Uncertainty analysis of large-scale systems using domain decomposition

By D. Ghosh, C. Farhat AND P. Avery

1. Motivation and objectives

A realistic analysis and design of physical systems must take into account uncertainties contributed by various sources such as manufacturing variability, insufficient data, unknown physics and aging. In a probabilistic framework these uncertainties are first modeled as random quantities with assigned probability distributions. The probabilistic nature of the response of the uncertain system under deterministic or random loading is then estimated using various available methods such as stochastic finite element methods (SFEM).

Let $(\Omega, \mathcal{F}, P)$ denote the underlying probability space of uncertainty, where Ω denotes the set of elementary events, $\theta \in \Omega$, $\mathcal{F}$ denotes a σ-algebra on that event set, and P denotes the probability measure. Let $\mathcal{D}$ denote the physical domain of the system, and $\boldsymbol{x} \in \mathcal{D}$. Consider the following elliptic equation to hold almost everywhere (a.e.) on Ω

$$\begin{aligned} \mathcal{L}(u(\boldsymbol{x},\theta)) &= f(\boldsymbol{x},\theta) \quad \boldsymbol{x} \in \mathcal{D}, \\ u(\boldsymbol{x},\theta) &= u_b(\boldsymbol{x},\theta) \quad \boldsymbol{x} \in \partial\mathcal{D}, \end{aligned} \tag{1.1}$$

where $u(\boldsymbol{x},\theta), f(\boldsymbol{x},\theta) : \mathcal{D} \times \Omega \to \mathbb{R}$. The uncertain parameters in this equation are embedded in the coefficients, and sometimes in the boundary conditions. The external forcing function f also can be random. In a general probabilistic framework, some of these random parameters and sometimes the force are modeled as random variables and some as random processes. Since the random processes are infinite dimensional objects, for computational purposes they are further discretized using a suitable basis set in the space of square-integrable random variables $L_2(\Omega)$. For example, when the covariance of a process is known, then the Karhunen-Loève (KL) expansion (Ghanem & Spanos 2003) can be used for such discretization. The set of all these random variables completely characterizes the uncertainty in the underlying system. These random variables may not be completely independent of each other, and their joint distribution will be non-Gaussian in general. This set can be transformed to a function of a set of independent standard normal (Gaussian) variables $\{\xi_i(\theta)\}_{i=1}^{i=m}$, also denoted by the m-dimensional vector $\boldsymbol{\xi}$, using various techniques (Ghanem & Doostan 2006; Das *et al.* 2006). In literature, m is often referred to as the stochastic dimension of the problem. Since u, the response of the system, will also be a function of $\boldsymbol{\xi}$, therefore $u(\boldsymbol{x},\theta)$ can now be denoted as $u(\boldsymbol{x},\boldsymbol{\xi})$.

The response of the system is next represented in a tensor product space as $u(\boldsymbol{x},\boldsymbol{\xi}) \in V(\mathcal{D}) \otimes L^2(\Omega)$. Here $V(\mathcal{D})$ may be a space of deterministic finite element bases, for example. Similarly a set of orthogonal bases are chosen in $L^2(\Omega)$. Since the basic random variables $\boldsymbol{\xi}$ are Gaussian, the natural choice of a set of orthogonal polynomials in these variables becomes the set of Hermite polynomials, and the resulting representation is called the polynomial chaos expansion (PCE) (Ghanem & Spanos 2003). Let us assume that the physical domain is discretized using an n degrees of freedom (dof) finite element

model, and the response is represented using a P-term PCE. For this problem, in order to estimate the chaos coefficients when a Galerkin projection is used to minimize the residual of the governing equation (Ghanem & Spanos 2003; Babŭska *et al.* 2005), the stochastic problem is translated into the following system of linear deterministic equations

$$\mathbf{K}\boldsymbol{u} = \boldsymbol{f}, \quad \mathbf{K} \in \mathbb{R}^{nP \times nP}, \quad \boldsymbol{u}, \boldsymbol{f} \in \mathbb{R}^{nP} \ . \tag{1.2}$$

Solving this large system of equations has drawn significant attention (Ghanem & Kruger 1996; Pellissetti & Ghanem 2000; Keese & Matthies 2005; Sarkar *et al.* 2006), however, it still remains as a major challenge in quantifying uncertainty in large-scale systems. This system is block-sparse, and usually is solved using iterative solvers such as conjugate gradient (CG). Like most large linear systems, a properly chosen preconditioner could help the convergence of the iterative solver. In this work a Block-Jacobi type preconditioner is used for this purpose, and is found to be very effective. To apply this preconditioner an $(n \times n)$ system needs to be solved repeatedly, for different multiple right-hand sides. To this end, a domain decomposition method called finite element tearing and interconnecting - dual primal (FETI-DP) is used as the solver. In particular, a variant of FETI-DP that is adapted to solve for multiple right-hand sides more efficiently is used. The proposed method is successfully tested by a numerical study. It should be noted that the presented framework is equally valid for other bases than the Hermite ones such as other types of polynomials (Xiu & Karniadakis 2003) or wavelet bases (LeMaitre *et al.* 2004).

It is noted that an overlapping additive Schwarz domain decomposition method with the similar multiple right-hand sides adaptation is used to apply a preconditioner in a recent paper (Jin *et al.* 2007); however, there the uncertainty model remains only as a special case of the general model considered. In another recent work on using domain decomposition in SFEM (Sarkar *et al.* 2006), a decomposition of the physical domain is considered first, and the Galerkin projection is applied considering this decomposition. Our work is completely different from this work, since we consider the entire physical domain during the Galerkin projection, and the domain decomposition is only used while implementing the preconditioner.

2. Detailed problem description and proposed solving strategy

2.1. *Polynomial chaos expansion*

In this paper the polynomial chaos bases (Ghanem & Spanos 2003) are used to represent the stochastic counterpart of the response $u(\boldsymbol{x}, \boldsymbol{\xi})$. Accordingly, any square integrable random variable, vector or process can be represented using a basis set $\{\psi_i(\boldsymbol{\xi})\}_{i=0}^{i=\infty}$, where the bases are chosen to be Hermite polynomials in the set of orthonormal variables $\boldsymbol{\xi}$. Let the function $p(\boldsymbol{\xi})$ denote the joint probability density function of the random vector $\boldsymbol{\xi}$. Introduce the notation

$$\mathbb{E}\{\cdot\} = \int_{\Omega} \cdot dP(\theta) = \int_{\mathbb{R}^m} \cdot p(\boldsymbol{\xi}) d\boldsymbol{\xi}$$

to denote the mathematical expectation of a random quantity. Then the bases ψ_i have the following properties:

$$\psi_0 \equiv 1 \ , \quad \mathbb{E}\{\psi_i\} = 0 \ \text{ for } \ i > 0 \ , \qquad \text{and } \ \mathbb{E}\{\psi_i \psi_j\} = \delta_{i,j} \mathbb{E}\{\psi_i^2\} \ ,$$

where $\delta_{i,j}$ denotes the Kronecker delta function. For computational purposes, only a finite number of bases — P are used. P depends upon m, the stochastic dimension of the problem and the highest degree of polynomials retained — also referred to as the order of expansion. Using both the deterministic or physical bases and the stochastic bases $\{\psi_j(\boldsymbol{\xi})\}_{j=0}^{j=P-1}$, the approximation $\hat{u}(\boldsymbol{\xi})$ to the response $u(\boldsymbol{x}, \boldsymbol{\xi})$ can be represented as

$$\hat{u}(\boldsymbol{\xi}) = \sum_{i=0}^{P-1} u_{(i)} \psi_i(\boldsymbol{\xi}) \ , \quad \hat{u}(\boldsymbol{\xi}), u_{(i)} \in \mathbb{R}^n \ . \tag{2.1}$$

The chaos coefficients $u_{(i)}$ completely capture the probabilistic description of $\hat{u}(\boldsymbol{\xi})$. For example, the mean and variance of the response at the j^{th} node can be readily computed from the above expansion as

$$\text{Mean} = \bar{u} = u_{(0)}(j), \quad \text{Variance} = Var(\hat{u}) = \sum_{i=1}^{i=P-1} \left(u_{(i)}(j)\right)^2 \mathbb{E}\{\psi_i^2\} \ , \tag{2.2}$$

where $u_{(i)}(j)$ denotes the j^{th} element of the vector $u_{(i)}$. To obtain the representation (2.1) the coefficients $u_{(i)}$ need to be estimated. For most applications, the computationally fastest method for this estimation is the *stochastic Galerkin finite element method* (Ghanem & Spanos 2003; Babŭska *et al.* 2005), where a variational formulation is constructed using the bases from the tensor product space $V(\mathcal{D}) \otimes L^2(\Omega)$. Accordingly, when a Galerkin projection is taken on the deterministic bases, Eq. 1.1 can be written as

$$K(\boldsymbol{\xi}) u(\boldsymbol{\xi}) = f(\boldsymbol{\xi}) \ , \quad K(\boldsymbol{\xi}) \in \mathbb{R}^{n \times n}, \quad f(\boldsymbol{\xi}) \in \mathbb{R}^n \ , \tag{2.3}$$

where

$$K(\boldsymbol{\xi}) = \sum_{i=0}^{L-1} K_{(i)} \psi_i(\boldsymbol{\xi}) \ , \qquad f(\boldsymbol{\xi}) = \sum_{i=0}^{M-1} f_{(i)} \psi_i(\boldsymbol{\xi}) \ ,$$
$$P \geq L, M, \quad K_{(i)} \in \mathbb{R}^{n \times n}, \quad f_{(i)} \in \mathbb{R}^n \ . \tag{2.4}$$

Next, when Eq. 2.3 is projected on the chaos bases, it yields a *system of systems* of deterministic equations

$$\mathbf{K}\mathbf{u} = \mathbf{f}, \quad \mathbf{K} \in \mathbb{R}^{\mathbf{nP} \times \mathbf{nP}}, \quad \mathbf{u}, \mathbf{f} \in \mathbb{R}^{\mathbf{nP}} \ , \tag{2.5}$$

where

$$\mathbf{K} = \begin{bmatrix} \sum_{i=0}^{L-1} K_{(i)} \mathbb{E}\{\psi_i \psi_0 \psi_0\} & \sum_{i=0}^{L-1} K_{(i)} \mathbb{E}\{\psi_i \psi_1 \psi_0\} & \dots & \sum_{i=0}^{L-1} K_{(i)} \mathbb{E}\{\psi_i \psi_{P-1} \psi_0\} \\ \sum_{i=0}^{L-1} K_{(i)} \mathbb{E}\{\psi_i \psi_0 \psi_1\} & \sum_{i=0}^{L-1} K_{(i)} \mathbb{E}\{\psi_i \psi_1 \psi_1\} & \dots & \sum_{i=0}^{L-1} K_{(i)} \mathbb{E}\{\psi_i \psi_{P-1} \psi_1\} \\ \dots & \dots & \dots & \dots \\ \sum_{i=0}^{L-1} K_{(i)} \mathbb{E}\{\psi_i \psi_0 \psi_{P-1}\} & \sum_{i=0}^{L-1} K_{(i)} \mathbb{E}\{\psi_i \psi_1 \psi_{P-1}\} & \dots & \sum_{i=0}^{L-1} K_{(i)} \mathbb{E}\{\psi_i \psi_{P-1} \psi_{P-1}\} \end{bmatrix} , \tag{2.6}$$

$$\mathbf{u} = \begin{Bmatrix} u_{(0)} \\ u_{(1)} \\ \cdot \\ u_{(P-1)} \end{Bmatrix} , \qquad \mathbf{f} = \begin{Bmatrix} f_{(0)} \\ f_{(1)} \\ \cdot \\ f_{(P-1)} \end{Bmatrix} \tag{2.7}$$

2.2. *Solving the linear system*

The matrix $\mathbf{K}$ is block-sparse, where each of the blocks is of size $(n \times n)$ and is also sparse itself. The block-sparsity results from the properties of the inner products of the polynomial chaos bases, while the sparsity within the individual blocks results from the deterministic finite element discretization. Clearly the problem size — which can be characterized by the number nP — depends on (i) the size of the physical system and spatial mesh resolution — that affects n, and (ii) the level of uncertainty — that affects P. The majority of the current literature addresses solvability of Eq. 2.5 for small or medium problems. However, as the problem size grows, solving this system becomes more challenging, as the memory and computational time requirement tend to a prohibitive range. Development of efficient computational techniques to solve this problem has therefore emerged as an active area of research in recent years.

The system (2.5) can be efficiently solved using an iterative technique such as the MINRES, or conjugate gradient (CG) algorithm (Ghanem & Kruger 1996; Pellissetti & Ghanem 2000; Keese & Matthies 2005) — the solver used in this paper. For solving large-scale linear systems, the preconditioners have been playing an important role (Benzi 2002). Using a preconditioning matrix $\mathbf{M}$, the system (2.5) is transformed as

$$\text{solve} \quad \mathbf{MKu} = \mathbf{Mf} \; ; \tag{2.8}$$

here $\mathbf{M}$ should be chosen such that (i) $\mathbf{MK}$ is well-conditioned, if possible $\mathbf{M} \approx \mathbf{K}^{-1}$, (ii) $\mathbf{M}$ should be easily computable. To this end, the matrix $\mathbf{K}$ needs to be examined closely.

Using the orthogonality of the chaos polynomials, it can be shown that the matrix $K_{(0)}$, stiffness of the mean system, contributes only toward the diagonal blocks of the matrix $\mathbf{K}$ and not to the off-diagonal blocks. This can be interpreted as the diagonal blocks have a contribution from the mean system properties, and often also from the fluctuation part of these properties — depending upon the expansion of $K(\boldsymbol{\xi})$ in Eq. 2.4, whereas the off-diagonal blocks have contributions only from the fluctuation part. Thus the diagonal blocks are dominant in some sense. For such matrices it is often expected and has been observed that a block-Jacobi preconditioner helps to accelerate the convergence. In this paper a *block-Jacobi-type* preconditioner is used, which is defined as

$$\mathbf{M} = \begin{bmatrix} \frac{1}{\mathbb{E}\{\psi_0^2\}} K_{(0)}^{-1} & 0 & \dots & 0 \\ 0 & \frac{1}{\mathbb{E}\{\psi_1^2\}} K_{(0)}^{-1} & 0 & 0 \\ \dots & \dots & \dots & \dots \\ 0 & 0 & \dots & \frac{1}{\mathbb{E}\{\psi_{P-1}^2\}} K_{(0)}^{-1} \end{bmatrix} . \tag{2.9}$$

The reason for deviating from the usual block-Jacobi preconditioner is as follows. In general, the diagonal blocks of the matrix $\mathbf{K}$ are not equal, therefore in the usual block-Jacobi preconditioner the diagonal blocks in $\mathbf{M}$ also will be different. However, in our construction of the preconditioner all the diagonal blocks in $\mathbf{M}$ are kept as same — except the scaling factors, which allows using a multiple right-hand sides version of a linear solver without sacrificing the power of the block-Jacobi preconditioner. It can be shown that for a Gaussian model of $K(\boldsymbol{\xi})$, that is, when in Eq. 2.4 the only non-zero coefficients $K_{(i)}$-s are $K_{(0)}$ and the ones corresponding to ξ_i-s, the preconditioner $\mathbf{M}$ coincides with the usual block-Jacobi preconditioner (Ghanem & Kruger 1996, Pellissetti & Ghanem 2000). In Keese & Matthies 2005, a preconditioner similar to the one described above was used, except there was no $\frac{1}{\mathbb{E}\{\psi_i^2\}}$ factor in the diagonal blocks. The reason we chose the factor is

that in the i^{th} diagonal block of $\mathbf{K}$, the coefficient of $K_{(0)}$ is $\mathbb{E}\{\psi_i^2\}$. Although the results are not mentioned in this paper, it was also verified through numerical experiments that this factor helps improving the convergence.

Each application of the preconditioner $\mathbf{M}$ requires the same matrix $K_{(0)}$ to be inverted P times. Note that this is the count for each PCG iteration. However, in the PCG algorithm the action of the preconditioner is implemented as a system solving approach rather than requiring the access to $K_{(0)}^{-1}$. Thus, a system $K_{(0)}z = d$ needs to be solved instead of computing $K_{(0)}^{-1}$ explicitly, where $z, d \in \mathbb{R}^n$, and d varies. The matrix $K_{(0)}$ is sparse, and large for large-scale problems. Thus an iterative solver can be used to this end. Since this system needs to be solved repeatedly for different right-hand sides, a solution method that can reduce the computational cost in successive solving is greatly desired.

Here a domain decomposition based solver named FETI-DP is used to solve $K_{(0)}z = d$. Since the application of the preconditioner requires solving the same linear system repeatedly with different right-hand sides, a multiple right-hand sides version of FETI-DP (Farhat & Chen 1994) is used here that accelerates the convergence of the PCG method. Typically any algorithm for solving a linear system with multiple right-hand sides requires storing a set of vectors. This set is used in subsequent solving to estimate the initial iterate. The size of the storage depends on the size of the vectors. For FETI-DP, these vectors correspond only to the interface dof. Since the number of interface dof is much smaller than the total dof in the whole domain, FETI-DP is storage-efficient.

3. Domain decomposition

In domain decomposition methods the computational domain is decomposed into a set of subdomains, and a divide-and-conquer strategy is developed that permits using multiple processors more efficiently than the traditional single domain approaches. In this work a particular domain decomposition method called FETI-DP (Farhat *et al.* 2000) is used. It is an iterative method where the subdomains communicate with each other through a set of Lagrange multipliers defined at subdomain boundaries. The equilibrium of each subdomain is satisfied at each iteration, whereas the continuity of the displacement field is achieved at convergence. The FETI-DP solver is scalable for fourth-order plate and shell problems. It was further adapted to repeatedly solve a given system for multiple right-hand sides efficiently (Farhat & Chen 1994). This adapted version, referred to here as the FETI-DP-mrhs, is described in Section 3.2, and will be used for applying the preconditioner (2.9).

3.1. *Description of the FETI-DP method*

Let the physical domain $\mathcal{D}$ be divided into N_s subdomains, denoting the s^{th} subdomain by $\mathcal{D}^{(s)}$ and its boundary $\partial\mathcal{D}^{(s)}$, as shown in Fig. 1. For the subdomain $\mathcal{D}^{(s)}$ denote the stiffness matrix as $K^{(s)}$, the component of the displacement field as $u^{(s)}$ and the component of the external force as $f^{(s)}$, respectively. Let the displacement field $u^{(s)}$ be partitioned as

$$u^{(s)} = \begin{bmatrix} u_r^{(s)} \\ u_{b_c}^{(s)} \end{bmatrix}, \quad \text{where } u_r^{(s)} = \begin{bmatrix} u_{int}^{(s)} \\ u_{b_r}^{(s)} \end{bmatrix}; \tag{3.1}$$

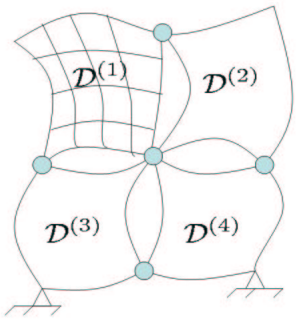

FIGURE 1. Decomposition of the physical domain $\mathcal{D}$

here the subscript b_c denotes the dof corresponding to the corner nodes, *int* denotes the internal dof, and b_r refers to dof at the nodes at the subdomain boundaries except the corners. The subscript r is to abbreviate the word *remainder*, referring to all the internal and boundary dof in the subdomain except the ones in the corners. The subdomain stiffness matrix $K^{(s)}$ and the force component $f^{(s)}$ can be partitioned accordingly as

$$K^{(s)} = \begin{bmatrix} K_{rr}^{(s)} & K_{rc}^{(s)} \\ K_{rc}^{(s)^T} & K_{cc}^{(s)} \end{bmatrix} , \quad f^{(s)} = \begin{bmatrix} f_r^{(s)} \\ f_{b_c}^{(s)} \end{bmatrix} . \tag{3.2}$$

At the subdomain interfaces, the continuity of the displacement field can be expressed as

$$u_{b_r}^{(s)} - u_{b_r}^{(q)} = 0 \ \text{ on } \partial\mathcal{D}^{(s)} \cap \partial\mathcal{D}^{(q)} . \tag{3.3}$$

However, at the corner nodes, the continuity of some or all components of the displacement field is enforced at each iteration. To implement this, a global vector is introduced to denote the corner dof as

$$u_c = \left[\ u_c^1, \cdots, u_c^j, \cdots, u_c^{N_c} \ \right]^T , \tag{3.4}$$

where N_c denotes the total number of corner nodes, u_c^j denotes a subset or all of the displacement dof associated with the j^{th} global node that is also a corner node. Let λ denote a vector-valued Lagrange multiplier defined globally over the subdomain interface dof except the corner points, that is, on the dof denoted by b_r.

Detailed derivation of the equations can be found at Farhat *et al.* 2000. However, it can be summarised as follows. First, the equilibrium equations for each subdomain on the residual dof level are constructed. Then, the equilibrium equation at global corner dof level is constructed, considering contribution from all the subdomains. Finally, after a few algebraic operations of these equilibrium equations and the continuity condition Eq. 3.3, a symmetric positive definite dual interface problem is formed as

$$A\lambda = g \quad A \in \mathbb{R}^{N_\lambda \times N_\lambda}, \quad g \in \mathbb{R}^{N_\lambda} , \tag{3.5}$$

where N_λ denotes the size of the vector λ, expressions of the matrix A and the vector g can be found in Farhat *et al.* 2000. Eq. 3.5 is solved using a PCG algorithm. We will call

this PCG as the inner-PCG to distinguish it from the PCG used to solve Eq. 2.5, which we will refer to as the outer-PCG.

Let λ^k denote the iterate at the k^{th} inner-PCG iteration. To compute the corresponding matrix-vector (mat-vec) product, the following linear system needs to be solved

$$K^*_{cc} x^k = y^k \ , \tag{3.6}$$

where the matrix K_{cc} is defined over the global corner dof. Eq. 3.6 is called the FETI-DP coarse problem. The words *dual-primal* appear in the name FEDI-DP because it involves the dual Lagrange multipliers λ and the primal displacement field u_c. Implementation details of the FETI-DP method, and finer improvements such as augmentation of the coarse problem can be found in Farhat *et al.* 2000 and Farhat *et al.* 2005 .

3.2. *FETI-DP with multiple right-hand sides*

As can be seen in Eq. 2.9, application of the block-Jacobi-type preconditioner requires solving the same system of equations with different right-hand sides. When a FETI-DP solver is used to implement this preconditioner, effectively the problem reduces to solving repeatedly Eq. 3.5 for different right-hand sides. Here one advantage is that the storage requirement for the Krylov subspace corresponds to λ, a vector defined only on the interface dof and not on the entire domain, which is a significant saving. Eq. 3.5 can be written in its multiple right-hand sides form as

$$A\lambda_j = g_j \ , \quad A \in \mathbb{R}^{N_\lambda \times N_\lambda}, \quad \lambda_j, g_j \in \mathbb{R}^{N_\lambda} \ , \quad j = 1, \ldots, n_{rhs} \ , \tag{3.7}$$

where n_{rhs} denotes the number of right-hand sides for which the system is to be solved, N_λ denotes the size of the vector λ_j. A and g_j denote the matrix on the left and the vector on the right-hand sides of Eq. 3.5, respectively. Computational cost of this repetitive solving can be greatly reduced by re-using the Krylov subspaces used by the iterative method used in solving Eq. 3.7.

For FETI-DP, a strategy of re-using the Krylov subspace (referred to here as FETI-DP-mrhs), was demonstrated in Farhat & Chen 1994. Here this strategy is described for the first two right-hand sides, that is, how the Krylov subspace $\mathcal{S}_1$ generated to compute λ_1 is used to compute λ_2. For the subsequent right-hand sides the same strategy follows. Consider for $j = 1$ the system $A\lambda_1 = g_1$ is solved, the generated Krylov subspace $\mathcal{S}_1$ is stored. For $j = 2$ the initial iterate λ_2^0 will be chosen from subspace $\mathcal{S}_1$ and the successive iterates will be selected from the space A-orthogonal to $\mathcal{S}_1$, denoted here as $\mathcal{S}_1^*$. For achieving this, first the space $\mathbb{R}^{N_\lambda}$ is decomposed as

$$\mathbb{R}^{N_\lambda} = \mathcal{S}_1 \oplus \mathcal{S}_1^* \ . \tag{3.8}$$

Then the solution λ_2 to be decomposed as

$$\lambda_2 = \lambda_2^0 + \lambda_2^* \ , \quad \text{where} \ \ \lambda_2^0 \in \mathcal{S}_1, \ \lambda_2^* \in \mathcal{S}_1^* \ , {\lambda_2^0}^T A \lambda_2^* = {\lambda_2^*}^T A \lambda_2^0 = 0 \ . \tag{3.9}$$

λ_2 is computed in two steps: first to find λ_2^0, and then to find λ_2^*. The first step is done as follows. Let S_1 denote a rectangular matrix of size $N_\lambda \times s_1$ whose columns are $A-$orthogonal and span the subspace $\mathcal{S}_1$, where s_1 denotes the dimension of the subspace $\mathcal{S}_1$. Note that $S_1^T A S_1$ is a diagonal matrix, let us denote the diagonal elements as $\beta_i, \ \{i = 1, \ldots, s_1\}$, then λ_2^0 is computed as

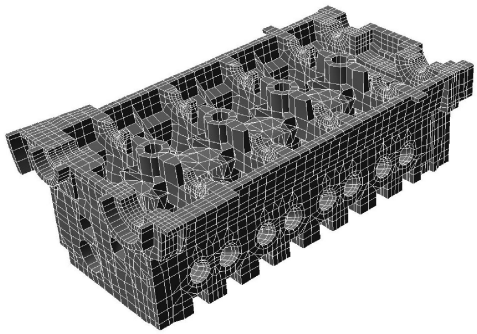

FIGURE 2. A finite element model of the cylinder head

$$\lambda_2^0 = S_1 y_2^0 \ , \tag{3.10}$$

where the elements $[y_2^0]_j$ of $y_2^0 \in \mathbb{R}^{s_1}$ are computed as

$$[y_2^0]_j = \frac{[S_1^T g_2]_j}{\beta_j} \quad j = 1, \ldots, s_1. \tag{3.11}$$

Once λ_2^0 is computed, then the PCG iterations are begun to solve $A\lambda_2 = g_2$, taking λ_2^0 as the initial iterate and taking search directions as $A-$orthogonal to λ_2^0, as follows from Eq. 3.9.

4. Numerical study

Three finite element models of a cylinder head of a car engine with three different mesh sizes are considered for implementing the proposed methodology and studying its effectiveness. These three models will be referred to here as CH1, CH2 and CH3; a typical model is shown in Fig. 2. The model CH1 has 54 198 dof, CH2 has 335 508 dof and CH3 has 2 290 437 dof. All the models are built using three types of elements: 3-D 8-node brick elements with 3dof/node, 3-D 3-AQR shell elements with 6dof/node, and 3-D 6-node pentahedral elements with 3dof/node. The cylinder head is made with five different components. The Young's modulus E_i of the materials of these five components are assumed to be independent random variables, expressed as

$$E_i = \bar{E}_i + \frac{\sigma_{E_i}}{\sqrt{2}}(\xi_i^2 - 1) \qquad i = 1, \ldots, 5 \ . \tag{4.1}$$

Here $\bar{E}_i$ are the mean values, σ_{E_i} are the standard deviations, and ξ_i are independent standard normal random variables. To ensure positivity of the Young's modulus a constraint is imposed as $\frac{\sigma_{E_i}}{\sqrt{2}} < \bar{E}_i$ for all i. In the present study σ_{E_i} is assumed to be 20% of $\bar{E}_i$ for all i, which satisfies this constraint. The system is loaded with a constant body force of $10\boldsymbol{g}$. This loading is chosen arbitrarily and without loss of generality. Since a linear static analysis is carried out here, the qualitative nature of the results should also follow for any other loading. The rigid body modes are removed from the system by properly constraining at the boundary. The displacement field is represented using fourth-order polynomial chaos expansion, the total number of chaos polynomials in this expansion becomes $P = 126$. The chaos coefficients are estimated by solving Eq. 2.5 by the PCG method, and using a block-Jacobi-type preconditioner as defined in Eq. 2.9. The

Number of subdomains	Total solution time (sec.)	Time taken in preconditioning (sec.)
22	361	257
44	317	233
223	453	371

TABLE 1. CPU time required in solving for Model CH1, the output displacement field is represented in fourth-order chaos expansion, total 8 processors are used. The total simulation time includes the time taken in preconditioning.

initial iterates for the PCG iterations are taken as zero vectors, and the stopping criteria is set as the relative residual — defined as the ratio of the 2-norms of the residual and the right-hand side — to be less than 10^{-8}. The FETI-DP-mrhs is used to implement this preconditioner, storing at most 1000 vectors from the Krylov subspaces generated by the FETI-DP solver.

Only the model CH1 is considered first. For this model, Eq. 2.5 is solved using 8 processors and three different domain decompositions: dividing the entire domain into 22, 44 and 223 subdomains, respectively. The computational time required to solve Eq. 2.5 including preconditioning, and the time required only at the preconditioning stage are presented in Table 1. It is observed from this table that in terms of computational time, there exists an optimal number of subdomains, which is 44 in this case, among the three decompositions considered. It is also observed that the majority of the computational work is in the preconditioning stage. This observation underscores the importance of accelerating the preconditioner. In Fig. 3 the number of PCG iterations within the FETI-DP solver — also referred to here as the inner-PCG iterations — is plotted against the number of times the FETI-DP solver is called. It is observed here that compared to the first few solves, the number of iterations dropped significantly afterward, that is, the mrhs version of FETI-DP helped significantly in reducing the computational cost of preconditioning. It is further observed in this figure that the trend of reduction of the number of inner-PCG iterations is almost similar for all three domain decompositions.

In the next step of the numerical study, all three models are considered. The models CH1, CH2 and CH3 are decomposed into 48, 64 and 320 subdomains, respectively. These numbers are chosen considering (i) an optimality study as presented in the previous paragraph, and (ii) load balancing — trying to keep the number of subdomains allocated per processor to be an integer value, in most cases. For three models and their corresponding domain decompositions Eq. 2.5 is solved using various numbers of processors, and the computational time and iteration counts are presented in Tables 2 and 3. The main observations from these tables are as follows.

The number of outer-PCG iterations, denoted here by $N_{iter,cg}$, does not change considerably with the mesh refinement of the models. However, the number of inner-PCG iterations, $N_{iter,FETI}$, increases noticeably as the mesh resolution (and simultaneously the number of subdomains) increases. Apparently it is expected that each outer-PCG iteration should call the FETI-DP solver $P = 126$ times, since there are 126 diagonal blocks in the preconditioner. However, in these tables, when the outer-PCG iteration count $N_{iter,cg}$ changes from 16 to 17, then the number of FETI-DP calls $N_{call,FETI}$ increases by 21, and not by 126. To resolve this contradiction, a closer examination of the final solution $\boldsymbol{u}$ revealed that among 126 number of $u_{(i)}$-s, only 21 turned out to be non-zero.

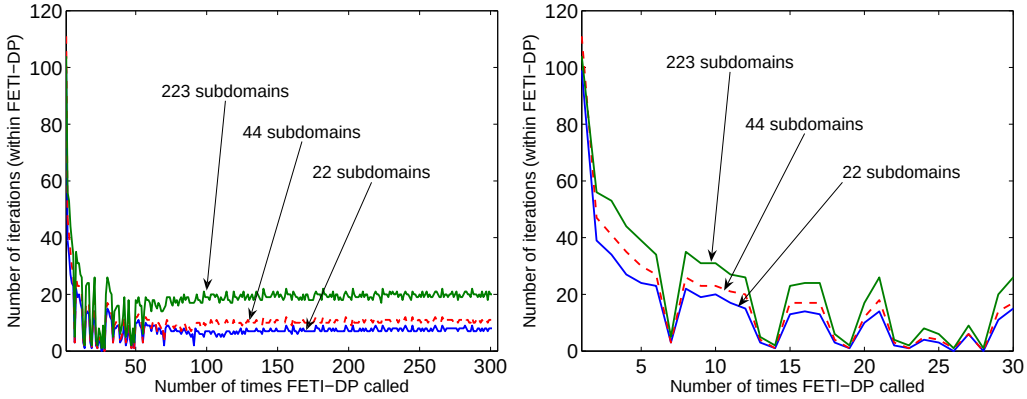

FIGURE 3. Efficiency of FETI-DP with multiple right-hand sides: Here the number of inner-PCG iterations — PCG iterations within the FETI-DP solver— is plotted as successive times this solver is called in the preconditioning stage of the large system arising from the stochastic Galerkin scheme. The figure on the right is a magnified part of the figure on the left. For the Model CH1, the output displacement field is represented in fourth-order chaos expansion.

Model	N_{proc}	$N_{iter,cg}$	$N_{call,FETI}$	$N_{iter,FETI}$	T_{sol} (sec.)	T_{prec} (sec.)
CH1	12	16	301	2687	322	241
	8			2687	376	281
	4			2686	465	347
CH2	12	17	322	4466	2265	1734
	8			4466	2673	2025

TABLE 2. Computational performance details for models CH1 and CH2, with different number of processors. The output displacement field is represented in fourth-order chaos expansion. N_{proc} : number of processors, $N_{iter,cg}$: total number of outer-PCG iterations, $N_{call,FETI}$: number of times the FETI-DP-mrhs solver is called, $N_{iter,FETI}$: total number of inner-PCG iterations (within FETI-DP-mrhs), T_{sol} : total solve time, T_{prec} : time taken at the preconditioning stage, here T_{sol} includes T_{prec}. The 8 processors are a subset of the 12 processors; the 4 processors are further a subset of these 8 processors. The model CH1 is divided into 48 subdomains, and CH2 into 64 subdomains.

To study the scalability properties, variation of the preconditioning time T_{prec} with respect to mesh resolution and number of processors should be observed. From these two tables it is observed that for a fixed number of processors, solving a problem that is 6 times larger — measured in terms of dof — takes approximately 6-7 times CPU time.

Finally, the sensitivity of the total number of inner-PCG iterations with respect to the dimension of the stored Krylov subspace in FETI-DP-mrhs is studied. For the model CH2, three different storages, 1000, 2000 and 4000 vectors are considered, and the corresponding inner-PCG iteration counts are computed and presented in Table 4. This study helps in deciding the computational budget considering the trade-off between the memory and CPU time requirement.

Model	N_{proc}	$N_{iter,cg}$	$N_{call,FETI}$	$N_{iter,FETI}$	T_{sol} (sec.)	T_{prec} (sec.)
CH2	60	17	322	4475	1041	834
CH3	60	16	301	7042	5621	4934

TABLE 3. Computational performance details for models CH2 and CH3, with different number of processors. The output displacement field is represented in fourth-order chaos expansion. N_{proc} : number of processors, $N_{iter,cg}$: total number of outer-PCG iterations, $N_{call,FETI}$: number of times the FETI-DP-mrhs solver is called, $N_{iter,FETI}$: total number of inner-PCG iterations (within FETI-DP-mrhs), T_{sol} : total solve time, T_{prec} : time taken at the preconditioning stage, here T_{sol} includes T_{prec}. The model CH2 is divided into 64 subdomains, and CH3 into 320 subdomains.

$\max_i(s_i)$	$N_{iter,FETI}$
1000	4466
2000	3184
4000	2838

TABLE 4. Effect of the maximum dimension of the stored Krylov subspace stored on the FETI-DP-mrhs acceleration. $\max_i(s_i)$: maximum number of orthogonal bases stored from any Krylov subspace $\mathcal{S}_i$, and $N_{iter,FETI}$: total number of inner-PCG iterations (within FETI-DP-mrhs). Model CH2.

5. Conclusions and future work

The numerical study demonstrated the success of the proposed methodology of analyzing uncertainty in large-scale problems. Using the block-Jacobi type preconditioner, the preconditioned conjugate gradient PCG method could solve the large problems within a few iterations. The domain decomposition based solver referred to as finite element tearing and interconnecting - dual primal (FETI-DP) worked successfully in implementing this preconditioner. The multiple right-hand sides (mrhs) version of FETI-DP helped significantly in reducing the total computational time.

The domain decomposition-based approach is a major step toward uncertainty quantification of real engineering systems through efficient usage of parallel computation. Future research directions in this area will include applications to various other kinds of engineering problems.

Acknowledgment

Financial support from the Advanced Simulation and Computing Program of the Department of Energy is gratefully acknowledged.

REFERENCES

BABŮSKA , & TEMPONE R. & ZOURARIS G. E. 2005 Solving elliptic boundary value problems with uncertain coefficients by the finite element method: the stochastic formulation *Computer Methods in Applied Mechanics and Engineering* **194**, 1251–1294.

Benzi M. 2002 Preconditioning techniques for large linear systems : a survey. *Journal of Computational Physics* **182**, 418–477.

Das S. , Ghanem R. & Spall J. 2006 Asymptotic sampling distribution for polynomial chaos representation of data: A maximum entropy and Fisher information approach. *45th IEEE Conference on Decision and Control, San Diego, CA* December 13-15.

Farhat C. , Li J. & Avery P. 2005 A FETI-DP method for parallel iterative solution of indefinite and complex-valued solid and shell vibration problems. *International Journal for Numerical Methods in Engineering* **63**, 398–427.

Farhat C. , Lesoinne M. & Pierson K. 2000 A scalable dual-primal domain decomposition method. *Numerical Linear Algebra with Applications* **7**, 687–714.

Farhat C. & Chen P. S. 1994 Tailoring domain decomposition methods for efficient parallel coarse grid solution and for systems with many right-hand sides. *Contemporary Mathematics* **180**, 401–406.

Ghanem R. & Doostan A. 2006 On the construction and analysis of stochastic predictive models: Characterization and propagation of the errors associated with limited data. *Journal of Computational Physics* **217(1)**, 63–81.

Ghanem R. & Kruger R. M. 1996 Numerical solution of spectral stochastic finite element systems. *Computer Methods in Applied Mechanics and Engineering* **129**, 289–303.

Ghanem R. & Spanos P. D. 2003 Stochastic Finite Elements: A Spectral Approach *Revised Edition, Dover Publications*

Jin C. , Cai X-C & Li C. 2007 Parallel domain decomposition methods for stochastic elliptic equations. *SIAM Journal on Scientific Computing, (Forthcoming).*

Keese A. & Matthies H. G. 2005 Hierarchical parallelisation for the solution of stochastic finite element equations. *Computers and Structures* **83**, 1033–1047.

Le Maitre O. P. , Najm H. , Ghanem R. & Knio O. 2004 Multi-resolution analysis of wiener-type uncertainty propagation schemes. *J. Comp. Ph.* **197(2)**, 502–531.

Pellissetti M. & Ghanem R. 2000 Iterative solution of systems of linear equations arising in the context of stochastic finite elements *Advances in Engng. Software* **31**, 607–616.

Sarkar A. , Benabbou N. & Ghanem R. 2006 Domain decomposition of stochastic PDEs and its parallel implementation. *International Symposium on High Performance Computing Systems, 14-17 May 2006, Newfoundland, Canada*

Xiu D. & Karniadakis G. 2003 Modeling uncertainty in flow simulations via generalized polynomial chaos. *J. Comp. Phys.* **187(1)**, 137–167.