SIAM/ASA J. UNCERTAINTY QUANTIFICATION
Vol. 1, pp. 192–217

Reduced Basis Collocation Methods for Partial Differential Equations with Random Coefficients*

Howard C. Elman† and Qifeng Liao‡

Abstract. The sparse grid stochastic collocation method is a new method for solving partial differential equations with random coefficients. However, when the probability space has high dimensionality, the number of points required for accurate collocation solutions can be large, and it may be costly to construct the solution. We show that this process can be made more efficient by combining collocation with reduced basis methods, in which a greedy algorithm is used to identify a reduced problem to which the collocation method can be applied. Because the reduced model is much smaller, costs are reduced significantly. We demonstrate with numerical experiments that this is achieved with essentially no loss of accuracy.

Key words. stochastic PDEs, collocation, reduced basis

AMS subject classifications. 35R60, 60H15, 65N22, 65N35

DOI. 10.1137/120881841

1. Introduction.

Let $(\Omega, \Sigma, \mathcal{P})$ be a complete probability space, where Ω is the sample space, $\Sigma \in 2^{\Omega}$ the σ-algebra, and $\mathcal{P} : \Sigma \to [0,1]$ the probability measure. Let $D \subset \mathbb{R}^d$ $(d = 1, 2, 3)$ be a bounded and connected domain with a polygonal boundary ∂D. We consider the problem of finding a random function $u(\vec{x}, \omega)$ mapping $D \times \Omega$ to $\mathbb{R}$ such that $\mathcal{P}$-a.e. in Ω,

$$\mathcal{L}(\vec{x}, \omega; u(\vec{x}, \omega)) = f(\vec{x}) \qquad \forall \vec{x} \in D, \tag{1.1}$$

$$\mathfrak{b}(\vec{x}, \omega; u(\vec{x}, \omega)) = g(\vec{x}) \qquad \forall \vec{x} \in \partial D, \tag{1.2}$$

where $\mathcal{L}$ is a partial differential operator and $\mathfrak{b}$ is a boundary operator, both of which can have random coefficients. In order to solve (1.1)–(1.2) numerically, the random coefficients in the operators should be represented by a finite number of random variables $\xi = [\xi_1(\omega), \ldots, \xi_M(\omega)]^T$. This could come from a variety of sources, for example, a truncated Karhunen–Loève (KL) expansion [2, 12], some partitioning of D into subdomains, or uncertain boundary conditions. Letting $\Gamma_i := [a_i, b_i]$ denote the image of $\xi_i(\omega)$ and $\Gamma := \prod_{i=1}^{M} \Gamma_i$ the image of ξ, (1.1)–(1.2) can be rewritten as follows: Find a function $u(\vec{x}, \xi)$ on $D \times \Gamma$ such that

$$\mathcal{L}(\vec{x}, \xi; u(\vec{x}, \xi)) = f(\vec{x}) \qquad \forall (\vec{x}, \xi) \in D \times \Gamma, \tag{1.3}$$

$$\mathfrak{b}(\vec{x}, \xi; u(\vec{x}, \xi)) = g(\vec{x}) \qquad \forall (\vec{x}, \xi) \in \partial D \times \Gamma, \tag{1.4}$$

*Received by the editors June 20, 2012; accepted for publication (in revised form) April 24, 2013; published electronically July 2, 2013. This work was supported in part by the U.S. Department of Energy under grants DEFG0204ER25619 and DE-SC0009301, and by the U.S. National Science Foundation under grant DMS1115317.
http://www.siam.org/journals/juq/1/88184.html

†Department of Computer Science and Institute for Advanced Computer Studies, University of Maryland, College Park, MD 20742 (elman@cs.umd.edu).

‡Department of Computer Science, University of Maryland, College Park, MD 20742 (qliao@umd.edu).

where $\mathcal{L}$ and $\mathfrak{b}$ are assumed to be *affinely* dependent on ξ. It is of interest to identify moments and cumulative distributions associated with the solution $u(\vec{x}, \xi)$. Our aim in this study is to develop a variant of the *sparse grid collocation method* for constructing solutions to (1.3)–(1.4), using *reduced basis methods* to enhance efficiency.

The sparse grid collocation method [22, 33] for stochastic partial differential equations (PDEs) is an example of a spectral method, in which discrete solutions are constructed using polynomials in the random variable ξ in combination with standard (e.g., finite element) spatial discretization. Collocation shares with Monte Carlo methods the feature that only solutions of a set of spatially discrete problems at a set of sample points $\{\xi^{(k)}\}$ are required, and it exhibits rapid convergence (nearly exponential in the number of sample points [22]), which makes it more efficient than Monte Carlo methods when the dimension M of the sample space is modest, say, on the order of 100 or smaller [19, 20]. However, the number of sample (collocation) points needed may still be large when accuracy requirements are strong, and if this is coupled with fine discretization for spatial accuracy, then the costs of stochastic collocation may be high. To address this issue, we combine collocation with *reduced basis methods* [4, 14, 21], in which parameterized discrete PDEs are projected into spaces of significantly smaller dimension. The reduced basis methodology is designed to decrease the cost of simulation of parameter-dependent models; in this study, we show that this idea can be used with stochastic collocation to the same effect. Cf. [8], which compares the properties of (full) collocation and reduced basis methods.

An outline of the rest of the paper is as follows. In sections 2 and 3, we review the collocation and reduced basis methods, and in section 4 we present our algorithm for combining them. In sections 5 and 6, we demonstrate the efficiency of the combined method for solving stochastic versions of the diffusion equation and the incompressible Navier–Stokes equations. Finally, in section 7 we make some concluding remarks.

2. Stochastic collocation on sparse grids.

The main idea of stochastic collocation methods is to seek a numerical approximation to the exact solution of (1.3)–(1.4) in the form

$$u^{sc}(\vec{x}, \xi) := \sum_{\boldsymbol{\xi}^{(k)} \in \Theta} u_c\left(\vec{x}, \xi^{(k)}\right) \tilde{L}_{\boldsymbol{\xi}^{(k)}}(\xi), \tag{2.1}$$

where $\Theta \subset \Gamma$ is a given sample set, $\{\tilde{L}_{\boldsymbol{\xi}^{(k)}}(\xi)\}$ are some global interpolation polynomials defined in Γ (e.g., Lagrange polynomials), and each coefficient function $u_c(\vec{x}, \xi^{(k)})$ is the solution of a deterministic problem corresponding to a given realization $\xi^{(k)}$ of the random variable ξ,

$$\mathcal{L}\left(\vec{x}, \xi^{(k)}; u\left(\vec{x}, \xi^{(k)}\right)\right) = f(\vec{x}) \qquad \forall \vec{x} \in D, \tag{2.2}$$

$$\mathfrak{b}\left(\vec{x}, \xi^{(k)}; u\left(\vec{x}, \xi^{(k)}\right)\right) = g(\vec{x}) \qquad \forall \vec{x} \in \partial D. \tag{2.3}$$

We will be concerned with sparse grid collocation as described in Xiu and Hesthaven [33], which is based on the methodology of sparse grid interpolation. We begin with a brief review of this interpolation technique. Without loss of generality, the image of ξ is assumed in this section to be $\Gamma^* := [-1, 1]^M$, since any finite $\Gamma = \prod_{i=1}^{M} [a_i, b_i]$ can be mapped to Γ^*. First, we consider a one-dimensional setting. Introducing a level index $i \in \mathbb{N}$, let $\Theta_1^i = \{\xi_j^i,\ j = 1 : m_i\}$

be a partitioning of the interval $[-1,1]$, where m_i is the number of partitioning points. For an arbitrary function $v(\xi) \in C([-1,1])$, its Lagrange interpolant is

$$U^i(v) = \sum_{j=1}^{m_i} v\left(\xi_j^i\right) L_j^i\left(\xi\right),$$

where

$$L_j^i\left(\xi\right) = \prod_{k=1,k\neq j}^{m_i} \frac{\xi - \xi_k^i}{\xi_k^j - \xi_k^i}.$$

A straightforward generalization for a function of M variables $v(\xi) \in C(\Gamma^*)$ is the tensor-product interpolant

$$\left(U^{i_1} \otimes \cdots \otimes U^{i_M}\right)(v) = \sum_{j_1=1}^{m_{i_1}} \cdots \sum_{j_M=1}^{m_{i_M}} v\left(\xi_{j_1}^{i_1}, \ldots, \xi_{j_M}^{i_M}\right) L_{j_1}^{i_1}\left(\xi_1\right) \cdots L_{j_1}^{i_M}\left(\xi_M\right).$$

This requires the function values at $\prod_{i=1}^M m_i$ nodes, which is of exorbitant size for large M and m_i. The number of nodes can be reduced dramatically using a sparse grid (Smolyak) operator [3],

$$A(q,M) := \sum_{q-M+1\leq|\mathbf{i}|\leq q} (-1)^{q-|\mathbf{i}|} \binom{M-1}{q-|\mathbf{i}|} \left(U^{i_1} \otimes \cdots \otimes U^{i_M}\right), \tag{2.4}$$

where $\mathbf{i} \in \mathbb{N}^M$, $|\mathbf{i}| = i_1 + \cdots + i_M$, and the index $q \geq M$ is called the sparse grid level. The sparse grid operator depends on function values at the *sparse grid points*

$$\Theta_q := \bigcup_{q-M+1\leq|\mathbf{i}|\leq q} \left(\Theta_1^{i_1} \otimes \cdots \otimes \Theta_1^{i_M}\right).$$

The size of the sample set (i.e., the number of sparse grid points) $|\Theta_q|$ can typically be chosen to be much smaller than $\prod_{i=1}^M m_i$ without significantly sacrificing interpolation accuracy [3].

Different choices of the one-dimensional partitioning sets $\{\Theta_1^i\}$ lead to different sparse grid operators, e.g., nested Clenshaw–Curtis abscissae [3, 33] and nonnested Gauss abscissae [9]. When nested abscissae are used, $A(q,M)(v)$ is an interpolant of $v(\xi)$ for any arbitrary $v(\xi) \in C(\Gamma)$. This is not true in general when nonnested abscissae are used.

In this study, we consider Clenshaw–Curtis interpolation, whose abscissae are the extrema of Chebyshev polynomials (see [33]), and Θ_q refers to the set consisting of Clenshaw–Curtis sparse grids. From [33], $|\Theta_q| \approx 2^{q-M}\frac{q!}{M!(q-M)!} \ll \prod_{i=1}^M m_i$. For this interpolation rule,

$$A(q,M)(v) \in \sum_{|\mathbf{i}|=q} \left(\boldsymbol{P}_{m_{i_1}-1} \otimes \cdots \otimes \boldsymbol{P}_{m_{i_M}-1}\right),$$

where $\boldsymbol{P}_{m_i-1}$ is the set of polynomials with degree at most $m_i - 1$ and

$$m_i = \begin{cases} 1, & i = 1, \\ 2^{i-1}+1, & i > 1. \end{cases}$$

In addition, $A(q,M)(v) = v$ whenever $v \in \sum_{|\mathbf{i}|=q} \left(\boldsymbol{P}_{m_{i_1}-1} \otimes \cdots \otimes \boldsymbol{P}_{m_{i_M}-1}\right)$. Like the generic form of collocation solutions (2.1), the specific form for sparse grid sampling is denoted by

$$u_q^{sc}\left(\vec{x}, \xi^{(k)}\right) := \sum_{\boldsymbol{\xi}^{(k)} \in \Theta_q} u_c\left(\vec{x}, \xi^{(k)}\right) L_{\boldsymbol{\xi}^{(k)}}(\xi), \tag{2.5}$$

where the interpolation polynomials $\{L_{\boldsymbol{\xi}^{(k)}}\}$ come from the definition of the Smolyak operator (2.4).[1]

We can also use the sparse grid formulation to perform quadrature to approximate the moments of u_q^{sc}. For example, the Clenshaw–Curtis quadrature rule (see [23] for details) computes the mean of $u_q^{sc}(\vec{x}, \xi)$,

$$\mathbb{E}\left(u_q^{sc}\right) = \mathbb{E}\left(u_q^{sc}(\vec{x}, \cdot)\right) := \int_\Gamma u_q^{sc}(\vec{x}, \xi)\, \rho(\xi)\, d\xi,$$

in the form

$$\tilde{\mathbb{E}}\left(u_q^{sc}\right) = \tilde{\mathbb{E}}\left(u_q^{sc}(\vec{x}, \cdot)\right) := \sum_{\boldsymbol{\xi}^{(k)} \in \Theta_q} u_c\left(\vec{x}, \xi^{(k)}\right) \rho\left(\xi^{(k)}\right) w_{\boldsymbol{\xi}^{(k)}}, \tag{2.6}$$

where $\{w_{\boldsymbol{\xi}^{(k)}}\}$ are the weights of the Clenshaw–Curtis sparse grid quadrature. It can be seen that the evaluation of the mean function (2.6) does not entail evaluation of the interpolation polynomials $L_{\boldsymbol{\xi}^{(k)}}$ in (2.5). An estimate for the variance can be computed in the same way.

Note that this Clenshaw–Curtis quadrature rule with level q is exact for polynomials in the space

$$\sum_{|\mathbf{i}|=q} \left(\boldsymbol{P}_{m_{i_1}} \otimes \cdots \otimes \boldsymbol{P}_{m_{i_M}}\right).$$

This implies $\mathbb{E}(u_q^{sc}(\vec{x}, \xi)) = \tilde{\mathbb{E}}(u_q^{sc}(\vec{x}, \xi))$ when ξ is uniformly distributed, i.e., when the density function $\rho(\xi)$ is a constant.

3. Discretization and reduced basis approximation.

In this section, we discuss finite element approximation and reduced basis methods for the problem (2.2)–(2.3). To simplify the presentation in this section, we will assume the problem satisfies homogeneous Dirichlet conditions. It is straightforward to generalize the approach to nonhomogeneous conditions, which will be discussed in section 6.

In general, we denote the weak form of the deterministic problem (2.2)–(2.3) corresponding to a given realization of ξ by

$$\mathfrak{B}_{\boldsymbol{\xi}}(u(\cdot, \xi), v) = l(v).$$

Let X^h be a spatial finite element approximation space (e.g., piecewise linear or quadratic polynomial spaces) of dimension N_h. A finite element formulation seeks an approximation $u_h(\cdot, \xi) \in X^h$ such that

$$\mathfrak{B}_\xi\left(u_h(\cdot, \xi), v\right) = l(v) \quad \forall v \in X^h. \tag{3.1}$$

[1] We refer to the MATLAB toolbox SPINTERP [17] for evaluating $L_{\boldsymbol{\xi}^{(k)}}(\xi)$.

In the following, for any ξ, the finite element solution $u_h(\cdot,\xi)$ is referred to as a snapshot associated with ξ. Grouping solutions of (3.1) with respect to all $\xi \in \Gamma$ together, we define a set consisting of all snapshots

$$S_\Gamma := \{u_h(\cdot,\xi),\ \xi \in \Gamma\}.$$

We will refer to this as the *full snapshot set*. Similarly, for a given finite set $\Theta \subset \Gamma$, we define a finite snapshot set

$$S_\Theta := \{u_h(\cdot,\xi),\ \xi \in \Theta\}.$$

The matrix form of S_Θ is denoted by $\mathbf{S}_\Theta \in \mathbb{R}^{N_h \times |\Theta|}$, i.e., each column of $\mathbf{S}_\Theta$ is the vector of basis function coefficients of a finite element solution.

In this study, we assume that the spatial mesh is sufficiently fine so that the finite element discretization error is acceptable (we refer to standard a posteriori error estimation techniques, e.g., [1, 11, 30]). With this assumption, the Galerkin equation (3.1) typically has many degrees of freedom. On the other hand, the size of the sample set $|\Theta_q|$ approximately equals $2^{q-M}\frac{q!}{M!(q-M)!}$ [33]. Although this may be much smaller than the size of the tensor sample set, it may still be very large if high accuracy with respect to collocation error is needed. The combination of large-scale spatial discretization and large numbers of sample (collocation) points can cause the cost of sparse grid collocation to be unacceptably high. One aim hence is to show that these costs can be reduced through the use of reduced basis methods.

That is, suppose we have a set of basis functions $Q = \{q_1, \ldots, q_N\} \subset X^h$, where $N \ll N_h$. In the Galerkin formulation of the reduced basis method, we seek an approximate solution $u_R(\vec{x},\xi) \in span(Q)$ such that

$$\mathfrak{B}_{\boldsymbol{\xi}}(u_R(\cdot,\xi), v) = l(v) \quad \forall v \in span(Q). \tag{3.2}$$

The reduced problem (3.2) tends to be much smaller than the full problem (3.1) [4, 21, 31]. Because of this, it is also likely to be cheaper to solve, especially if the computation is done carefully using precomputed quantities when possible. In the following, we demonstrate this approach using two types of benchmark problems. For the first, we use the diffusion equation as a prototypical example of a linear problem; the approach considered is generally applicable to linear problems, such as the Stokes equations. For the second, we explore the methodology for quadratic problems as exemplified by the Navier–Stokes equations. We note that the effective use of reduced basis methods for highly nonlinear problems is more challenging and represents an area of active research; see, for example, [7].

3.1. Linear operators. When the operator $\mathcal{L}$ of (1.3) is linear as a spatial differential operator, the discrete weak formulation (3.1) leads to a linear system

$$\mathbf{A}_{\boldsymbol{\xi}}\mathbf{u}_{\boldsymbol{\xi}} = \mathbf{f} \tag{3.3}$$

of order N_h. Since $\mathcal{L}$ is linear, $\mathbf{A}_{\boldsymbol{\xi}}$ is independent of $\mathbf{u}_{\boldsymbol{\xi}}$, and by assumption it is affinely dependent on the parameter $\xi \in \Gamma$. That is, it has the form

$$\mathbf{A}_{\boldsymbol{\xi}} = \sum_{i=1}^{K} \phi_i(\xi) A_i, \tag{3.4}$$

where $\{A_i\}$ are parameter-independent matrices and $\phi_i(\xi) \in \mathbb{R}$.

For $q_i \in Q$, let $\mathbf{q}_i$ be the vector of nodal coefficient values associated with q_i, and let $\mathbf{Q} = [\mathbf{q}_1, \ldots, \mathbf{q}_N] \in \mathbb{R}^{N_h \times N}$ be the matrix representation of Q. Then the linear system associated with the reduced problem (3.2) can be written as

$$\mathbf{Q}^T \mathbf{A}_{\boldsymbol{\xi}} \mathbf{Q} \tilde{\mathbf{u}}_{\boldsymbol{\xi}} = \mathbf{Q}^T \mathbf{f}. \tag{3.5}$$

With the expansion (3.4), (3.5) can be written as

$$\left(\sum_{i=1}^{K} \phi_i(\xi) \underbrace{\mathbf{Q}^T A_i \mathbf{Q}}_{A_{R,i}} \right) \tilde{\mathbf{u}}_{\boldsymbol{\xi}} = \underbrace{\mathbf{Q}^T \mathbf{f}}_{\mathbf{f}_R}. \tag{3.6}$$

Once the parameter-independent reduced matrices $\{A_{R,i}\}$ and vector $\mathbf{f}_R$ are precomputed, the reduced problem (3.5) for any arbitrary point $\xi \in \Gamma$ can be assembled with $O(N^2)$ operations. For u_R defined by (3.2), we will also need an estimate for a norm of the error $e_{\boldsymbol{\xi}} := u_h(\cdot, \xi) - u_R(\cdot, \xi)$, which we would also like to compute with complexity dependent only on N and not N_h. This means that standard a posteriori error estimators [1, 15, 18] for finite element methods should be excluded, since they would incur a cost proportional to N_h. An effective alternative is the *dual-based* indicator developed in [26, 31, 32], which depends on an estimate for the coercivity constant associated with the problem. If only a rough error estimate is required, we can use a simple residual indicator,

$$\eta_{Q,\boldsymbol{\xi}} := \frac{\|\mathbf{A}_{\boldsymbol{\xi}} \mathbf{Q} \tilde{\mathbf{u}}_{\boldsymbol{\xi}} - \mathbf{f}\|_2}{\|\mathbf{f}\|_2}. \tag{3.7}$$

We will use this in our experiments. It can be computed efficiently using the relation

$$\begin{aligned} \|\mathbf{A}_{\boldsymbol{\xi}} \mathbf{Q} \tilde{\mathbf{u}}_{\boldsymbol{\xi}} - \mathbf{f}\|_2^2 &= (\mathbf{A}_{\boldsymbol{\xi}} \mathbf{Q} \tilde{\mathbf{u}}_{\boldsymbol{\xi}} - \mathbf{f}, \mathbf{A}_{\boldsymbol{\xi}} \mathbf{Q} \tilde{\mathbf{u}}_{\boldsymbol{\xi}} - \mathbf{f}) = (\mathbf{A}_{\boldsymbol{\xi}} \mathbf{Q} \tilde{\mathbf{u}}_{\boldsymbol{\xi}}, \mathbf{A}_{\boldsymbol{\xi}} \mathbf{Q} \tilde{\mathbf{u}}_{\boldsymbol{\xi}}) - 2(\mathbf{A}_{\boldsymbol{\xi}} \mathbf{Q} \tilde{\mathbf{u}}_{\boldsymbol{\xi}}, \mathbf{f}) + (\mathbf{f}, \mathbf{f}) \\ &= \left(\left(\sum_{i=1}^{K} \phi_i A_i \right) \mathbf{Q} \tilde{\mathbf{u}}_{\boldsymbol{\xi}}, \left(\sum_{i=1}^{K} \phi_i A_i \right) \mathbf{Q} \tilde{\mathbf{u}}_{\boldsymbol{\xi}} \right) - 2 \left(\left(\sum_{i=1}^{K} \phi_i A_i \right) \mathbf{Q} \tilde{\mathbf{u}}_{\boldsymbol{\xi}}, \mathbf{f} \right) + (\mathbf{f}, \mathbf{f}) \\ &= \tilde{\mathbf{u}}_{\boldsymbol{\xi}}^T \left(\sum_{i=1}^{K} \sum_{j=1}^{K} \phi_i \phi_j \mathbf{Q}^T A_i^T A_j \mathbf{Q} \right) \tilde{\mathbf{u}}_{\boldsymbol{\xi}} - 2 \tilde{\mathbf{u}}_{\boldsymbol{\xi}}^T \sum_{i=1}^{K} (\phi_i \mathbf{Q}^T A_i^T \mathbf{f}) + \mathbf{f}^T \mathbf{f}. \end{aligned} \tag{3.8}$$

Once the matrices $\{\mathbf{Q}^T A_i^T A_j \mathbf{Q}\}$, vectors $\{\mathbf{Q}^T A_i^T \mathbf{f}\}$, and $\mathbf{f}^T \mathbf{f}$ are precomputed, the cost for computing $\eta_{Q,\boldsymbol{\xi}}$ is $O(N^2)$. Similar economies can be achieved with dual-based estimators [31, 32]. In the terminology of [4], we refer to the reduced dense matrices and vectors $\{A_{R,i}\}$, $\mathbf{f}_R$ in (3.6), and $\{\mathbf{Q}^T A_i^T A_j \mathbf{Q}\}$, $\{\mathbf{Q}^T A_i^T \mathbf{f}\}$, $\mathbf{f}^T \mathbf{f}$ in (3.8) as *offline reduced matrices and vectors*.

3.2. Quadratic operators.

When $\mathcal{L}$ depends quadratically on the solution u, the algebraic form of the discrete problem (3.1) can be written as

$$\mathbf{A}_{\mathbf{u},\boldsymbol{\xi}} \mathbf{u} = \mathbf{f}, \tag{3.9}$$

where $\mathbf{A}_{\mathbf{u},\boldsymbol{\xi}}$ is linearly dependent on $\mathbf{u}$; i.e., if the solution takes the form $\mathbf{u} = \sum_i c_i \mathbf{u}_i$, then

$$\mathbf{A}_{\mathbf{u},\boldsymbol{\xi}} = \sum_i c_i \mathbf{A}_{\mathbf{u}_i,\boldsymbol{\xi}}. \tag{3.10}$$

Solving (3.9) requires a nonlinear iteration that entails the solution of a linearized problem at every step, for example, as in Picard and Newton iterations [11, pp. 327–329]. At iteration step n, the linearized problem for computing the solution at step $n+1$ can be written as

$$\mathbf{A}_{\mathbf{u}^n,\boldsymbol{\xi}}\mathbf{u}^{n+1} = \mathbf{f}. \tag{3.11}$$

Like (3.5), the reduced version of (3.11) is

$$\mathbf{Q}^T\mathbf{A}_{\mathbf{u}^n,\boldsymbol{\xi}}\mathbf{Q}\tilde{\mathbf{u}}^{n+1} = \mathbf{Q}^T\mathbf{f}, \tag{3.12}$$

where $\mathbf{u}^n = \mathbf{Q}\tilde{\mathbf{u}}^n = \sum_{i=1}^N \tilde{u}_i^n\mathbf{q}_i$ with $\tilde{\mathbf{u}}^n = [\tilde{u}_1^n,\ldots,\tilde{u}_N^n]^T$ and $\mathbf{Q} = [\mathbf{q}_1,\ldots,\mathbf{q}_N]$. Using (3.10), we can rewrite (3.12) as

$$\left(\sum_{i=1}^N \tilde{u}_i^n\mathbf{Q}^T\mathbf{A}_{\mathbf{q}_i,\boldsymbol{\xi}}\mathbf{Q}\right)\tilde{\mathbf{u}}^{n+1} = \mathbf{Q}^T\mathbf{f}. \tag{3.13}$$

Under the assumption that $\mathbf{A}_{\mathbf{q}_i,\boldsymbol{\xi}}$ is affinely parameter-dependent, which implies

$$\mathbf{A}_{\mathbf{q}_i,\boldsymbol{\xi}} = \sum_{j=1}^K \phi_j(\xi)\, A_{\mathbf{q}_i,j}, \tag{3.14}$$

the reduced linear system for quadratic operators can be finally stated as

$$\left(\sum_{i=1}^N\sum_{j=1}^K \phi_j(\xi)\,\tilde{u}_i^n\underbrace{\mathbf{Q}^T A_{\mathbf{q}_i,j}\mathbf{Q}}_{\tilde{A}_{i,j}}\right)\tilde{\mathbf{u}}^{n+1} = \mathbf{Q}^T\mathbf{f}. \tag{3.15}$$

Once the parameter-independent and solution-independent matrices $\{\tilde{A}_{i,j}\}$ are precomputed, the reduced system (3.15) can be assembled with a cost $O(N^3)$. Similarly to (3.8), we can also develop a reduced version of the residual indicator

$$\eta_{Q,\boldsymbol{\xi},n} := \frac{\|\mathbf{A}_{\mathbf{u}^n,\boldsymbol{\xi}}\mathbf{Q}\tilde{\mathbf{u}}^n - \mathbf{f}\|_2}{\|\mathbf{f}\|_2}. \tag{3.16}$$

With some precomputed offline reduced matrices and vectors as discussed in section 3.1, it can be verified that computing the reduced residual indicator for quadratic operators also costs $O(N^3)$. See [31] for efficient methods for computing dual-based error estimates.

In the next section, a systematic way for constructing the reduced basis $\mathbf{Q}$ together with computing the collocation solution is introduced.

4. Reduced basis collocation.

We introduce a reduced basis collocation method for cheaply computing the collocation solution $u_q^{sc}(\cdot,\xi)$ in (2.5). Our main idea is to use u_R, the solution of the reduced problem (3.2), to serve in place of the collocation coefficient function $u_c(\cdot,\xi)$ in (2.5) wherever possible. That is, given a collocation point $\xi^{(k)}$, we compute $u_R(\cdot,\xi^{(k)})$ and an error indicator such as (3.7) or (3.16). If this error indicator is smaller than some specified tolerance, we use $u_R(\cdot,\boldsymbol{\xi}^{(k)})$; if the error indicator is too large, then we compute the snapshot $u_h(\cdot,\boldsymbol{\xi}^{(k)})$ and use it as $u_c(\cdot,\xi)$ in (2.5). In the latter case, we also augment the reduced basis with this snapshot. Our strategy is described in detail as follows.

1. Starting with level $p = M$ (the set Θ_M has only one point, which is denoted by $\xi^{(0)}$), compute the snapshot $u_h(\cdot, \xi^{(0)})$. Initialize the reduced basis $Q = \{u_h(\cdot, \xi^{(0)})\}$. In addition, use $u_h(\cdot, \xi^{(0)})$ to serve as the coefficient function $u_c(\cdot, \xi^{(0)})$ in (2.5).
2. Consider one higher level, i.e., $p+1$. Looping over sparse grid points in level $p+1$, for each point $\xi^{(k)}$, compute the reduced solution $u_R(\cdot, \xi^{(k)})$ in (3.2) and estimate a norm of the error $e_{\boldsymbol{\xi}^{(k)}} = u_h(\cdot, \xi^{(k)}) - u_R(\cdot, \xi^{(k)})$.
   (a) If the estimated error is smaller than a given tolerance, use $u_R(\cdot, \boldsymbol{\xi}^{(k)})$ to serve as the coefficient function $u_c(\cdot, \xi^{(k)})$ in (2.5).
   (b) If the estimated error is larger than the tolerance, compute the snapshot $u_h(\cdot, \xi^{(k)})$ and augment the reduced basis Q with it. Then, use $u_h(\cdot, \xi^{(k)})$ to serve as the coefficient function $u_c(\cdot, \xi^{(k)})$ in (2.5).
3. Update the sparse grid level (i.e., let $p = p + 1$), and repeat step 2, until the level p reaches the given maximum level q.

This strategy is stated more formally in Algorithm 1 below. Here, *tol* stands for a given tolerance and the orthogonal complement $T(\mathbf{u}_{\boldsymbol{\xi}^{(k)}})$ is defined as

$$T(\mathbf{u}_{\boldsymbol{\xi}^{(k)}}) = \mathbf{u}_{\boldsymbol{\xi}^{(k)}} - \Pi_{\mathbf{Q}}(\mathbf{u}_{\boldsymbol{\xi}^{(k)}}),$$

where $\Pi_{\mathbf{Q}}(\mathbf{u}_{\boldsymbol{\xi}^{(k)}})$ is the image of $\mathbf{u}_{\boldsymbol{\xi}^{(k)}}$ under the L^2-projection from $\mathbb{R}^{N_h}$ to $span\{\mathbf{q}_i\}_{i=1}^{N}$. $T(\mathbf{u}_{\boldsymbol{\xi}^{(k)}})$ can be computed using the Gram–Schmidt process, as implemented in the MATLAB function **qr**. In what follows, the sparse grid collocation solution associated with Algorithm 1 is denoted by u_q^{rsc}, and the full sparse grid collocation solution whose coefficient functions are standard finite element solutions is denoted by u_q^{hsc}. In general, computing u_q^{rsc} should be much less expensive than computing u_q^{hsc}, and as we will show in the following sections, the accuracy of the reduced solution is often comparable to that of the full solution.

Remark. In more standard use of reduced bases, the term "offline" refers to a (possibly expensive) set of computations that are done once to produce the reduced basis and quantities such as $\{A_{R,i}\}$ in (3.6) prior to the repeated solution of reduced problems for the purposes of simulation [4]. In contrast, for collocation, these computations are an integral part of Algorithm 1 and not merely a preprocessing step. However, all such "large scale" computations take place at the initial step or after the `else` clause of the inner `for` loop. It is easy to show that these computations all have complexity $O(N_h)$, as does the full system solve when multigrid is used. Thus they can be viewed as part of the overhead of generating the reduced basis solution. As we also show in the next section, this overhead diminishes in impact as the sparse grid level increases, so the role of the "offline" computations is largely analogous to what happens in a standard setting. See section 5.4 for additional discussion of this point.

5. Numerical study for diffusion problems.

In this section, we consider diffusion problems, whose governing equations are

$$-\nabla \cdot a(\cdot, \xi) \nabla u(\cdot, \xi) = 1 \quad \text{in} \quad D \times \Gamma, \tag{5.1}$$

$$u(\cdot, \xi) = 0 \quad \text{on} \quad \partial D_D \times \Gamma, \tag{5.2}$$

$$\frac{\partial u(\cdot, \xi)}{\partial n} = 0 \quad \text{on} \quad \partial D_N \times \Gamma, \tag{5.3}$$

Algorithm 1 Reduced basis collocation, homogeneous boundary conditions.

Compute S_{Θ_M} (Θ_M contains only one point).
Initialize the reduced basis matrix $\mathbf{Q} := \mathbf{S}_{\Theta_M}/\|\mathbf{S}_{\Theta_M}\|_2$.
Construct the offline reduced matrices and vectors.
for $p = 1 : q$ **do**
  for $k = 1 : |\Theta_{M+p}|$ **do**
    Compute $\tilde{\mathbf{u}}_{\boldsymbol{\xi}^{(k)}}$ by solving (3.6) and compute an error indicator, e.g., $\eta_{Q,\boldsymbol{\xi}^{(k)}}$ of (3.7).
    if $\eta_{Q,\boldsymbol{\xi}^{(k)}} < tol$ **then**
      Use the reduced solution derived from $\tilde{\mathbf{u}}_{\boldsymbol{\xi}^{(k)}}$ to serve as $u_c(\cdot, \xi^{(k)})$ in (2.5).
    else
      Compute the full solution vector $\mathbf{u}_{\boldsymbol{\xi}^{(k)}}$ by solving (3.3).
      Use the full solution derived from $\mathbf{u}_{\boldsymbol{\xi}^{(k)}}$ to serve as $u_c(\cdot, \xi^{(k)})$ in (2.5).
      Compute $T(\mathbf{u}_{\boldsymbol{\xi}^{(k)}})$, the orthogonal complement of $\mathbf{u}_{\boldsymbol{\xi}^{(k)}}$ with respect to $\mathbf{Q}$.
      Augment the reduced basis matrix $\mathbf{Q} := [\mathbf{Q}, T(\mathbf{u}_{\boldsymbol{\xi}^{(k)}})]$.
      Reconstruct the offline reduced matrices and vectors.
    end if
  end for
end for

where $\partial D = \partial D_D \cup \partial D_N$. The weak formulation is to find $u(\cdot, \xi) \in H_0^1(D)$ such that $(a\nabla u, \nabla v) = (1, v)$ for all $v \in H_0^1(D)$. We discretize in space using a bilinear ($\boldsymbol{Q}_1$) finite element approximation [5, 11].

To assess the accuracy of solutions obtained using the full and reduced basis collocation methods, we use the differences between moments of the solutions and that of a reference solution. In particular, we introduce the quantities

$$\epsilon_h := \left\| \tilde{\mathbb{E}}\left(u_q^{hsc}\right) - \tilde{\mathbb{E}}\left(u_r^{hsc}\right) \right\|_0 \Big/ \left\| \tilde{\mathbb{E}}\left(u_r^{hsc}\right) \right\|_0, \tag{5.4}$$

$$\epsilon_R := \left\| \tilde{\mathbb{E}}\left(u_q^{rsc}\right) - \tilde{\mathbb{E}}\left(u_r^{hsc}\right) \right\|_0 \Big/ \left\| \tilde{\mathbb{E}}\left(u_r^{hsc}\right) \right\|_0, \tag{5.5}$$

where the *reference collocation solution* u_r^{hsc} is a full collocation solution with a large grid level r (we take $r \geq q+2$) and the norm is the functional L^2-norm. Similarly, we approximate the variance as $\tilde{\mathbb{V}}\left(u_q^{sc}\right) := \tilde{\mathbb{E}}\left(\left(u_q^{sc}\right)^2\right) - \left(\tilde{\mathbb{E}}\left(u_q^{sc}\right)\right)^2$ and compute the relative errors

$$\zeta_h := \left\| \tilde{\mathbb{V}}\left(u_q^{hsc}\right) - \tilde{\mathbb{V}}\left(u_r^{hsc}\right) \right\|_0 \Big/ \left\| \tilde{\mathbb{V}}\left(u_r^{hsc}\right) \right\|_0, \tag{5.6}$$

$$\zeta_R := \left\| \tilde{\mathbb{V}}\left(u_q^{rsc}\right) - \tilde{\mathbb{V}}\left(u_r^{hsc}\right) \right\|_0 \Big/ \left\| \tilde{\mathbb{V}}\left(u_r^{hsc}\right) \right\|_0. \tag{5.7}$$

We will also examine the performance of Monte Carlo simulation using a sample set Θ_{mc} consisting of $|\Theta_{mc}|$ realizations of ξ, where the Monte Carlo errors for the mean and variance

are measured as

$$\epsilon_{mc} := \left\| \left(\frac{1}{|\Theta_{mc}|} \sum_{\boldsymbol{\xi} \in \Theta_{mc}} u_h(\cdot, \xi) \right) - \tilde{\mathbb{E}}\left(u_r^{hsc}\right) \right\|_0 \Big/ \left\| \tilde{\mathbb{E}}\left(u_r^{hsc}\right) \right\|_0, \tag{5.8}$$

$$\zeta_{mc} := \left\| \tilde{\mathbb{V}}_{\Theta_{mc}} - \tilde{\mathbb{V}}\left(u_r^{hsc}\right) \right\|_0 \Big/ \left\| \tilde{\mathbb{V}}\left(u_r^{hsc}\right) \right\|_0, \tag{5.9}$$

with $\tilde{\mathbb{V}}_{\Theta_{mc}} := \left(\frac{1}{|\Theta_{mc}|} \sum_{\boldsymbol{\xi} \in \Theta_{mc}} \left(u_h(\cdot, \xi)\right)^2\right) - \left(\frac{1}{|\Theta_{mc}|} \sum_{\boldsymbol{\xi} \in \Theta_{mc}} u_h(\cdot, \xi)\right)^2$. We note that reduced basis methods can also be combined with Monte Carlo methods, as discussed in [4].

5.1. Test problem 1: Piecewise constant diffusion coefficient. We consider the diffusion problem posed on the spatial domain $D = (-1, 1) \times (-1, 1)$, divided into N_D equal-sized subdomains. A pure Dirichlet condition is applied (i.e., $\partial D_N = \emptyset$). Figure 1 illustrates two cases of domain partitionings. The permeability coefficient $a(\cdot, \xi)$ is defined to be constant on each subdomain, i.e.,

$$a(\cdot, \xi)|_{D_k} = \xi_k, \quad k = 1 : N_D,$$

where the random variable $\xi = [\xi_1, \ldots, \xi_k, \ldots, \xi_{N_D}]^T$ is independently and uniformly distributed in $\Gamma := [0.01, 1]^{N_D}$. It is also assumed that the flux $a(\cdot, \xi)\nabla u(\cdot, \xi)$ is continuous across subdomain interfaces. We consider two variants of this example, one where the domains consist of vertical strips, the other using squares (see Figure 1).

(a) Case 1: N_D subdomains. (b) Case 2: $N_D = \tilde{N} \times \tilde{N}$ subdomains.

Figure 1. *Domain partitionings.*

It follows from (3.2) that the reduced basis method seeks a solution in the space $span(Q)$. If each function in the full snapshot set S_Γ can be approximated well by a linear combination of a finite set of linearly independent functions (referred to as a "basis" of S_Γ), then with this set as the reduced basis Q, the reduced solution $u_R(\cdot, \xi)$ is close to the finite element solution $u_h(\cdot, \xi)$. The size of this basis (we refer to it below as the "rank" of S_Γ) is then crucial. If the rank of S_Γ is much smaller than N_h, then Algorithm 1 can cheaply compute an accurate reduced collocation solution (i.e., $u_q^{rsc} \approx u_q^{hsc}$).

For this test problem, we check the rank of S_Γ as follows. We first generate a sample set Θ

Table 1
Estimated rank for the full snapshot set S_Γ of test problem 1, case 1.

Grid \ N_D	2	3	4	5	6	7	8	9	10
$33^2 = 1089$	3	12	18	30	40	53	55	76	84
$65^2 = 4225$	3	12	18	30	40	48	55	70	87
$129^2 = 16641$	3	12	18	28	39	48	55	72	81

Table 2
Estimated rank for the full snapshot set S_Γ of test problem 1, case 2.

Grid \ N_D	4	9	16	25	36	49	64
$33^2 = 1089$	27	121	193	257	321	385	449
$65^2 = 4225$	28	148	290	465	621	769	897
$129^2 = 16641$	28	153	311	497	746	1016	1298

consisting of 3000 random points in Γ and construct the corresponding snapshot set S_Θ.[2] We use the MATLAB function **rank** to compute the rank of $\mathbf{S}_\Theta$ (the matrix of S_Θ; see section 3). Due to the large number of sample points, this rank can serve as an estimate of the rank of the full snapshot set S_Γ.

Tables 1 and 2 show the estimated ranks for the two variants of the benchmark diffusion problems, using three different mesh sizes for the spatial discretization. It can be seen that the ranks tend to increase linearly with the number of subdomains, and they exhibit little dependence on the size of the grid. This suggests that the rank depends on properties of the underlying PDE, and it indicates that fine-grid discrete problems can be projected into subspaces of significantly smaller dimensions without sacrificing accuracy. (Although for large N_D, in cases of 36 or more square subdomains (see Table 2), the ranks are growing as the mesh is refined, they also appear to be tending to a limit with increasing number of grid points.) We also note that for some of these cases (of large N_D), the large ranks will make the reduced problems expensive to solve, although these costs would be smaller than those of the full system solves for fine enough spatial discretization.

Next, we use Algorithm 1 to compute the collocation solutions for a collection of examples of test problem 1. Tables 3–4 show results for case 1 (vertical subdomains) and Tables 5–6 for case 2 (square subdomains); all these experiments used a 65×65 discrete spatial grid. The tables show the number of full system solves $N_{full\ solve}$ and the size of the sample set $|\Theta_q|$. For example, in the case of 2×2 subdomains and $tol = 10^{-4}$ (Table 5), there are seven full system solves at the sparse grid level $q = 5$, which means that the residual error indicator is above the tolerance at seven sparse grid points among the total of nine sparse grid collocation points. This is not surprising, since the size of the reduced basis is very small at this stage (it grows from 1 to 8), and in this trivial case, there is no advantage for the reduced basis. For higher levels of sparse grids, however, the advantages become clear. At level $q = 6$, there are 41 sparse grid points, and full system solves are needed at 12 of them, and at level $q = 7$, full

[2]We have repeated this test more than ten times for different random sets, and no significantly different results were found.

Table 3

Number of full system solves for test problem 1, case 1, with 5×1 subdomains ($N_D = 5$) and a 65×65 spatial grid.

q	5	6	7	8	9	10	11	12	13	16
$\lvert\Theta_q\rvert$ / tol	1	11	61	241	801	2433	6993	19313	51713	869505
10^{-3}	1	10	9	0	0	0	0	0	0	0
10^{-4}	1	10	11	1	0	0	0	0	0	0
10^{-5}	1	10	13	0	0	0	0	0	0	0

Table 4

Number of full system solves for test problem 1, case 1, with 9×1 subdomains ($N_D = 9$), $tol = 10^{-4}$, and a 65×65 spatial grid.

q	9	10	11	12	13	14	15	16	17
$\lvert\Theta_q\rvert$	1	19	181	1177	6001	26017	100897	361249	1218049
$N_{full\ solve}$	1	18	34	2	1	1	0	0	0

Table 5

Number of full system solves for test problem 1, case 2, with 2×2 subdomains ($N_D = 4$) and a 65×65 spatial grid.

q	4	5	6	7	8	9	10	11	12	15
$\lvert\Theta_q\rvert$ / tol	1	9	41	137	401	1105	2929	7537	18945	271617
10^{-3}	1	7	11	3	0	0	0	0	0	0
10^{-4}	1	7	12	3	0	0	0	0	0	0
10^{-5}	1	7	13	2	3	0	0	0	0	0

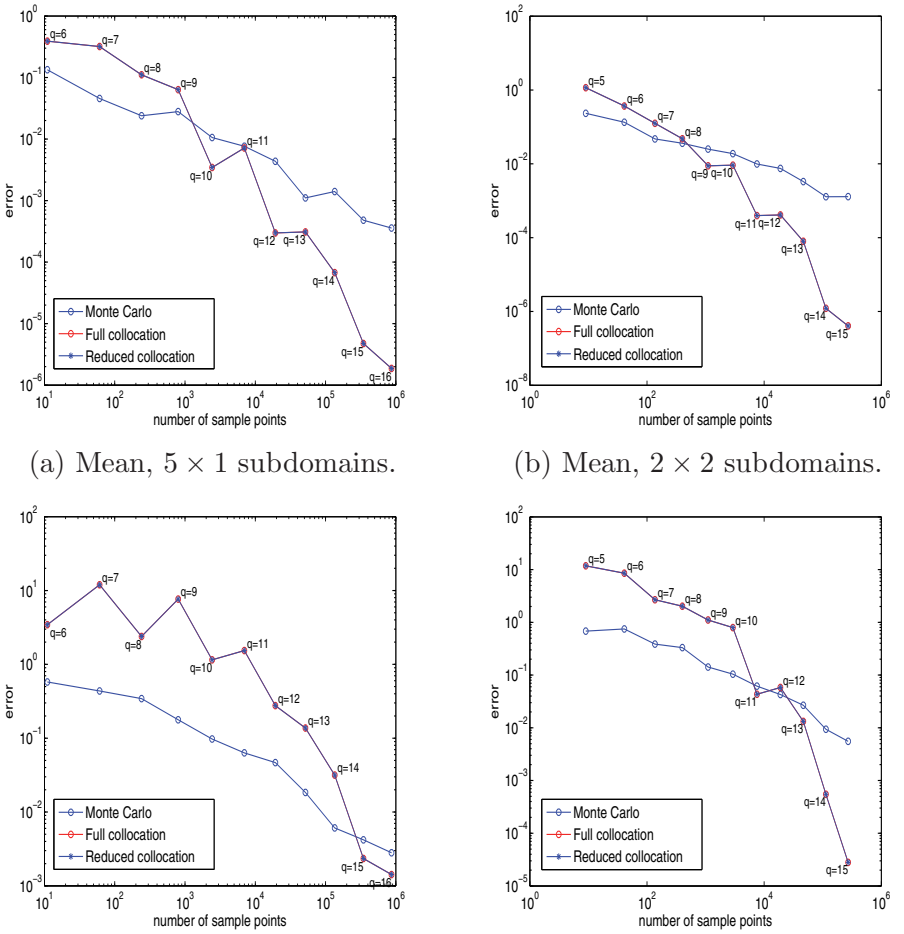
Table 6

Number of full system solves for test problem 1, case 2, with 4×4 subdomains ($N_D = 16$), $tol = 10^{-4}$, and a 65×65 spatial grid.

q	16	17	18	19	20	21
$\lvert\Theta_q\rvert$	1	33	545	6049	51137	353729
$N_{full\ solve}$	1	32	168	27	3	4

solves are needed at just 3 of 137 sparse grid points. For levels higher than 7, no full system solve is needed, which means that the reduced basis with size $N = 23$ can provide as accurate (with respect to the error indicator) a solution as the full collocation solution. This trend holds for all the examples: the number of full system solves required to generate the reduced collocation solution is dramatically smaller than the number of collocation points. Moreover, the required number of full system solves needed for the reduced basis is comparable to the ranks of the sets of full snapshots shown in Tables 1–2. For example, with four (2×2) square subdomains, the rank is 28.

For the cases of 5×1 and 2×2 subdomains, Figure 2 provides a more refined assessment of accuracy, using the relative errors in the mean (5.5) (for $tol = 10^{-4}$) and variance (5.7), where the reference levels are taken to be $r = 18$ for five vertical subdomains and $r = 17$ for four



(a) Mean, 5×1 subdomains. (b) Mean, 2×2 subdomains.

(c) Variance, 5×1 subdomains. (d) Variance, 2×2 subdomains.

Figure 2. *Test problem* 1, ϵ_R *(for tol* $= 10^{-4}$*),* ϵ_h*,* ϵ_{mc} *(top) and* ζ_R*,* ζ_h*,* ζ_{mc} *(bottom).*

square subdomains. The figure also shows similar quantities for the Monte Carlo method, ϵ_{mc} of (5.8) and ζ_{mc} of (5.9). The errors for the full collocation means (ϵ_h of (5.4)) and variances (ζ_h of (5.6)) are also plotted, but there is no visual difference between ϵ_h and ϵ_R or between ζ_h and ζ_R. Thus, the reduced collocation solution is as accurate as the full collocation solution with respect to both mean and variance. The means are also considerably more accurate than those obtained from Monte Carlo solution; the variances (for both versions of collocation) are more accurate than for Monte Carlo simulation only if a deep enough sparse grid is used.

5.2. Test problem 2: Truncated KL expansion coefficients. The spatial domain for this test problem is $D = (0, 1) \times (0, 1)$. Mixed boundary conditions are applied—the condition (5.2) is applied on the left ($x = 0$) and right ($x = 1$) boundaries, and (5.3) is applied on the top and bottom boundaries. The problem is discretized in space on a uniform 65×65 grid.

The diffusion coefficient for this test problem is assumed to be a random field with mean

Table 7

Number of full system solves for $c = 4$ with $M = 5$ in test problem 2.

q	5	6	7	8	9	10	
tol \ $\lvert\Theta_q\rvert$	1	11	61	241	801	2433	Total
10^{-5}	1	9	8	3	0	2	23
10^{-6}	1	10	12	10	2	1	36
10^{-7}	1	10	21	9	7	2	50
10^{-8}	1	10	26	18	5	2	62

Table 8

Number of full system solves for $c = 2.5$ with $M = 8$ in test problem 2.

q	8	9	10	11	12	13	
tol \ $\lvert\Theta_q\rvert$	1	17	145	849	3937	15713	Total
10^{-5}	1	14	16	4	1	1	37
10^{-6}	1	16	27	6	10	2	62
10^{-7}	1	16	50	9	8	2	86
10^{-8}	1	16	69	16	5	8	115

function $a_0(\vec{x})$, constant variance σ, and covariance function $C(\vec{x}, \vec{y})$,

$$C(\vec{x}, \vec{y}) = \sigma \exp\left(-\frac{|x_1 - y_1|}{c} - \frac{|x_2 - y_2|}{c}\right), \tag{5.10}$$

where c is the correlation length. This random field can be approximated by a truncated KL expansion [2, 9, 12]

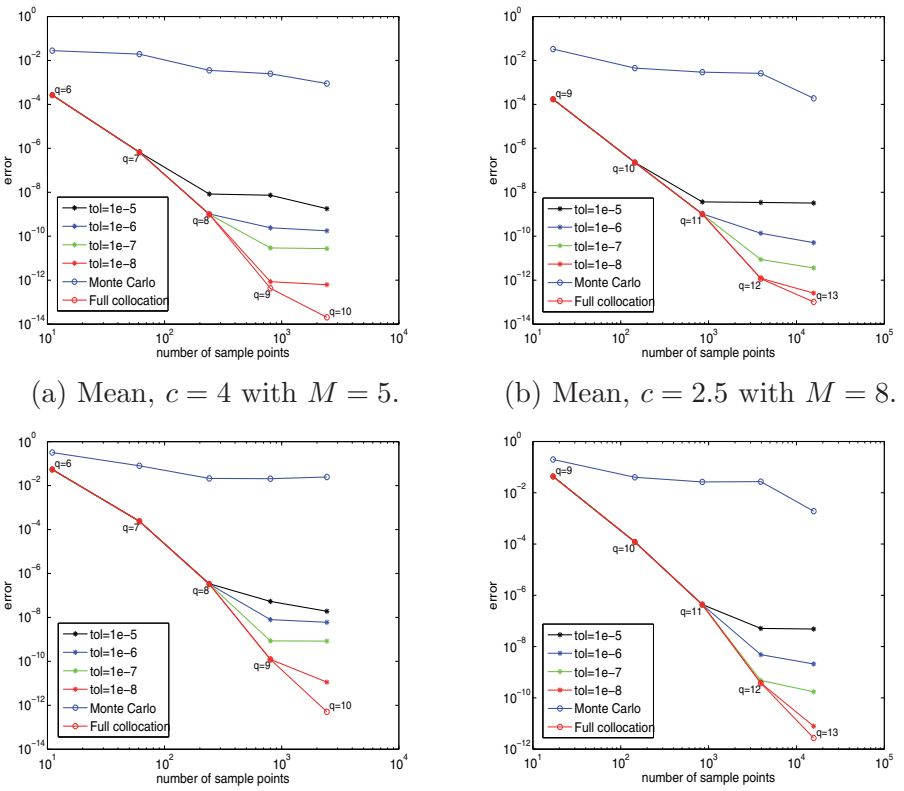
$$a(\vec{x}, \xi) \approx a_0(\vec{x}) + \sum_{k=1}^{M} \sqrt{\lambda_k} a_k(\vec{x}) \xi_k,$$

where $a_k(\vec{x})$ and λ_k are the eigenfunctions and eigenvalues of (5.10), and the random variables $\{\xi_k\}$ are assumed to be independently and uniformly distributed in $\Gamma := [-1, 1]^M$.

The error associated with truncation of the KL expansion depends on the amount of total variance captured, $\delta_{KL} := (\sum_{k=1}^{M} \lambda_k)/(|D|\sigma^2)$ [6, 27]. We chose M to be large enough so that $\delta_{KL} > 95\%$. The correlation length has an effect on this requirement—small c leads to large M.

For our experiments, we set $a_0(\vec{x}) = 1$ and $\sigma = 0.5$ and examine two values of the correlation length: $c = 4$ with $M = 5$ and $c = 2.5$ with $M = 8$. Tables 7 and 8 show the numbers of full system solves needed in Algorithm 1, for various choices of the level q and tolerance tol. Figure 3 shows the relative errors in the mean, ϵ_h, ϵ_R, and ϵ_{mc}, and the variance, ζ_h, ζ_R, and ζ_{mc}. The reference solutions correspond to reference levels $r = 12$ for $M = 5$ and $r = 15$ for $M = 8$.

The results for this example are consistent with those for problem 1 (and in this example, the trends for means and variances are identical). As the tolerance tol decreases or the sparse grid level q increases, somewhat more full systems need to be solved, but as above the number



(a) Mean, $c = 4$ with $M = 5$. (b) Mean, $c = 2.5$ with $M = 8$.

(c) Variance, $c = 4$ with $M = 5$. (d) Variance, $c = 2.5$ with $M = 8$.

Figure 3. *Test problem* 2: ϵ_h, ϵ_R, *and* ϵ_{mc} *(top) and* ζ_h, ζ_R, *and* ζ_{mc} *(bottom).*

of such solves is dramatically lower than is needed for full collocation. Moreover, the sizes of the reduced basis are very small, so the reduced system solves are inexpensive. In particular, for the extreme cases in the two examples, the full collocation method required 2433 and 15713 solves, respectively, in contrast to at most 62 and 115 for reduced basis collocation. Figure 3 shows again that there is little significant difference between the full collocation and reduced collocation solutions, and that mild tolerances for constructing the reduced basis can achieve acceptable accuracy in the reduced collocation solution.

5.3. Tolerance and error indicator in Algorithm 1. We discuss some issues related to the use of the residual error indicator (3.7) and the tolerance *tol* in Algorithm 1. For the error indicator, first, as observed above, an advantage of (3.7) is that it can be computed at cost independent of N_h using (3.8). However, this advantage holds only if the tolerance is not too small. In particular, since (3.8) requires a subtraction, the floating point computation will be accurate only if it is not strongly affected by cancellation, which is true only when the square of the residual norm is significantly larger than the machine precision. Thus, we can use this (economical) strategy only if the tolerance is not too small, on the order of 10^{-7} or larger.

For the results shown in Tables 3, 7, and 8, we found (3.8) to be reliable for the tolerances above the dotted lines. For the results below these lines, we computed the residual norm

directly, which incurs a cost proportional to N_h. Our expectation is that this cost can be avoided through use of a more effective error indicator such as the dual-based method of [26, 31].

Tables 3–8 together with Figures 2 and 3 show the impact of the tolerance *tol* on the performance of Algorithm 1. For test problem 1, Figure 2 shows that with a modest value $tol = 10^{-4}$, the errors for reduced collocation are virtually identical to those for full collocation, and (for both methods) as the level q is increased, the relative mean errors are reduced by approximately six orders of magnitude. We also found virtually no difference between these results and those for $tol = 10^{-3}$ or, for 2×2 subdomains, $tol = 10^{-5}$. Performance for test problem 2 is more sensitive to tolerance (Figure 3). A tolerance of 10^{-6} produces solutions with approximately six digits of accuracy for fine enough sparse grid, although more stringent tolerance is needed for accuracy comparable to that of the full collocation method.

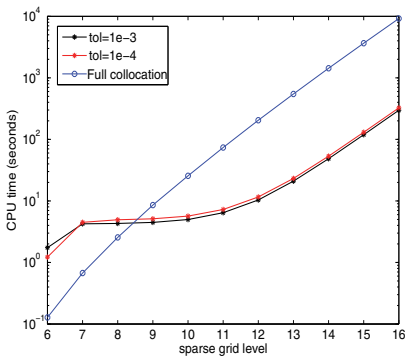
5.4. Cost assessment. In this section, we look more closely at the costs of reduced basis collocation (Algorithm 1) to get a better understanding of its effectiveness. The costs of full and reduced collocation can be specified as follows:

Full: (# of collocation points) × (cost of full system solve)
Reduced: (# of collocation points where error tolerance is met)
× (cost of reduced system solve) +
(# of collocation points where error tolerance is not met)
× (cost of augmenting reduced basis and updating offline quantities).

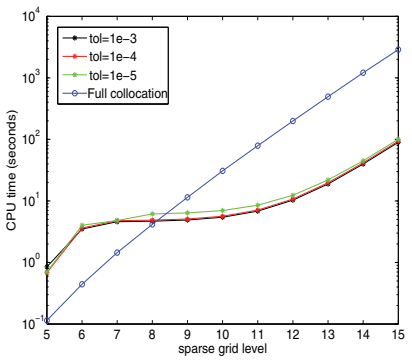
Thus, reduced basis collocation will be cheaper if the reduced solves (3.5) are less expensive than the full solves (3.3) *and* the number of collocation points where the estimated error is greater than the tolerance is not too large. The latter requirement is needed because of the overhead of augmenting the reduced basis.

The experimental results presented in sections 5.1–5.2 show that as the grid level increases, the size N of the reduced basis required for an accurate representation of the collocation solution is significantly smaller than the number of collocation points; i.e., the estimated error of the reduced solution is usually smaller than the tolerance. Thus, for deep enough grids, the main issue is the relative cost of the full and reduced solves. Standard complexity results take the cost of the reduced solve by direct methods to be $O(N^3)$ and that of the full solve using an optimal algorithm such as multigrid to be $O(N_h)$. N_h depends on spatial accuracy requirements. The experimental results indicate that N depends on the number of parameters in the problem but essentially not on the spatial grid size (Tables 1–2) and also not on the depth of the sparse grid (Tables 3–8). Thus, for any particular PDE, a fine enough spatial grid will make $N_h \gg N$ and the reduced basis method will be more efficient, although clearly for some choices of N and N_h, especially when the number of parameters is large, the full collocation method will be cheaper.

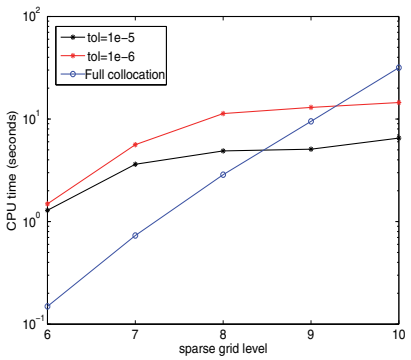
For a more concrete assessment of this point, Figure 4 shows the CPU times needed to generate the full and reduced collocation solutions for the four examples tested in sections 5.1–5.2. For reduced collocation, these are the times required for Algorithm 1. For full collocation, they reflect the time for assembling using (3.4) and solving $|\Theta_q|$ full systems at each sparse grid level q. The results were obtained in MATLAB on a MacBook Pro with 2.2



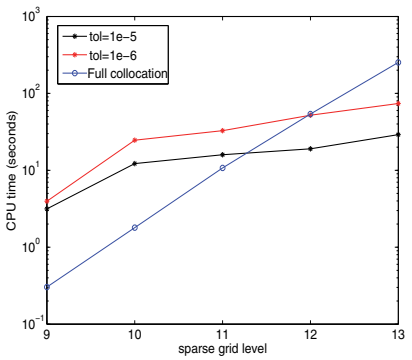
(a) CPU time, 5×1 subdomains.



(b) CPU time, 2×2 subdomains.



(c) CPU time, $c = 4$ with $M = 5$.



(d) CPU time, $c = 2.5$ with $M = 8$.

Figure 4. *CPU times for full collocation and reduced collocation with several choices of tol. Top: Test problem* 1. *Bottom: Test problem* 2.

GHz Intel Core i7 processor. All system solves used the MATLAB "backslash" operator.[3] For all these examples, the reduced basis method is more efficient when the sparse grid becomes deep enough. That is, the reduced basis is small enough that direct solution of the reduced problems is of low cost. For modest grid levels, the overhead of generating the reduced basis plays more of a role, and the reduced basis method is not as effective. When the number of parameters increases, as for test problem 2, $M = 8$ (bottom right of the figure), a deeper sparse grid is needed for the advantage of the reduced basis approach to be manifest.

6. Numerical study for incompressible flow problems.

We next consider a nonlinear example, the steady-state Navier–Stokes equations

$$-\nu(\cdot,\xi)\nabla^2\vec{u}(\cdot,\xi) + \vec{u}(\cdot,\xi)\cdot\nabla\vec{u}(\cdot,\xi) + \nabla p(\cdot,\xi) = 0 \quad \text{in} \quad D\times\Gamma, \tag{6.1}$$

$$\nabla\cdot\vec{u}(\cdot,\xi) = 0 \quad \text{in} \quad D\times\Gamma, \tag{6.2}$$

$$\vec{u}(\cdot,\xi) = \vec{g}(\cdot,\xi) \quad \text{on} \quad \partial D\times\Gamma. \tag{6.3}$$

[3]The sparse full systems come from a 65×65 spatial grid, giving $N_h = 4225$. For large N_h, such systems could be solved more efficiently using multigrid. We found MATLAB's sparse direct solver to be faster than multigrid for these examples.

The notation in (6.1)–(6.3) is standard: $\vec{u}(\cdot,\xi)$ is the flow velocity, $p(\cdot,\xi)$ is the scalar pressure, and $\nu(\cdot,\xi) > 0$ is the fluid viscosity parameter. We assume that there may be some uncertainty in the viscosity parameter $\nu(\cdot,\xi)$ (for example, in models of multiphase flows [16, 24, 29]) or the boundary data $\vec{g}(\cdot,\xi)$.

6.1. Specification of the problem. As discussed in section 2, stochastic collocation methods solve a deterministic problem at each sample point $\xi \in \Theta_q$. With the standard function space notation

$$\boldsymbol{H}^k := H^k(\Omega)^2, \quad k \in \mathbb{N}, \qquad \boldsymbol{H}^1_E := \left\{\vec{u} \in \boldsymbol{H}^1 \,|\, \vec{u} = \vec{g}\,(\cdot,\xi) \ \text{ on } \partial\Omega\right\},$$
$$\boldsymbol{H}^1_0 := \left\{\vec{u} \in \boldsymbol{H}^1 \,|\, \vec{u} = \vec{0} \ \text{ on } \partial\Omega\right\}, \qquad L^2_0(\Omega) := \left\{q \in L^2(\Omega) \,|\, \textstyle\int_\Omega q\, d\Omega = 0\right\},$$

the weak form of the deterministic problem associated with (6.1)–(6.3) is as follows: Find $\vec{u} \in \boldsymbol{H}^1_E$ and $p \in L^2_0(D)$, such that

$$(\nu\nabla\vec{u}, \nabla\vec{v}\,) + (\vec{u}\cdot\nabla\vec{u}, \vec{v}\,) - (p, \nabla\cdot\vec{v}\,) = 0 \quad \forall\vec{v} \in \boldsymbol{H}^1_0, \tag{6.4}$$

$$(\nabla\cdot\vec{u}, q) = 0 \quad \forall q \in L^2_0(D). \tag{6.5}$$

Mixed finite element approximation of (6.4)–(6.5) is obtained by choosing finite-dimensional subspaces X^h_E, X^h_0, and M^h of $\boldsymbol{H}^1_E$, $\boldsymbol{H}^1_0$, and M^h, respectively. This leads to the discrete Galerkin formulation: Find $\vec{u} \in X^h_E$ and $p \in M^h$ such that (6.4) holds for all $v \in X^h_0$ and (6.5) holds for all $q \in M^h$. We use the div-stable $\boldsymbol{Q}_2$–$\boldsymbol{P}_{-1}$ (biquadratic velocity–linear discontinuous pressure [11]) spatial discretization and denote the dimensions of X^h_0 and M^h, i.e., the numbers of velocity and pressure degrees of freedom, by $N_{h,u}$ and $N_{h,p}$, respectively.

To handle the quadratic term $(\vec{u}\cdot\nabla\vec{u}, \vec{v}\,)$ in (6.4), we use a Picard iteration as discussed in [11, pp. 324–327] and implemented in the IFISS software package [10, 28]; it is straightforward to extend the results in this section to Newton iteration. To start the Picard iteration, we can solve a discrete Stokes problem to obtain an initial guess: Find $\vec{u}^0 \in X^h_E$ and $p^0 \in M^h$ such that

$$(\nabla\vec{u}^{\,0}, \nabla\vec{v}\,) - (p^0, \nabla\cdot\vec{v}\,) = 0 \quad \forall\vec{v} \in X^h_0, \tag{6.6}$$

$$(\nabla\cdot\vec{u}^{\,0}, q) = 0 \quad \forall q \in M^h. \tag{6.7}$$

The Picard iteration then computes a sequence of corrections at step n: Find $\delta\vec{u} \in X^h_0$ and $\delta p \in M^h$, such that

$$\begin{aligned}(\nu\nabla\delta\vec{u}, \nabla\vec{v}\,) + (\vec{u}^{\,n}\cdot\nabla\delta\vec{u}, \vec{v}\,) - (\delta p, \nabla\cdot\vec{v}\,) \\ = -(\nu\nabla\vec{u}^{\,n}, \nabla\vec{v}\,) - (\vec{u}^{\,n}\cdot\nabla\vec{u}^{\,n}, \vec{v}\,) + (p^n, \nabla\cdot\vec{v}\,) \quad \forall\vec{v} \in X^h_0,\end{aligned} \tag{6.8}$$

$$(\nabla\cdot\delta\vec{u}, q) = -(\nabla\cdot\vec{u}^{\,n}, q) \quad \forall q \in M^h. \tag{6.9}$$

The velocity and pressure are then updated by

$$\vec{u}^{\,n+1} = \vec{u}^{\,n} + \delta\vec{u}, \quad p^{n+1} = p^n + \delta p.$$

Since the reduced basis methods discussed in section 3 are built on homogeneous Dirichlet conditions, some care must be taken in treatment of inhomogeneous conditions. We use an

approach described in [13], which is to first find a particular function $\vec{u}_{bc}^{\,0}$ that satisfies the Dirichlet boundary conditions and then write

$$\vec{u}^{\,0} = \vec{u}_{bc}^{\,0} + \vec{u}_{in}^{\,0},$$

where $\vec{u}_{in}^{\,0}$ satisfies homogeneous boundary conditions. We refer to $\vec{u}_{in}^{\,0}$ as the *interior part* of the Stokes solution. Then, for the initial step (6.6)–(6.7), we solve a modified Stokes problem: Find $\vec{u}_{in}^{\,0} \in X_0^h$ and $p^0 \in M^h$ such that

$$(\nabla \vec{u}_{in}^{\,0}, \nabla \vec{v}\,) - (p^0, \nabla \cdot \vec{v}\,) = -(\nabla \vec{u}_{bc}^{\,0}, \nabla \vec{v}\,) \quad \forall \vec{v} \in X_0^h, \tag{6.10}$$

$$(\nabla \cdot \vec{u}_{in}^{\,0}, q) = -(\nabla \cdot \vec{u}_{bc}^{\,0}, q) \quad \forall q \in M^h. \tag{6.11}$$

We have some flexibility in the choice of the particular function $\vec{u}_{bc}^{\,0}$. We use a simple one in which $\vec{u}_{bc}^{\,0}$ is the interpolant of the boundary data $\vec{g}(\cdot, \xi)$; that is, we define $\vec{u}_{in}^{\,0}$ and $\vec{u}_{bc}^{\,0}$ as

$$\vec{u}_{in}^{\,0} = \begin{cases} \vec{u}^{\,0} & \text{on spatial grid points in } D, \\ 0 & \text{on spatial grid points on } \partial D, \end{cases} \tag{6.12}$$

$$\vec{u}_{bc}^{\,0} = \begin{cases} 0 & \text{on spatial grid points in } D, \\ \vec{g}(\cdot, \xi) & \text{on spatial grid points on } \partial D. \end{cases} \tag{6.13}$$

Then for the Picard iteration step, no special treatment of boundaries is needed, since the correction function $\delta \vec{u}$ satisfies a homogeneous boundary condition.

The algebraic equations associated with (6.10)–(6.11) can be written as

$$\begin{bmatrix} A & B^T \\ B & 0 \end{bmatrix} \begin{bmatrix} \mathbf{u}^0 \\ \mathbf{p}^0 \end{bmatrix} = \begin{bmatrix} \mathbf{f}_{\boldsymbol{\xi}} \\ \mathbf{g}_{\boldsymbol{\xi}} \end{bmatrix},$$

and for the Picard step (6.8)–(6.9), the corresponding equations are

$$\begin{bmatrix} A_{\boldsymbol{\xi}} + N_{\mathbf{u}^n, \boldsymbol{\xi}} & B^T \\ B & 0 \end{bmatrix} \begin{bmatrix} \delta \mathbf{u} \\ \delta \mathbf{p} \end{bmatrix} = \begin{bmatrix} \mathbf{f}^r_{\mathbf{u}^n, \mathbf{p}^n, \boldsymbol{\xi}} \\ \mathbf{g}^r_{\mathbf{u}^n, \mathbf{p}^n, \boldsymbol{\xi}} \end{bmatrix},$$

where $[\mathbf{u}^n, \mathbf{p}^n]^T$ is the solution vector at the most recent iteration step, and $N_{\mathbf{u}^n, \boldsymbol{\xi}}$ is the quadratic term considered in (3.11).

6.2. Formulation of the reduced problem. We follow the development in section 3 to define the reduced versions of (6.10)–(6.11) and (6.8)–(6.9). To begin, we introduce reduced bases $Q_u := \{\vec{u}_1, \ldots, \vec{u}_{N_u}\} \subset X_0^h$ for velocity and $Q_p := \{q_1, \ldots, q_{N_p}\} \subset M^h$ for pressure with $N_p < N_u \ll N_{h,u}$. We then seek $\vec{u}_R^{\,0} \in span\{Q_u\}$ and $p_R^0 \in span\{Q_p\}$ such that

$$(\nabla \vec{u}_R^{\,0}, \nabla \vec{v}\,) - (p_R^0, \nabla \cdot \vec{v}\,) = -(\nabla \vec{u}_{bc}^{\,0}, \nabla \vec{v}\,) \quad \forall \vec{v} \in span\{Q_u\}, \tag{6.14}$$

$$(\nabla \cdot \vec{u}_R^{\,0}, q) = -(\nabla \cdot \vec{u}_{bc}^{\,0}, q) \quad \forall q \in span\{Q_p\}. \tag{6.15}$$

A Picard iteration step entails finding $\delta \vec{u}_R \in span\{Q_u\}$ and $\delta p_R \in span\{Q_p\}$ such that

$$\begin{aligned} &(\nu \nabla \delta \vec{u}_R, \nabla \vec{v}\,) + (\vec{u}_R^{\,n} \cdot \nabla \delta \vec{u}_R, \vec{v}\,) - (\delta p_R, \nabla \cdot \vec{v}\,) \\ &\qquad = -(\nu \nabla \vec{u}_R^{\,n}, \nabla \vec{v}\,) - (\vec{u}_R^{\,n} \cdot \nabla \vec{u}_R^{\,n}, \vec{v}\,) + (p_R^n, \nabla \cdot \vec{v}\,) \quad \forall \vec{v} \in span\{Q_u\}, \end{aligned} \tag{6.16}$$

$$(\nabla \cdot \delta \vec{u}_R, q) = -(\nabla \cdot \vec{u}_R^{\,n}, q) \quad \forall q \in span\{Q_p\}. \tag{6.17}$$

With $\mathbf{Q}_u$ and $\mathbf{Q}_p$ representing the matrix form of the reduced velocity and pressure bases, respectively, the linear system for the reduced problem (6.14)–(6.15) is

$$\begin{bmatrix} \mathbf{Q}_u^T A \mathbf{Q}_u & \mathbf{Q}_u^T B^T \mathbf{Q}_p \\ \mathbf{Q}_p^T B \mathbf{Q}_u & 0 \end{bmatrix} \begin{bmatrix} \tilde{\mathbf{u}}^0 \\ \tilde{\mathbf{p}}^0 \end{bmatrix} = \begin{bmatrix} \mathbf{Q}_u^T \mathbf{f}_{\boldsymbol{\xi}} \\ \mathbf{Q}_p^T \mathbf{g}_{\boldsymbol{\xi}} \end{bmatrix}, \tag{6.18}$$

and for (6.16)–(6.17),

$$\begin{bmatrix} \mathbf{Q}_u^T \left(A_{\boldsymbol{\xi}} + N_{\tilde{\mathbf{u}}^n, \boldsymbol{\xi}}\right) \mathbf{Q}_u & \mathbf{Q}_u^T B^T \mathbf{Q}_p \\ \mathbf{Q}_p^T B \mathbf{Q}_u & 0 \end{bmatrix} \begin{bmatrix} \delta \tilde{\mathbf{u}} \\ \delta \tilde{\mathbf{p}} \end{bmatrix} = \begin{bmatrix} \mathbf{Q}_u^T \mathbf{f}^r_{\tilde{\mathbf{u}}^n, \tilde{\mathbf{p}}^n, \boldsymbol{\xi}} \\ \mathbf{Q}_p^T \mathbf{g}^r_{\tilde{\mathbf{u}}^n, \tilde{\mathbf{p}}^n, \boldsymbol{\xi}} \end{bmatrix}, \tag{6.19}$$

where $[\tilde{\mathbf{u}}^n, \tilde{\mathbf{p}}^n]^T$ is the reduced solution vector at the most recent step. The residual error indicator is taken to be the discrete nonlinear residual associated with (6.4)–(6.5),

$$\eta_{Q,\boldsymbol{\xi},n} := \left\| \begin{bmatrix} A_{\boldsymbol{\xi}} + N_{\tilde{\mathbf{u}}^n, \boldsymbol{\xi}} & B^T \\ B & 0 \end{bmatrix} \begin{bmatrix} Q_u \tilde{\mathbf{u}}^n \\ Q_p \tilde{\mathbf{p}}^n \end{bmatrix} \right\|_2 \Bigg/ \left\| \begin{bmatrix} \mathbf{f}_{\boldsymbol{\xi}} \\ \mathbf{g}_{\boldsymbol{\xi}} \end{bmatrix} \right\|_2. \tag{6.20}$$

Using the techniques introduced in sections 3.1 and 3.2, once offline reduced matrices and vectors are precomputed, the reduced linear systems (6.18) and (6.19) can be assembled with costs $O(N_u^2)$ and $O(N_u^3)$, respectively, and for modest tolerances, the residual indicator can be evaluated with a cost $O(N_u^3)$.[4]

The details of reduced basis collocation for the steady-state Navier–Stokes equations are presented in Algorithm 2 below. In the algorithm, $\vec{u}_{in}\left(\cdot, \boldsymbol{\xi}^{(k)}\right)$ is the interior part of $\vec{u}\left(\cdot, \boldsymbol{\xi}^{(k)}\right)$ (see (6.12)). For the reduced basis, we note that the spaces generated by a set of snapshots

$$\left\{ \begin{pmatrix} \vec{u}_{in}\left(\cdot, \xi^{(1)}\right) \\ p\left(\cdot, \xi^{(1)}\right) \end{pmatrix}, \ldots, \begin{pmatrix} \vec{u}_{in}\left(\cdot, \xi^{(N)}\right) \\ p\left(\cdot, \xi^{(N)}\right) \end{pmatrix} \right\}$$

do not automatically satisfy an inf-sup condition

$$\gamma_R := \min_{0 \neq q_R \in span\{Q_p\}} \max_{\vec{0} \neq \vec{v}_R \in span\{Q_u\}} \frac{(q_R, \nabla \cdot \vec{v}_R)}{|\vec{v}_R|_1 \, \|q_R\|_0} \geq \gamma^* > 0$$

for γ^* independent of Q_u and Q_p. To ensure stability in this sense, we use an approach described in [25], which enriches the set of velocity snapshots with $\{\vec{r}(\cdot, \boldsymbol{\xi}^{(k)})\}_{k=1}^N$ satisfying

$$\left(\nabla \vec{r}\left(\cdot, \xi^{(k)}\right), \nabla \vec{v}\right) = \left(p\left(\cdot, \xi^{(k)}\right), \nabla \cdot \vec{v}\right) \quad \forall \vec{v} \in X_0^h. \tag{6.21}$$

These enriching functions are *supremizers* that satisfy [25]

$$\vec{r}\left(\cdot, \boldsymbol{\xi}^{(k)}\right) = \arg \sup_{\vec{v} \in X_0^h} \frac{\left(p\left(\cdot, \xi^{(k)}\right), \nabla \cdot \vec{v}\right)}{|\vec{v}|_1}.$$

It follows that the reduced bases generated in Algorithm 2 are stable in the sense that

$$\gamma_R \geq \gamma_h := \min_{0 \neq q \in M^h} \max_{\vec{0} \neq \vec{v} \in X_0^h} \frac{(q, \nabla \cdot \vec{v})}{|\vec{v}|_1 \, \|q\|_0}. \tag{6.22}$$

In addition, it is clear that $N_u = 2N_p$ for Q_u and Q_P generated in Algorithm 2.

[4] In the experiments described below, the error indicator was computed directly at cost $O(N_h)$.

Algorithm 2 Reduced basis collocation for Navier–Stokes problems.

Start with level M ($\Theta_M = \{\boldsymbol{\xi}^{(0)}\}$), and compute $\vec{u}(\cdot, \boldsymbol{\xi}^{(0)})$ and $p(\cdot, \boldsymbol{\xi}^{(0)})$.
Compute the supremizer function $r(\cdot, \boldsymbol{\xi}^{(0)})$ of (6.21).
Initialize the reduced basis $Q_u = \{\vec{u}_{in}(\cdot, \boldsymbol{\xi}^{(0)}), \vec{r}(\cdot, \boldsymbol{\xi}^{(0)})\}$ and $Q_p = \{p(\cdot, \boldsymbol{\xi}^{(0)})\}$.
Construct the offline reduced matrices and vectors.
for $p = 1 : q$ **do**
  for $k = 1 : |\Theta_{M+p}|$ **do**
    Compute the reduced solution and an error indicator $\eta_{Q,\boldsymbol{\xi}^{(k)},n}$.
    if $\eta_{Q,\boldsymbol{\xi}^{(k)},n} < tol$ **then**
      Use the reduced solution to serve as $u_c(\cdot, \xi^{(k)})$ in (2.5).
    else
    - Compute the full solutions $\vec{u}(\cdot, \xi^{(k)})$ and $p(\cdot, \xi^{(k)})$.
    - Use the full solution to serve as $u_c(\cdot, \xi^{(k)})$ in (2.5).
    - Compute the supremizer function $r(\cdot, \xi^{(k)})$ of (6.21).
    - Augment Q_u with $\{\vec{u}_{in}(\cdot, \xi^{(k)}), \vec{r}(\cdot, \xi^{(k)})\}$ and Q_p with $\{p(\cdot, \xi^{(k)})\}$, by Gram–Schmidt orthogonalization.
    - Reconstruct the offline reduced matrices and vectors.

    end if
  end for
end for

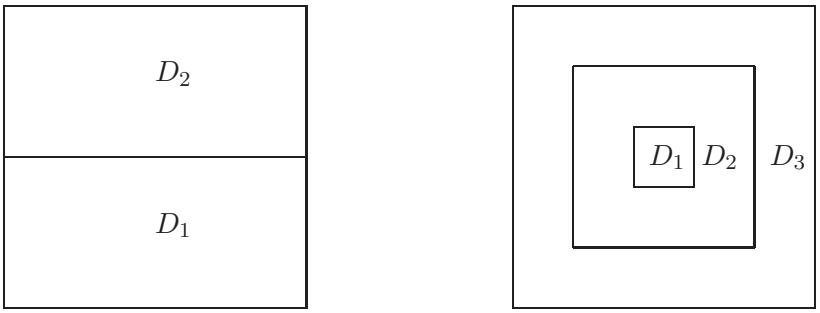
6.3. Test problem 3: Driven cavity flow with uncertainty in viscosity. The flow domain here is the square $D = (-1, 1) \times (-1, 1)$. The velocity profile

$$u = 1 - x^4, \quad v = 0, \tag{6.23}$$

is imposed on the top boundary ($y = 1$), and all other boundaries are no-slip and no-penetration so that $\vec{u} = (0, 0)$. As in test problem 1, we divide the square domain into N_D subdomains, and the viscosity is defined to be constant on each subdomain, $\nu(\cdot, \xi)|_{D_k} = \xi_k$, $k = 1 : N_D$, where the random variable $\xi = [\xi_1, \ldots, \xi_{N_D}]^T$ is uniformly distributed in $\Gamma = [0.01, 1]^{N_D}$. Two examples are shown in Figure 5. In case 1, the square domain is equally divided into two parts, and in case 2, the domain is subdivided into an interior square centered at $(0, 0)$ and two square annuli. Each of the subdomains has width 0.4. Results for uniform 33×33 and 65×65 spatial grids are reported below.

The number of full system solves are shown in Tables 9 and 10 for domain case 1 and case 2, respectively, where two tolerance values (10^{-4} and 10^{-5}) are tested. Exactly as for the diffusion equations (5.4)–(5.8), we compute the mean function errors for the velocity and pressure solutions. Figures 6 and 7 show the errors, where we used the reference level $r = 11$ for both types of domain.

The reduced inf-sup constants γ_R for domain case 2 discretized on a 65×65 spatial grid are shown in Table 11. The square of the discrete inf-sup constant for this element and mesh is known to be $\gamma_h^2 = 0.2137$ [11, p. 271]. It is evident from Table 11 that γ_R^2 is bounded below by 0.2137, which is consistent with (6.22). As the size of the reduced basis increases, γ_R^2 becomes closer to γ_h^2.



(a) Case 1. (b) Case 2.

Figure 5. *Domain partitionings for driven cavity flow.*

Table 9
Number of full system solves for test problem 3, domain case 1.

	q	2	3	4	5	6	7	8	9	
	$\lvert\Theta_q\rvert$	1	5	13	29	65	145	321	705	Total
tol	Grids									
10^{-4}	33×33	1	4	5	6	6	7	6	3	38
10^{-4}	65×65	1	4	5	5	5	5	5	2	32
10^{-5}	33×33	1	4	8	7	8	10	11	5	54
10^{-5}	65×65	1	4	8	7	8	9	9	3	49

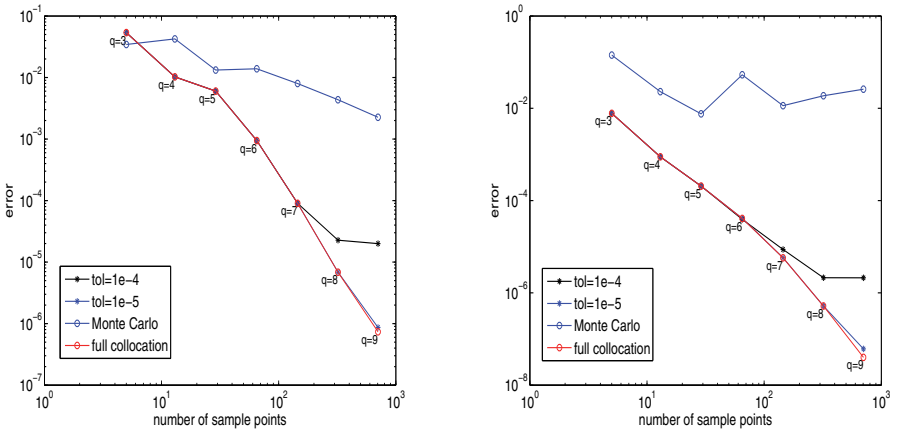
Table 10
Number of full system solves for test problem 3, domain case 2.

	q	3	4	5	6	7	8	9	
	$\lvert\Theta_q\rvert$	1	7	25	69	177	441	1073	Total
tol	Grids								
10^{-4}	33×33	1	6	17	23	26	26	25	124
10^{-4}	65×65	1	6	16	20	21	21	18	103
10^{-5}	33×33	1	6	18	29	40	44	41	179
10^{-5}	65×65	1	6	18	27	32	40	32	156

6.4. Test problem 4: Driven cavity flow with uncertain boundary conditions. The flow domain in this section is also the square $D = (-1, 1) \times (-1, 1)$, and the boundary condition (6.23) is specified at the top boundary. Unlike as section 6.3, where the other boundaries are assumed to be nonslip, we now assume there is some uncertainty on these boundaries:

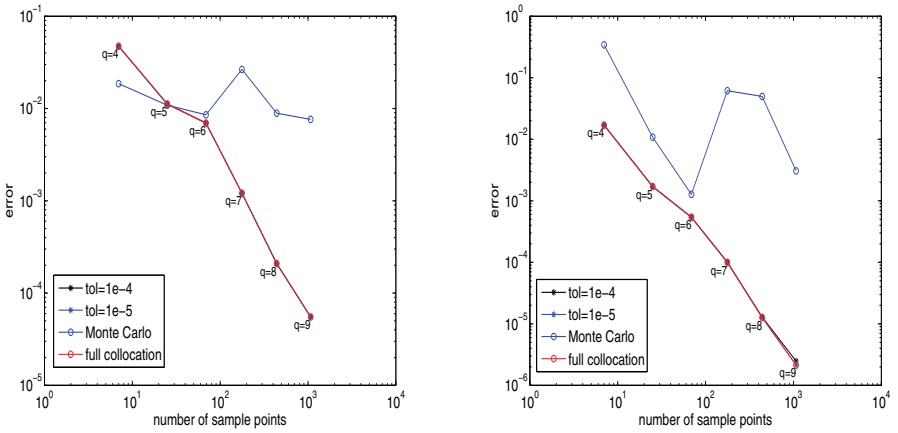
$$\begin{array}{lll} u = 0, & v = \xi_1 \left(1 - y^4\right) & \text{for } x = 1, \\ u = \xi_2 \left(1 - x^4\right), & v = 0 & \text{for } y = -1, \\ u = 0, & v = \xi_3 \left(1 - y^4\right) & \text{for } x = -1, \end{array}$$

where $\xi = [\xi_1, \xi_2, \xi_3]^T$ is assumed to be independently and uniformly distributed in $[-0.1, 0.1]^3$. The viscosity is taken to be a deterministic constant, $\nu = 1/500$ here. Instead of the uniform meshes used in previous examples, stretched meshes of sizes 33×33 and 65×65 are used to



(a) Velocity mean error. (b) Pressure mean error.

Figure 6. *Mean function errors of test problem* 3 *with domain case* 1, 65×65 *grid.*



(a) Velocity mean error. (b) Pressure mean error.

Figure 7. *Mean function errors of test problem* 3 *with domain case* 2, 65×65 *grid.*

Table 11

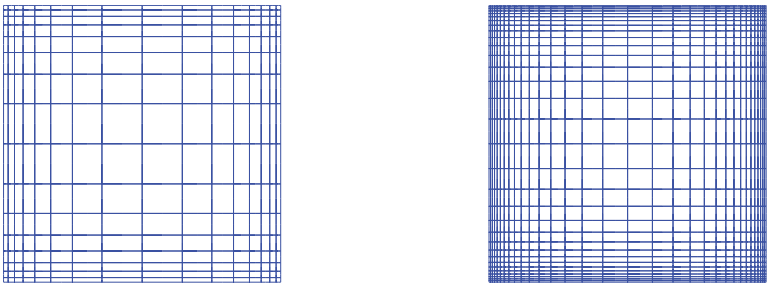
Inf-sup constants of reduced basis for test problem 3*, domain case* 2.

N_u	2	4	20	50	100	200
γ_R^2	0.2431	0.2430	0.2374	0.2359	0.2327	0.2292

discretize D (see Figure 8).[5] Table 12 shows the number of full system solves for this test problem for a range of tolerances, and Figure 9 shows the error in mean functions, where the reference level is $r = 11$.

The trends for all the examples in this section (test problem 4) and the preceding one (test

[5]These are generated by IFISS [28] using the default setting for mesh generation.

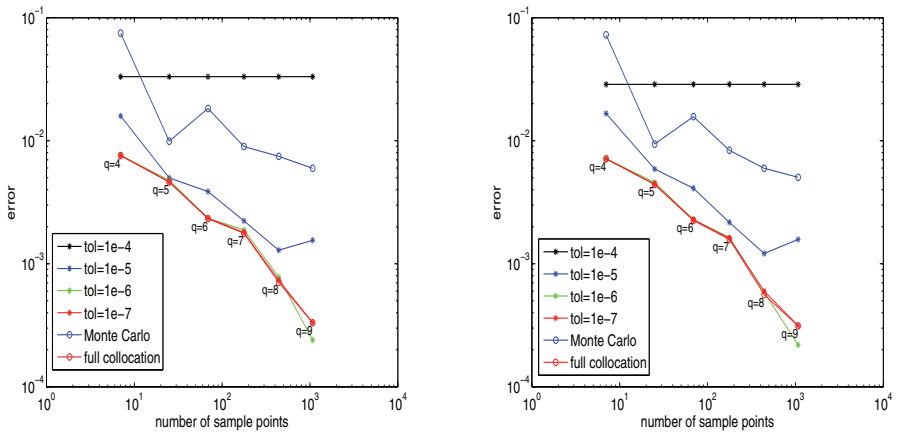


(a) Stretched 33×33 grid. (b) Stretched 65×65 grid.

Figure 8. *Meshes for test problem 4.*

Table 12
Number of full system solves for test problem 4.

	q	3	4	5	6	7	8	9	
tol	Grids \ $\lvert\Theta_q\rvert$	1	7	25	69	177	441	1073	Total
10^{-4}	33×33	1	3	1	1	0	0	0	6
10^{-4}	65×65	1	3	0	0	0	0	0	4
10^{-5}	33×33	1	6	7	4	1	0	0	19
10^{-5}	65×65	1	4	4	3	1	0	0	13
10^{-6}	33×33	1	6	15	10	8	2	0	42
10^{-6}	65×65	1	6	10	5	2	0	0	24
10^{-7}	33×33	1	6	17	27	18	10	5	84
10^{-7}	65×65	1	6	16	13	9	4	1	50



(a) Velocity mean error. (b) Pressure mean error.

Figure 9. *Mean function errors of test problem 4, stretched 65×65 grid.*

problem 3) are consistent with those for the diffusion equation. In particular, the number of full system solves required for reduced basis collocation does not increase as the number of the spatial degrees of freedom increases; these numbers are significantly smaller than what is needed for full collocation, and with moderate values of *tol* (10^{-4} for problem 3 and 10^{-6} for problem 4) the reduced solutions are as accurate as those obtained from full collocation.

7. Concluding remarks. We conclude with a brief summary of our observations from this study. The main one, seen in all the examples considered, is that the reduced basis method can be used to significantly reduce the dimension of the discrete problems that need to be solved to construct collocation solutions of stochastic partial differential equations. Moreover, the computational results indicate that the dimensions of the reduced bases do not depend on the sizes of the discrete spatial problems that the reduced problems approximate. This suggests that the reduced dimensions depend on properties of the underlying partial differential equations and that the combined reduced basis collocation method is of potential benefit whenever spatial accuracy is of importance.

REFERENCES

[1] M. Ainsworth, *A posteriori error estimation for lowest order Raviart–Thomas mixed finite elements*, SIAM J. Sci. Comput., 30 (2007), pp. 189–204.

[2] I. Babuška, F. Nobile, and R. Tempone, *A stochastic collocation method for elliptic partial differential equations with random input data*, SIAM J. Numer. Anal., 45 (2007), pp. 1005–1034.

[3] V. Barthelmann, E. Novak, and K. Ritter, *High dimensional polynomial interpolation on sparse grids*, Adv. Comput. Math., 12 (2000), pp. 273–288.

[4] S. Boyaval, C. L. Bris, T. Lelièvre, Y. Maday, N. Nguyen, and A. Patera, *Reduced basis techniques for stochastic problems*, Arch. Comput. Methods Engrg., 17 (2010), pp. 1–20.

[5] D. Braess, *Finite Elements*, Cambridge University Press, London, 1997.

[6] J. L. Brown, Jr., *Mean square truncation error in series expansions of random functions*, J. Soc. Indust. Appl. Math., 8 (1960), pp. 28–32.

[7] S. Chaturantabut and D. C. Sorensen, *Nonlinear model reduction via discrete empirical interpolation*, SIAM J. Sci. Comput., 32 (2010), pp. 2737–2764.

[8] P. Chen, A. Quarteroni, and G. Rozza, *Comparison between Reduced Basis and Stochastic Collocation Methods for Elliptic Problems*, Tech. Rep. 42.2012, MATHICSE, EPFL, 2012.

[9] H. C. Elman, C. W. Miller, E. T. Phipps, and R. S. Tuminaro, *Assessment of collocation and Galerkin approaches to linear diffusion equations with random data*, Int. J. Uncertain. Quantif., 1 (2011), pp. 19–34.

[10] H. C. Elman, A. Ramage, and D. J. Silvester, *Algorithm* 886: *IFISS, a Matlab toolbox for modelling incompressible flow*, ACM Trans. Math. Software, 33 (2007), 14.

[11] H. C. Elman, D. J. Silvester, and A. J. Wathen, *Finite Elements and Fast Iterative Solvers*, Oxford University Press, New York, 2005.

[12] R. Ghanem and P. Spanos, *Stochastic Finite Elements: A Spectral Approach*, Dover, New York, 2003.

[13] M. Gunzburger, J. Peterson, and J. Shadid, *Reduced-order modeling of time-dependent PDEs with multiple parameters in the boundary data*, Comput. Methods Appl. Mech. Engrg., 196 (2007), pp. 1030–1047.

[14] B. Haasdonk and M. Ohlberger, *Reduced basis method for finite volume approximations of parametrized linear evolution equations*, M2AN Math. Model. Numer. Anal., 42 (2008), pp. 277–301.

[15] D. Kay and D. Silvester, *A posteriori error estimation for stabilized mixed approximations of the Stokes equations*, SIAM J. Sci. Comput., 21 (1999), pp. 1321–1336.

[16] J. Kim, *Phase field computations for ternary fluid flows*, Comput. Methods Appl. Mech. Engrg., 196 (2007), pp. 4779–4788.

[17] A. Klimke, *Sparse Grid Interpolation Toolbox—User's Guide*, Tech. Rep. IANS report 2007/017, University of Stuttgart, 2007.
[18] Q. Liao and D. Silvester, *A simple yet effective a posteriori estimator for classical mixed approximation of Stokes equations*, Appl. Numer. Math., 62 (2012), pp. 1242–1256.
[19] X. Ma and N. Zabaras, *An adaptive hierarchical sparse grid collocation algorithm for the solution of stochastic differential equations*, J. Comput. Phys., 228 (2009), pp. 3084–3113.
[20] X. Ma and N. Zabaras, *An adaptive high-dimensional stochastic model representation technique for the solution of stochastic partial differential equations*, J. Comput. Phys., 229 (2010), pp. 3884–3915.
[21] N. C. Nguyen, K. Veroy, and A. T. Patera, *Certified real-time solution of parametrized partial differential equations*, in Handbook of Materials Modeling, S. Yip, ed., Kluwer Academic, Dordrecht, 2005, pp. 1523–1558.
[22] F. Nobile, R. Tempone, and C. G. Webster, *A sparse grid stochastic collocation method for partial differential equations with random input data*, SIAM J. Numer. Anal., 46 (2008), pp. 2309–2345.
[23] E. Novak and K. Ritter, *High dimensional integration of smooth functions over cubes*, Numer. Math., 75 (1996), pp. 79–97.
[24] M. A. Olshanskii and A. Reusken, *Analysis of a Stokes interface problem*, Numer. Math., 103 (2006), pp. 129–149.
[25] A. Quarteroni and G. Rozza, *Numerical solution of parametrized Navier-Stokes equations by reduced basis methods*, Numer. Methods Partial Differential Equations, 23 (2007), pp. 923–948.
[26] G. Rozza, D. Huynh, and A. Patera, *Reduced basis approximation and a posteriori error estimation for affinely parametrized elliptic coercive partial differential equations: Application to transport and continuum mechanics*, Arch. Comput. Methods Eng., 15 (2008), pp. 229–275.
[27] C. Schwab and R. A. Todor, *Karhunen-Loève approximation of random fields by generalized fast multipole methods*, J. Comput. Phys., 217 (2006), pp. 100–122.
[28] D. J. Silvester, H. C. Elman, and A. Ramage, *Incompressible Flow and Iterative Solver Software (IFISS) Version* 3.2, http://www.manchester.ac.uk/ifiss/ (May 2012).
[29] Z. Tan, D. V. Le, Z. Li, K. M. Lim, and B. C. Khoo, *An immersed interface method for solving incompressible viscous flows with piecewise constant viscosity across a moving elastic membrane*, J. Comput. Phys., 227 (2008), pp. 9955–9983.
[30] R. Verfürth, *A Review of A Posteriori Error Estimation and Adaptive Mesh-Refinement Techniques*, Wiley–Teubner, Chichester, UK, 1996.
[31] K. Veroy and A. Patera, *Certified real-time solution of the parametrized steady incompressible Navier-Stokes equations: Rigorous reduced-basis a posteriori error bounds*, Internat. J. Numer. Methods Fluids, 47 (2005), pp. 773–788.
[32] K. Veroy, A. Patera, and D. V. Rovas, A posteriori *error estimation for reduced-basis approximation of parameterized elliptic partial differential equations: "Convex inverse" bound conditioners*, Control Optim. Calculus Var., 8 (2002), pp. 1007–1028.
[33] D. Xiu and J. S. Hesthaven, *High-order collocation methods for differential equations with random inputs*, SIAM J. Sci. Comput., 27 (2005), pp. 1118–1139.