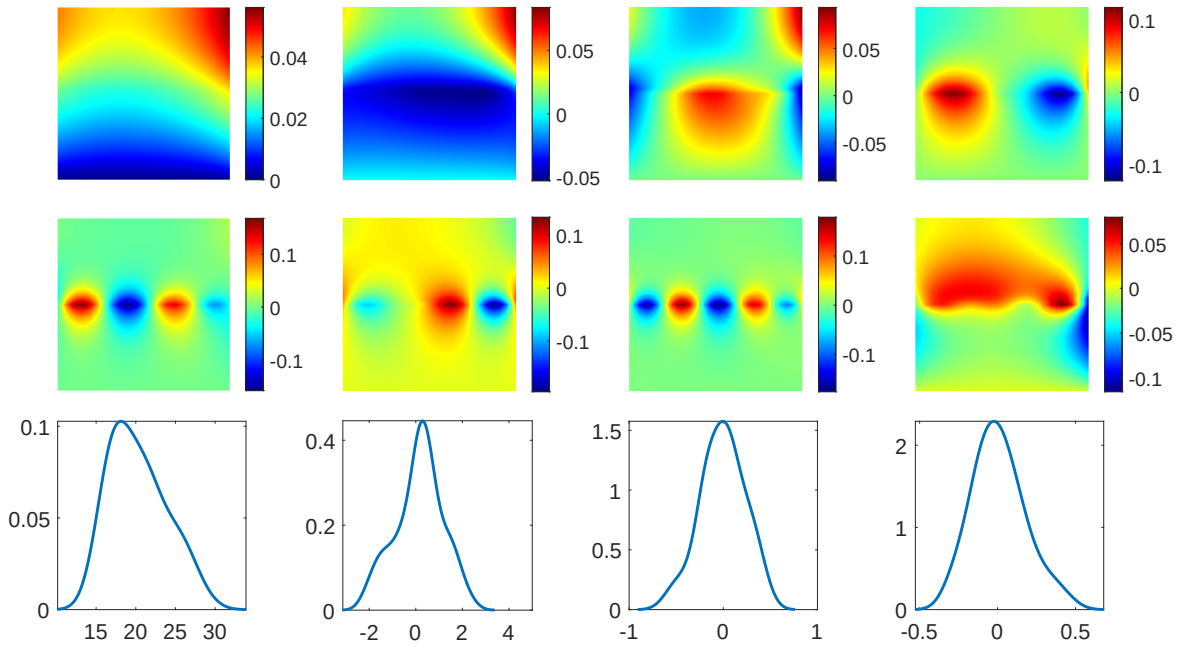


Figure 12: Solutions of the components: the first four solutions $\{\mathbf{d}_i\}_{i=1}^4$ (the first row), the fifth to eighth solutions $\{\mathbf{d}_i\}_{i=5}^8$ (the second row) and PDFs of the first four random variables $\{\lambda_i(\theta)\}_{i=1}^4$ (the third row).

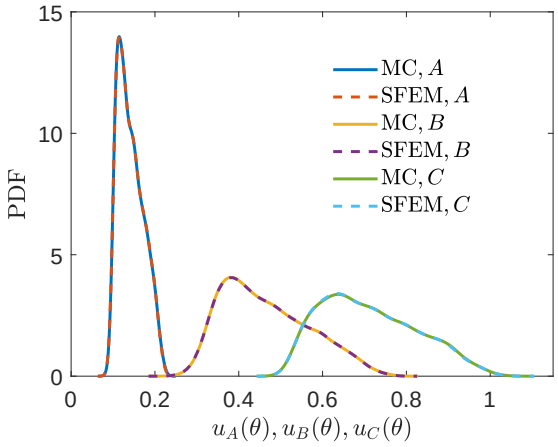


Figure 13: PDFs of the stochastic solutions of points A, B, C obtained by the proposed method and 1×10^4 MCS.

the proposed method have good matches with MCS, which demonstrates the high accuracy of the proposed method in this example.

6.2.3. *Reduced-order model*

Eq. (37) is considered as a reduced-order equation of the original stochastic finite element equation (30) and the matrix $\mathbf{D}$ in Eq. (37) is considered as a set of reduced bases. Thus the proposed method provides a powerful way to generate a reduced-order model of the original stochastic problem. For the explanation of this point, we compare stochastic solutions of full- and reduced-order models of three sample realizations listed in Tab. 2. Three realizations of the random meshes,

Table 2: Three sample realizations of the random variables.

	$\xi_1(\theta)$	$\xi_2(\theta)$	$\xi_3(\theta)$	$\xi_4(\theta)$	$\xi_5(\theta)$	$c_1(\theta)$	$c_2(\theta)$
Sample 1	0.1194	0.0566	−0.5075	−0.8939	−0.8390	1.6139	3.2612
Sample 2	−1.4709	0.3656	1.4673	1.3410	0.8728	1.1345	2.7835
Sample 3	0.5906	−0.8849	−1.5840	0.2493	−1.0121	1.2153	2.4775

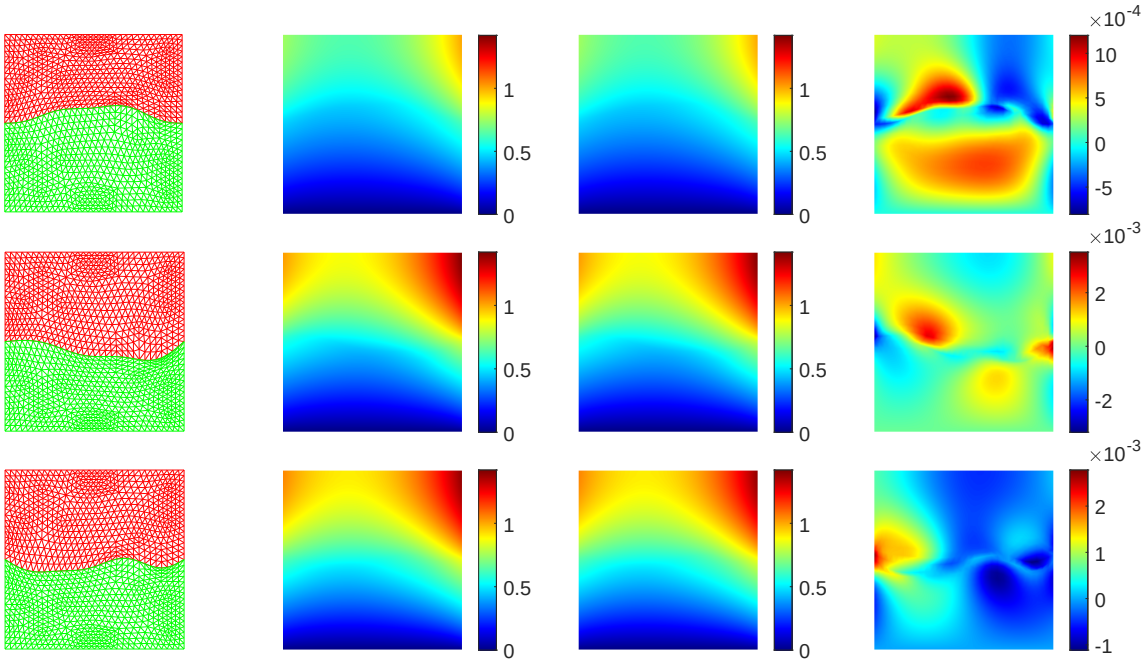


Figure 14: Comparisons between full- and reduced-order models of three sample realizations shown in Tab. 2: the random meshes (the first column), the full-order solutions (the second column), the reduced-order solutions (the third column) and their absolute errors (the fourth column).

the full-order stochastic solutions, the reduced-order stochastic solutions and their absolute errors are depicted in Fig. 14. The maximum of the absolute errors of three realizations is less than 3×10^{-3}, which indicates that the stochastic solutions of reduced-order models have comparable accuracy to the full-order models and the proposed method can provide an accurate reduced-order model of the original problem.

6.2.4. High-dimensional stochastic problems

In this section, we show that the proposed method can be applied to high-dimensional stochastic problems without any modification. The truncation of KL expansion Eq. (58) is set as $r = 50$. A total of 52 random variables are considered in this example. The sample sizes $n_{s,1} = 520$ in step 6 and $n_{s,2} = 1 \times 10^4$ in step 28 in Algorithm 2 are used in this case. Corresponding iterative errors $\varepsilon_{\mathbf{u},k}$ in step 26 in Algorithm 2 calculated using Eq. (52) are depicted in Fig. 15 and 28 items are retained to achieve the convergence error. Compared to the low-dimensional case, the high-dimensional case requires more retained items to capture the high-accuracy stochastic solution. The PDFs of the stochastic solutions of the point B (shown in Fig. 10b) obtained by the proposed method and MCS are compared in Fig. 16. The PDF obtained by the proposed method is in good agreement with MCS, which indicates that the proposed method has good accuracy even for high-dimensional stochastic problems. The PDF shown in Fig. 16 is close to that of the low-dimensional

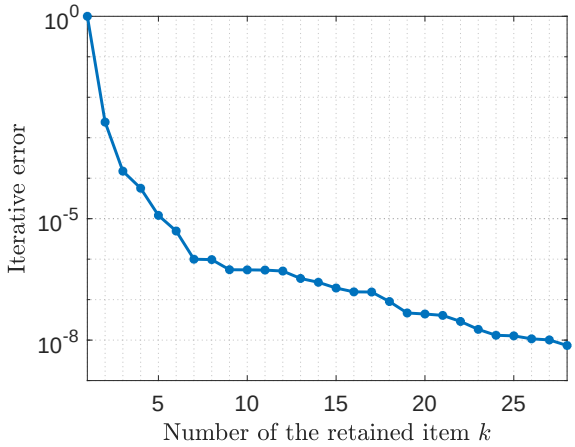


Figure 15: Iterative errors of different numbers of the retained item k calculated by Eq. (52) for the stochastic dimension 52.

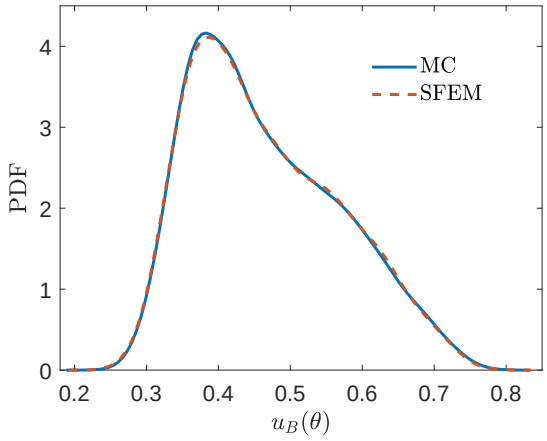


Figure 16: PDFs of the stochastic solutions of the point B when the stochastic dimension is 52.

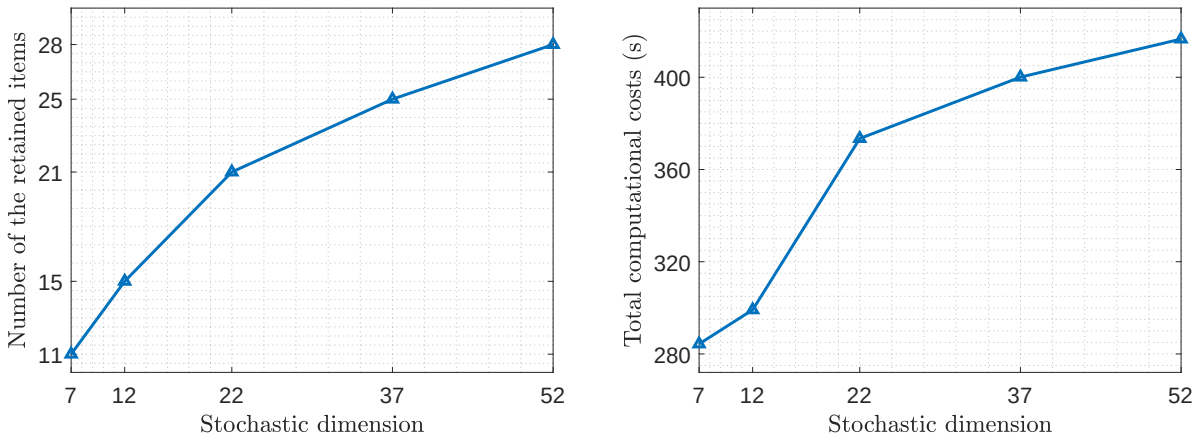
case shown in Fig. 13 since the last few terms expanded by KL expansion Eq. (58) may have less contributions to the quantities of interest. It is noted that the proposed method solves both low- and high-dimensional stochastic problems using a unified frame and no modification for high-dimensional cases is needed. Only more retained terms are required to capture the large variability of the stochastic solution if there are large uncertainties caused by high-dimensional expansions.

To show the computational efficiency of the proposed method, the computational costs of the stochastic dimensions 7 and 52 are listed in Tab. 3. As the stochastic dimension increases, the total computational cost of the proposed method increases slowly and the recomputing cost slightly increases since the number of the retained items increases. Compared to MCS, the proposed method can solve high-dimensional stochastic problems with low computational costs, thus it avoids the curse of dimensionality successfully.

Table 3: Computational costs of the stochastic dimensions 7 and 52.

Dimension	7 (=5+2)		52 (=50+2)	
Method	SFEM	MCS	SFEM	MCS
Solving costs	264.76		382.75	
Recomputing costs	19.56		33.83	
Total costs (second)	284.32	2124.76	416.58	2359.72

The computational efficiency of the proposed method with respect to different stochastic dimensions are further studied. The truncation r in Eq. (58) is set as 5, 10, 20, 35, 50, respectively, and corresponding total dimensions are 7, 12, 22, 37, 52. We execute the iterations corresponding to different stochastic dimensions until the specified convergence error $\varepsilon_{\mathbf{u},k} = 1 \times 10^{-8}$ is achieved. The number of retained terms k and corresponding total computational costs are respectively shown in Fig. 17a and Fig. 17b, which demonstrates that the number of retained items increase slightly as the stochastic dimension increases. Corresponding computational costs also does not increase dramatically with the stochastic dimension. For the low dimensions (not greater than 12) and the high dimensions (not less than 22), the computational cost is almost proportional to the stochastic dimension. The cost jumping between the dimensions 12 and 22 may be caused by the large variability induced by the increased dimension.



(a) Numbers of the retained items for different dimensions. (b) Total computational costs for different dimensions.

Figure 17: Numbers of the retained items (left) and corresponding computational costs (right) for different stochastic dimensions.

6.3. Example 3: study case from orthodontics with random material properties and random geometry

6.3.1. Problem setting

In this example, we consider a typical case of orthodontics that involves the human tooth shown in Fig. 18 (left), which is from [46]. It is only a simplified model of the human tooth and more

realistic models can be found in [47]. The Young's modulus of multi-layer material of the tooth are modeled as Gaussian random variables due to individual differences in patients. The Poisson's ratio and the mean values of Young's modulus are listed in Tab. 4. Standard deviations of all Young's modulus are 0.1 times the mean values. In the numerical implementation, to ensure that the Young's modulus is positive, random samples of the Young's modulus less than 1×10^{-3} are dropped out, thus they are considered as truncated Gaussian random variables in practice. But it is noted that the truncation usually results in a loss of coercivity in the bilinear form of the finite element approximation [40, 48, 49]. Better stochastic modeling of the material properties should further considered to avoid this issue. The force induced by the orthodontic appliance is applied to the model in the horizontal direction and its magnitude is 1N.

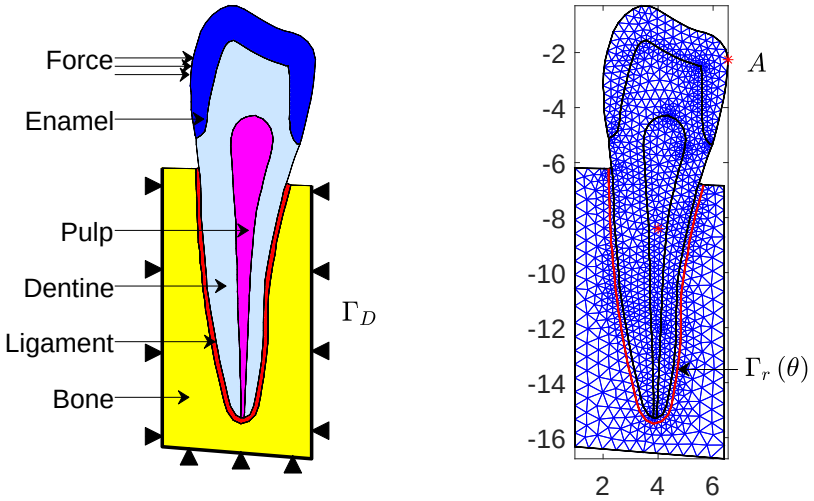


Figure 18: The geometry of tooth model (left) and its finite element mesh (right).

Table 4: Mean values of Young's modulus and Poisson's ratio.

	Bone	Ligament	Dentine	Pulp	Enamel
Young's modulus	12 Gpa	100 Mpa	18 Gpa	4 Mpa	90 Gpa
Poisson's ratio	0.2	0.3	0.2	0.35	0.2

In order to formulate a good orthodontics plan, we need to predict the deformation of the tooth

after each orthodontic appliance according to the randomness of material properties and current orthodontics outcomes (i.e. the current position of the tooth). The current position of the tooth is also uncertain due to the randomness of material properties. As shown in Fig. 18 (right), the random position is modeled by the rotation angle $\alpha(\theta)$ of the tooth around a point (the red point in Fig. 18 (right)), where $\alpha(\theta)$ is a uniform random variable on $[-2^\circ, 2^\circ]$. Thus the interface $\Gamma_r(\theta)$ between the bone and the ligament is considered as a random interface depending on the position of the tooth. The quantity of interest during orthodontics is the horizontal displacement of the tooth. In this paper, we focus on the horizontal stochastic displacement $u_{A,x}(\theta)$ of the point A shown in Fig. 18 (right). We adopt the two-dimensional elastic equation discussed in Eq. (20). The Dirichlet boundary condition is given by $u(\theta) = 0$ on the boundary Γ_D (shown in Fig. 18 (left)). The reference domain and its mesh are depicted in Fig. 18 (right), which has the same boundary Γ_D as the random domain and includes 1394 nodes, 2670 linear triangular elements and 2788 degrees of freedom in total.

6.3.2. Numerical results

In this example, the sample sizes $n_{s,1} = 60$ in step 6 and $n_{s,2} = 1 \times 10^4$ in step 28 in Algorithm 2 are used for the implementation. For different numbers of the retained item, iterative errors $\varepsilon_{\mathbf{u},k}$ in step 26 in Algorithm 2 calculated by Eq. (52) are shown in Fig. 19 and only six items are retained, which achieves a high-accuracy solution with fewer items due to the small variability of

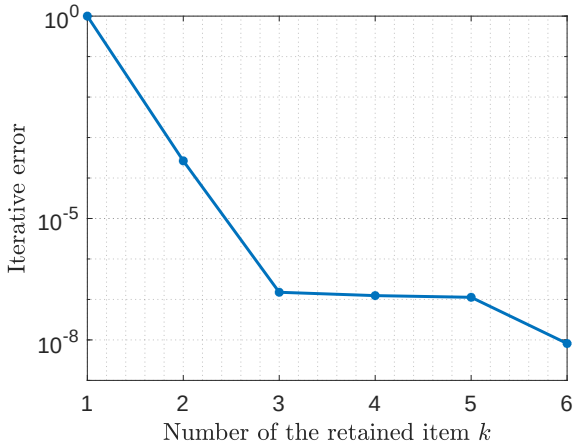


Figure 19: Iterative errors of different numbers of the retained item k calculated by Eq. (52).

the random geometry. The PDFs of the horizontal stochastic displacement $u_{A,x}(\theta)$ of the point A obtained by the proposed method and 1×10^4 MCS are compared in Fig. 20 and they are very consistent. Compared to the computational time 2948.69s for MCS, there are only 192.84s for executing the proposed method, including the solving time 189.04s and the recomputing time 3.80s, which significantly saves computational effort.

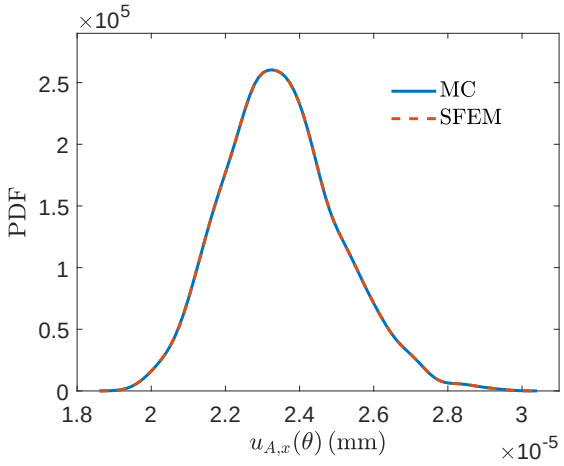


Figure 20: PDFs of the horizontal stochastic displacement $u_{A,x}(\theta)$ of the point A obtained by the proposed method and 1×10^4 MCS.

6.4. Example 4: nonlinear stochastic heat equation defined on a random domain

6.4.1. Problem setting

In this example, to test the applicability of the proposed method to nonlinear PDEs on random domains, we consider a nonlinear stochastic heat equation

$$\begin{cases} -\nabla \cdot \left(c\left(T,\theta\right)\nabla T\left(x\left(\theta\right),y\left(\theta\right),\theta\right)\right) = 0 & \text{in} \quad \mathcal{D}\left(\theta\right) \\ -\overrightarrow{\mathbf{n}} \cdot \nabla T\left(x\left(\theta\right),y\left(\theta\right),\theta\right) = f\left(x\left(\theta\right),y\left(\theta\right)\right) & \text{on} \quad \Gamma_r\left(\theta\right) \end{cases} \tag{61}$$

defined on the random domain $\mathcal{D}(\theta)$ as shown in Fig. 21, where the nonlinear stochastic coefficient $c(T,\theta)$ is given by $c(T,\theta) = \sqrt{10 + T^2(\theta)}$, the heat flux $f(x(\theta), y(\theta)) = 100\text{W/m}$ is applied to the random boundary $\Gamma_r(\theta)$, the temperature $T = 0$ on upper and lower boundaries (red lines shown in Fig. 21), $\overrightarrow{\mathbf{n}}$ is the outward normal. It is noted that the applied position of $f(x(\theta), y(\theta))$ is also random due to the randomness of the boundary $\Gamma_r(\theta)$, and the coefficient $c(T,\theta)$ is considered

as a random field defined on the random domain due to the spatial dependence of the stochastic solution $T(x(\theta), y(\theta), \theta)$. The geometric uncertainties, the reference domain and the mesh are the same as those in Fig. 2 in Example 6.1. Further, to consider the correlations of the geometric uncertainties, the Pearson correlation coefficient of any two random variables in $x_c(\theta)$, $y_c(\theta)$, $l_x(\theta)$ and $l_y(\theta)$ (see from Example 6.1) is 0.2.

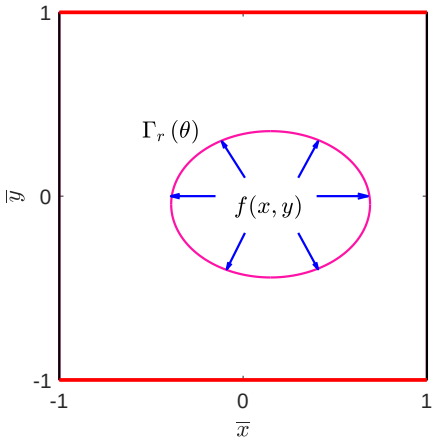


Figure 21: The domain with a random inner boundary and a moving force.

For the numerical implementation, the fixed-point iteration is adopted to deal with the nonlinearity in Eq. (61) [36, 50]. Similar to Eq. (32), we can get the following linearized stochastic finite element equation about the unknown couple $\{\lambda_k(\theta), \mathbf{d}_k\}$ based on the previous stochastic solution approximation $\mathbf{T}_{k-1}(\theta)$

$$\mathbf{K}(\mathbf{T}_{k-1}(\theta))\,\lambda_k(\theta)\,\mathbf{d}_k = \mathbf{F}(\theta) - \mathbf{K}(\mathbf{T}_{k-1}(\theta))\,\mathbf{T}_{k-1}(\theta), \tag{62}$$

where $\mathbf{K}(\mathbf{T}_{k-1}(\theta))$ is the linearized stochastic matrix assembled using the previous approximation $\mathbf{T}_{k-1}(\theta)$, $\mathbf{F}(\theta)$ is the stochastic vector related to the heat flux $f(x(\theta), y(\theta))$. In this example, $\mathbf{T}_0(\theta) = 0$ is set to initialize the above iteration.

6.4.2. Numerical results

Algorithm 2 still can be used to solve Eq. (62). But before performing each inner loop, the stochastic matrix $\mathbf{K}(\mathbf{T}_{k-1}(\theta))$ is updated and reassembled based on $\mathbf{T}_{k-1}(\theta)$. The sample sizes $n_{s,1} = 40$ in step 6 in Algorithm 2 and $n_{s,2} = 1 \times 10^4$ in step 28 are used. To well capture the

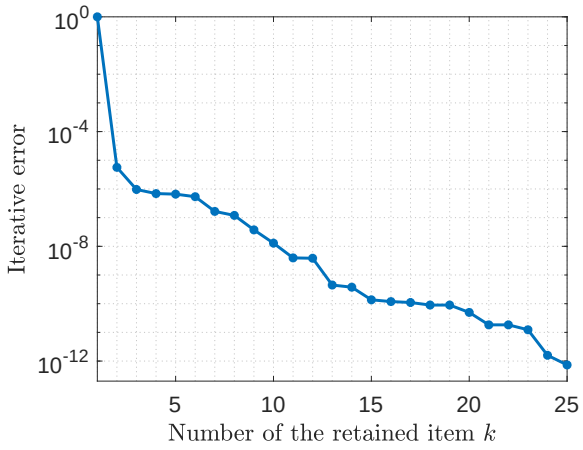


Figure 22: Iterative errors of different numbers of the retained item k calculated by Eq. (52).

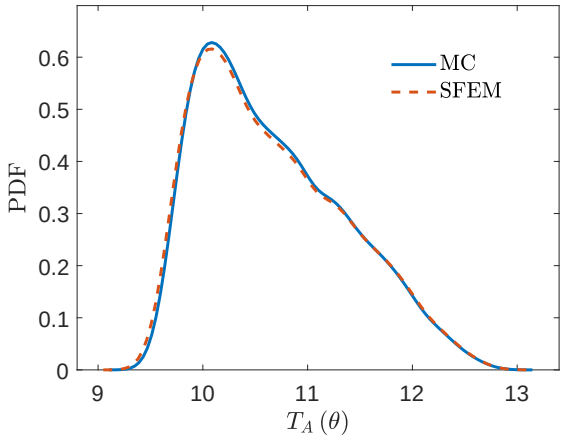


Figure 23: PDFs of the horizontal stochastic displacement $u_{A,x}(\theta)$ of the point A obtained by the proposed method and 1×10^4 MCS.

nonlinearity, the stopping criterion $\varepsilon_{\mathbf{u}}$ in step 5 is set as 1×10^{-12} in this case. Iterative errors $\varepsilon_{\mathbf{u},k}$ corresponding to different numbers of the retained item is still calculated by Eq. (52) and shown in Fig. 22, which demonstrates that the convergence of the proposed method for nonlinear problems is still good enough. PDFs of the stochastic solution $T_A(\theta)$ of the point A (see Fig. 3b) obtained by the proposed SFEM and 1×10^4 MCS are plotted in Fig. 23. They are still in good agreement. Regarding the computational efficiency, MCS costs 3324.56s, while the proposed SFEM takes 351.57s, including the solving time 289.64s and the recomputing time 61.93s, which indicates that the proposed method is still efficient for nonlinear cases.

7. Conclusions

In this paper we develop an efficient stochastic finite element method for solving deterministic/stochastic PDEs defined on random domains and illustrate its effectiveness using four numerical examples. A reference domain is used to generate a mesh topology and to represent random nodal coordinates of the random domain. Random meshes of the random domain are obtained by combining the mesh topology of the reference domain and the random nodal coordinates of the random domain. In this way, the proposed method still solves the PDEs on the random domain instead of the reference domain, which decouples the differential operator of the PDE and the random domain and can be implemented via existing FEM assembly codes. The proposed method can be applied to high-dimensional stochastic problems without any modification and avoids the curse of dimensionality to a great extent, which has been demonstrated by an example of up to 52 stochastic dimensions. Also, a nonlinear heat equation defined on a random domain has been used to verify the applicability of the proposed method to nonlinear PDEs on random domains. However, it is noted that the non-intrusive assembly of the stochastic stiffness matrix costs a lot of storage memory compared to intrusive ways, thus it is attractive to develop an intrusive assembly of stochastic stiffness matrices.

Acknowledgments

The authors are grateful to the Alexander von Humboldt Foundation and the International Research Training Group 2657 (IRTG 2657) funded by the German Research Foundation (DFG) (Grant number 433082294).

References

[1] O. Le Maître, O. M. Knio, Spectral methods for uncertainty quantification: with applications to computational fluid dynamics, Springer Science & Business Media, 2010.

[2] H. Dai, R. Zhang, M. Beer, A new method for stochastic analysis of structures under limited observations, Mechanical Systems and Signal Processing 185 (2023) 109730.

[3] S. Chen, W. Chen, A new level-set based approach to shape and topology optimization under geometric uncertainty, Structural and Multidisciplinary Optimization 44 (2011) 1–18.

[4] W. Zhang, Z. Kang, Robust shape and topology optimization considering geometric uncertainties with stochastic level set perturbation, International Journal for Numerical Methods in Engineering 110 (2017) 31–56.

[5] A. Nouy, A. Clement, eXtended Stochastic Finite Element Method for the numerical simulation of heterogeneous materials with random material interfaces, International Journal for Numerical Methods in Engineering 83 (2010) 1312–1344.

[6] C. Lang, A. Doostan, K. Maute, Extended stochastic FEM for diffusion problems with uncertain material interfaces, Computational Mechanics 51 (2013) 1031–1049.

[7] M. Diez, E. F. Campana, F. Stern, Design-space dimensionality reduction in shape optimization by Karhunen–Loève expansion, Computer Methods in Applied Mechanics and Engineering 283 (2015) 1525–1544.

[8] S. Badia, J. Hampton, J. Principe, Embedded multilevel monte carlo for uncertainty quantification in random domains, International Journal for Uncertainty Quantification 11 (2021).

[9] D. Xiu, D. M. Tartakovsky, Numerical methods for differential equations in random domains, SIAM Journal on Scientific Computing 28 (2006) 1167–1185.

[10] C. Canuto, T. Kozubek, A fictitious domain approach to the numerical solution of PDEs in stochastic domains, Numerische Mathematik 107 (2007) 257–293.

[11] A. Nouy, M. Chevreuil, E. Safatly, Fictitious domain method and separated representations for the solution of boundary value problems on uncertain parameterized domains, Computer Methods in Applied Mechanics and Engineering 200 (2011) 3066–3082.

[12] M. Papadrakakis, V. Papadopoulos, Robust and efficient methods for stochastic finite element analysis using Monte Carlo simulation, Computer Methods in Applied Mechanics Engineering 134 (1996) 325–340.

[13] J. N. Fuhg, A. Fau, U. Nackenhorst, State-of-the-art and comparative review of adaptive sampling methods for kriging, Archives of Computational Methods in Engineering (2020) 1–59.

[14] G. Lin, A. M. Tartakovsky, D. M. Tartakovsky, Uncertainty quantification via random domain decomposition and probabilistic collocation on sparse grids, Journal of Computational Physics 229 (2010) 6995–7012.

[15] Y. N. Lazarev, P. Petrov, D. M. Tartakovsky, Interface dynamics in randomly heterogeneous porous media, Advances in Water Resources 28 (2005) 393–403.

[16] P. S. Mohan, P. B. Nair, A. J. Keane, Stochastic projection schemes for deterministic linear elliptic partial differential equations on random domains, International Journal for Numerical Methods in Engineering 85 (2011) 874–895.

[17] V. D. Liseikin, Grid generation methods, Springer, 2017.

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

[19] D. Xiu, Numerical methods for stochastic computations: a spectral method approach, Princeton University Press, 2010.

[20] J. E. Castrillón-Candás, F. Nobile, R. F. Tempone, Analytic regularity and collocation approximation for elliptic

PDEs with random domain deformations, Computers & Mathematics with Applications 71 (2016) 1173–1197.

[21] R. Glowinski, T.-W. Pan, J. Periaux, A fictitious domain method for Dirichlet problem and applications, Computer Methods in Applied Mechanics and Engineering 111 (1994) 283–303.

[22] A. Nouy, F. Schoefs, N. Moës, X-SFEM, a computational technique based on X-FEM to deal with random shapes, European Journal of Computational Mechanics/Revue Européenne de Mécanique Numérique 16 (2007) 277–293.

[23] A. Nouy, A. Clement, F. Schoefs, N. Moës, An extended stochastic finite element method for solving stochastic partial differential equations on random domains, Computer Methods in Applied Mechanics and Engineering 197 (2008) 4663–4682.

[24] A. R. Khoei, Extended finite element method: theory and applications, John Wiley & Sons, 2014.

[25] D. Xiu, G. E. Karniadakis, The Wiener–Askey polynomial chaos for stochastic dierential equations, SIAM Journal on Scientific Computing 24 (2002) 619–644.

[26] I. Babuška, J. Chleboun, Effects of uncertainties in the domain on the solution of Neumann boundary value problems in two spatial dimensions, Mathematics of Computation 71 (2002) 1339–1370.

[27] I. Babuška, J. Chleboun, Effects of uncertainties in the domain on the solution of Dirichlet boundary value problems, Numerische Mathematik 93 (2003) 583–610.

[28] A. Kundu, S. Adhikari, M. Friswell, Stochastic finite elements of discretely parameterized random systems on domains with boundary uncertainty, International Journal for Numerical Methods in Engineering 100 (2014) 183–221.

[29] H. Harbrecht, M. Peters, M. Siebenmorgen, Analysis of the domain mapping method for elliptic diffusion problems on random domains, Numerische Mathematik 134 (2016) 823–856.

[30] J. E. Castrillón-Candás, F. Nobile, R. F. Tempone, A hybrid collocation-perturbation approach for PDEs with random domains, Advances in Computational Mathematics 47 (2021) 1–35.

[31] A. Ammar, A. Huerta, F. Chinesta, E. Cueto, A. Leygue, Parametric solutions involving geometry: a step towards efficient shape optimization, Computer Methods in Applied Mechanics and Engineering 268 (2014) 178–193.

[32] A. Courard, D. Néron, P. Ladevèze, L. Ballere, Integration of PGD-virtual charts into an engineering design process, Computational Mechanics 57 (2016) 637–651.

[33] S. M. Shontz, S. A. Vavasis, A mesh warping algorithm based on weighted Laplacian smoothing., in: IMR, Citeseer, 2003, pp. 147–158.

[34] M. Selim, R. Koomullil, Mesh deformation approaches – a survey, Journal of Physical Mathematics 7 (2016) 1–9.

[35] T. J. Hughes, The finite element method: linear static and dynamic finite element analysis, Courier Corporation, 2012.

[36] K.-J. Bathe, Finite element procedures, Klaus-Jurgen Bathe, 2006.

[37] Z. Zheng, H. Dai, Structural stochastic responses determination via a sample-based stochastic finite element method, Computer Methods in Applied Mechanics and Engineering 381 (2021) 113824.

[38] Z. Zheng, M. Beer, H. Dai, U. Nackenhorst, A weak-intrusive stochastic finite element method for stochastic structural dynamics analysis, Computer Methods in Applied Mechanics and Engineering 399 (2022) 115360.

[39] F. Chinesta, R. Keunings, A. Leygue, The proper generalized decomposition for advanced numerical simulations: a primer, Springer Science & Business Media, 2013.

[40] A. Nouy, A generalized spectral decomposition technique to solve a class of linear stochastic partial differential equations, Computer Methods in Applied Mechanics and Engineering 196 (2007) 4521–4537.

[41] R. Ghanem, W. Brzakala, Stochastic finite-element analysis of soil layers with random interface, Journal of Engineering Mechanics 122 (1996) 361–369.

[42] Z. Zheng, H. Dai, Simulation of multi-dimensional random fields by Karhunen–Loève expansion, Computer Methods in Applied Mechanics and Engineering 324 (2017) 221–247.

[43] Z. Zheng, H. Dai, Y. Wang, W. Wang, A sample-based iterative scheme for simulating non-stationary non-Gaussian stochastic processes, Mechanical Systems and Signal Processing 151 (2021) 107420.

[44] P. D. Spanos, M. Beer, J. Red-Horse, Karhunen–Loève expansion of stochastic processes with a modified exponential covariance kernel, Journal of Engineering Mechanics 133 (2007) 773–779.

[45] M. G. Faes, M. Broggi, P. D. Spanos, M. Beer, Elucidating appealing features of differentiable auto-correlation functions: A study on the modified exponential kernel, Probabilistic Engineering Mechanics (2022) 103269.

[46] I. González, M. A. Valdebenito, J. Correa, H. A. Jensen, Calculation of second order statistics of uncertain linear systems applying reduced order models, Reliability Engineering & System Safety 190 (2019) 106514.

[47] E. Li, J. Chen, Z. Zhang, J. Fang, G. Liu, Q. Li, Smoothed finite element method for analysis of multi-layered systems – Applications in biomaterials, Computers & Structures 168 (2016) 16–29.

[48] I. Babuška, P. Chatzipantelidis, On solving elliptic stochastic partial differential equations, Computer Methods in Applied Mechanics and Engineering 191 (2002) 4093–4122.

[49] P. Frauenfelder, C. Schwab, R. A. Todor, Finite elements for elliptic problems with stochastic coefficients, Computer Methods in Applied Mechanics and Engineering 194 (2005) 205–228.

[50] J. N. Reddy, An introduction to nonlinear finite element analysis: with applications to heat transfer, fluid mechanics, and solid mechanics, OUP Oxford, 2014.