MULTILEVEL ROM

MICHAEL SCHNEIER, H.-W. VAN WYK

1. Introduction/Problem Setting. In the fields of science and engineering mathematical models are utilized to understand and predict the behavior of complex systems. Common input data/parameters for these type of models include forcing terms, boundary conditions, model coefficients and the computational domain itself. Often times for any number of reasons there is a degree of uncertainty involved with these different types of inputs. In order to obtain an accurate model we must incorporate this uncertainty into the governing equations, and quantify its effects on the output of the simulation.

In this paper we focus on systems which can be modeled by elliptic partial differential equations (PDEs) with random input data. We note that the multilevel method and reduced basis method which we will discuss can be applied in a wide variety of frameworks including elliptic, parabolic, and hyperbolic PDE's. For the purpose of this paper we will only examine a subclass of elliptic PDE's as these have been the most examined problems for the multilevel and reduced basis methods.

Let D be a bounded Lipschitz domain in $\mathbb{R}^d, d \in \{1,2,3\}$ and let (Ω, F, P) denote a complete probability space. Here Ω is the set of outcomes, $F \subset 2^\Omega$ is the σ-algebra of events, and $P :\to [0,1]$ is a probability measure. In this paper we focus on solving the stochastic elliptic problem: find $u : \Omega \times D \to \mathbb{R}$ such that it holds almost surely:

$$\begin{aligned} -\nabla\left(a(x,\omega)\nabla u(x,\omega)\right) &= f \quad x \in D, \\ u(x,\omega) &= 0 \quad x \in \partial D, \end{aligned} \tag{1.1}$$

We assume that f is a deterministic function and $a(x,\omega)$ is a random field on the probability space (Ω, F, P). We will make the following assumption for the random coefficient $a(x,\omega)$ to guarantee that (1.1) has a solution:

Assumption 1 (Uniform boundedness) There exist constants $0 < a_{min} < a_{max} < \infty$ such that

$$P(\omega \in \Omega : a_{min} < a(x,\omega) < a_{max} \ \ \forall x \in D) = 1. \tag{1.2}$$

By this assumption we have that the coefficient is coercive and continuous, hence by the Lax-Milgram theorem problem (1.1) will be well posed. Next let $\boldsymbol{y} = (y_1(\omega), ..., y_N(\omega), ...)$ be either a finite or infinite number of random variables and denote by $\Gamma_i = y_i(\Omega)$ the image of y_i. The deterministic formulation of (1.1) can be written as: find $u : D \times \Gamma \to \mathbb{R}$ such that:

$$\begin{aligned} -\nabla(a(x,\boldsymbol{y})\nabla u(x,\boldsymbol{y})) &= f \quad \forall (x,\boldsymbol{y}) \in D \times \Gamma \\ u(x,\boldsymbol{y}) &= 0 \quad \forall (x,\boldsymbol{y}) \in D \times \Gamma. \end{aligned} \tag{1.3}$$

We can now write the corresponding weak formulation, let L^2_φ be the space of square integrable functions on Γ with respect to the measure φ. We also denote by $V = H^1_0(D)$ the Sobolev space and let V' be the corresponding dual space. The weak formulation of (1.3) can be written as: find $u(x,\boldsymbol{y}) \in V \otimes L^2_\varphi(\Gamma)$ such that:

$$\int_\Gamma \int_D a(x,\boldsymbol{y})\nabla u(x,\boldsymbol{y}) \cdot \nabla v(x,\boldsymbol{y})\varphi(\boldsymbol{y})dxd\boldsymbol{y} = \int_\Gamma \int_D f(x)v(x,\boldsymbol{y})\varphi(\boldsymbol{y})dxd\boldsymbol{y} \quad \forall\, v \in V \otimes L^2_\varphi(\Gamma). \tag{1.4}$$

Once again by assumption 1 and the Lax-Milgram thoerem there exists a unique solution to (1.4) for any $f \in V'$.

We note that in this setting when we talk about the solution of a stochastic pde we are generally interested in a large class of statistical quantities of interest Q. If we let $G(u(x,\boldsymbol{y}))$ denote a physical quantity of interest

(i.e. spatial mean, function values, etc) the QOI will take the form of a stochastic integral

$$Q = \mathbb{E}[G(u(x, \boldsymbol{y}))] = \int_{\Omega} G(u(x, \boldsymbol{y})))\rho(\boldsymbol{y})d\boldsymbol{y} \tag{1.5}$$

In order to evaluate this integral a number of methods have been developed. One of these is the Monte Carlo methods Monte Carlo methods which involves sampling the PDE at different points in the parameter space, and then solving the subsequent deterministic problem. The other class of methods are known as Stochastic Collocation methods. Using a deterministic grid of points and the corresponding realizations of the PDE at these grids points, SC methods construct a polynomial approximation of the solution. For the purpose of this paper we will use the SC method known as sparse grids utilizing the nested Crenshaw Curtis rule. Full details about this method can be found in section 2 of [**?**].

Two methods which have been studied recently to further cut down on the cost of calculating these QOI's are the reduced basis method and the multilevel collocation method. Reduced basis methods seek to cut down on the computational cost of full finite element solutions by using a subspace of the full finite element space as a solution. Multilevel methods on the other hand seek to balance different errors present in the problem so that the total computational work done is optimal. We will see in the following sections by combining these two ideas it is possible to achieve a significant cost savings.

The rest of this paper will be laid out as follows. In section 2 we will introduce the reduced basis method and go into some detail about how the procedure is done. In section 3 we introduce the multilevel method and detail our implementation of the reduced basis into this framework. In section 4 we present number numerical experiments which will illustrate the performance of our reduced basis multilevel method.

2. Reduced Basis. In this section we provide a brief overview of reduced basis techniques which we will incorporate into our algorithm later on. For a more in depth discussion of reduced basis methods we refer the reader to [**?**], [**?**]. Before we begin we must once again place a restriction on the diffusion coefficient $a(x, \boldsymbol{y})$.

Assumption (Linear noise) The coefficient $a(x, \boldsymbol{y})$ can be written in the form

$$a(x, \boldsymbol{y}) = a_0(x) + \sum_{k=1}^{N} a_k(x)y_k(\omega). \tag{2.1}$$

This assumption will prove critical for the efficient calculation of the reduced basis.

Now letting $X = V \otimes L^2_{\varphi}(\Gamma)$, we can rewrite (1.4) in its bilinear form as: find $u \in X$ such that

$$A(u(y), v; y) = f(v) \quad \forall v \in X \tag{2.2}$$

and the associated Galerkin projection problem as:

$$A(u_{\mathcal{N}}(y), v; y) = f(v) \quad \forall v \in X_{\mathcal{N}} \tag{2.3}$$

where $X_{\mathcal{N}}$ is a linear subspace of dimension $\mathcal{N} >> 1$ that is an approximation of the solutions to (2.2) (i.e. finite element space) such that the approximation error $||u(y) - u_{\mathcal{N}}(y)||_X$ is sufficiently small $\forall y \in \Gamma$. In setting up our least squares problem (**??**) we will be required to solve (2.3) for many different values of y. This proves to be computationally prohibitive due to the high dimensionality $\mathcal{N}$ of the space $X_{\mathcal{N}}$. A reduced basis approach seeks to cut down on the cost of each individual realization of (2.3) by constructing a linear subspace of $X_{\mathcal{N}}$

$$X_{\mathcal{N},N} = \text{Span}(u_{\mathcal{N}}(y_n^N)) \quad n = 1, ..., N \tag{2.4}$$

where $N << \mathcal{N}$, that serves as a good approximation to our finite element space. Here each individual realization $u_{\mathcal{N}}(y_n)$ present in our reduced basis is known as a snapshot. Using our reduced basis we can now rewrite the Galerkin projection problem (2.3) as

$$A(u_{\mathcal{N},N}(y), v; y) = f(v) \quad \forall v \in X_{\mathcal{N},N}. \tag{2.5}$$

This process fundamentally relies on the fact that the manifold $\mathcal{M}_{\mathcal{N}} = \{u_{\mathcal{N}}(y), y \in \Gamma\}$ is sufficiently smooth and of low dimension so that it can be well approximated by a linear space of dimension much smaller than $\mathcal{N}$.

As we consider the discrete equations associated with (2.5), it is important to be careful about selecting the reduced basis. Assuming the snapshots $u_{\mathcal{N}}(y_n^N)$ serve as a good approximation to our space, they will become increasingly collinear as N increases. For this reason we will apply a Gram-Schmidt orthogonalization process to each individual snapshot. This gives us the basis $\{\zeta_n^{\mathcal{N}}\}_{n=1,...,N}$ with the property that $(\zeta_n, \zeta_m)_X = \delta_{mn}$ where δ_{mn} is the Kronecker delta symbol. Now letting

$$u_{\mathcal{N},N}(y) = \sum_{m=1}^{N} u_{Nm}^{\mathcal{N}}(y)\zeta_m \tag{2.6}$$

and $v = \zeta_n$, $1 < n < N$ we can then write (2.5) as

$$\sum_{m=1}^{N} A(\zeta_m^{\mathcal{N}}, \zeta_n^{\mathcal{N}}; y)u_{Nm}^{\mathcal{N}}(y) = f(\zeta_n^{\mathcal{N}}), \quad 1 \le n \le N. \tag{2.7}$$

This can be broken down even further; using assumption 3 we can rewrite (2.7) as

$$\sum_{m=1}^{N} u_{Nm}^{\mathcal{N}}(y)\big(\sum_{k=1}^{K} y_k A_k(\zeta_m^{\mathcal{N}}, \zeta_n^{\mathcal{N}})\big) = f(\zeta_n^{\mathcal{N}}), \quad 1 \le n \le N, 1 \le k \le K. \tag{2.8}$$

Since the stiffness matrix is no longer dependent on the values y_k, we can pre-compute the values (ζ_k) $1 \le k \le N$ and store them for later usage. Doing this we do not need to recompute the stiffness matrix for each value y.

Traditionally the reduced method has been characterized by its decomposition into an offline procedure, where the reduced basis is constructed, and an online portion where we now use our computationally inexpensive approximations. For the rest of this section we will examine the details of the offline and online procedures, the approximation properties of the reduced basis, and the incorporation of the reduced basis into our least squares problem.

2.1 Offline Stage

In the offline stage we typically construct our reduced basis through a greedy algorithm. Two key components of this algorithm are the training set $\Xi_{\text{train}} \subset \Gamma$ we select snapshots from, and the a posterior error estimator $\triangle_N^u(y)$ which we use to validate that our reduced basis is indeed a good approximant. In this subsection we will focus on a brief outline of the offline algorithm and the selection of the training set leaving detailed discussion of the a posterior error estimate for the following subsection.

A rough outline of the greedy algorithm reads :
1: Begin offline portion
2: randomly select $y^1 \in \Xi_{\text{train}}$
3: compute $u_{\mathcal{N}}(y^1)$ and initialize the reduced basis $X_{\mathcal{N},1} = \text{Span}(u_{\mathcal{N}}(y^1))$
4: for $n = 2$ till N_{max}
5: $y^n = \text{argmax}\{\triangle_{n-1}^u(y), y \in \Xi_{\text{train}}\}$
6: compute $u_{\mathcal{N}}(y^n)$ and let $X_{\mathcal{N},m} = \text{Span}(u_{\mathcal{N}}(y^m)), m = 1, ..., n$
7: end for

Alternatively to terminating the algorithm when we reach some level N_{max} we can preset some error tolerance ϵ_{tol} and end the algorithm when the approximation error is judged to be sufficiently small i.e. $\{\triangle_N^u(y) \le \epsilon_{tol}, \forall y \in \Xi_{\text{train}}\}$.

In constructing our training set Ξ_{train} for this algorithm we wish to pick samples that are most representative of our manifold so that they minimize the error between the finite element space and the reduced

space $||u_{\mathcal{N}}(y) - u_{\mathcal{N},N}(y)||_X$. How the points should be selected and the size of the training set is extremely problem dependent. In practice there are two approaches commonly used to construct Ξ_{train}. The first is an adaptive approach which starts with a small number of sample points and then greedily enriches the sample space based on these initial points (see [**?**]). The other method is to randomly sample the parameter space Γ according to the probability distribution present in the problem. Throughout the rest of this section we will denote the cardinality of our training set by n_{train}.

2.2 Posteriori Error Estimator

As can be seen in the above subsection, in order for the reduced basis algorithm to be efficient and accurate it is necessary that the posterior error bound $\triangle_N^u(y)$ be both sharp and inexpensive to compute. To do so we will utilize a classical technique based on residuals [**?**]. Below we will summarize the derivation and computational decomposition of the residual as done in [**?**].

First we assume $\forall y \in \Gamma$ there exists a lower bound α_{LB} for the coercivity constant of $A(\cdot, \cdot; y)$ on $X_{\mathcal{N}}$, that is

$$0 < \alpha_{\text{LB}} \leq \alpha_{\text{c}} := \inf_{z \in X_{\mathcal{N}} \setminus \{0\}} \frac{A(z, z; y)}{||z||_X^2}, \quad \forall y \in \Gamma. \tag{2.9}$$

This lower bound can either be given by an a priori analysis prior to discretization, or it can approximated numerically using an optimization technique (see section 9 [**?**]). Now we let $R(v; y) \in X'_{\mathcal{N}}$ be the residual in the bilinear form defined by

$$R(v; y) = f(v) - A(z, v; y), \quad \forall z, v \in X_{\mathcal{N}}, \forall y \in \Gamma \tag{2.10}$$

and define $\hat{g}(y) \in X_{\mathcal{N}}$ by

$$(\hat{g}(y), v)_{X_{\mathcal{N}}} = R(v; y) \ \forall v \in X_{\mathcal{N}}. \tag{2.11}$$

where $\hat{g}(y)$ is unique and guaranteed to exist by the Riesz representation theorem. Then for the posterior error estimator we have the following,

PROPOSITION 2.1. *For the linear subspace $X_{\mathcal{N},N} \subset X_{\mathcal{N}}$ there exists an error bound $\triangle_N^u(y)$ such that*

$$||u_{\mathcal{N}}(y) - u_{\mathcal{N},N}(y)|| \leq \triangle_N^u(y) := \frac{||\hat{g}(y)||_X}{\alpha_{LB}(y)} \tag{2.12}$$

Proof. Defining the error $g(y) := ||u_{\mathcal{N}}(y) - u_{\mathcal{N},N}(y)||_X$ and letting $v = g(y)$, it then follows from (2.3) and (2.10) that:

$$\begin{aligned} \alpha_{\text{LB}}(y)||g(y)||_X^2 &\leq A(g(y), g(y); y) = R(g(y); y) \\ &\leq ||R(\cdot), y||_{X'_{\mathcal{N}}}||g(y)||_X = ||\hat{g}(y)||_X||g(y)||_X \end{aligned} \tag{2.13}$$

□

Using (2.8) we expand the residual (2.10) in the form

$$R(v; y) = l(v) - \sum_{m=1}^{N} u_{Nm}^{\mathcal{N}}(\sum_{k=0}^{K} y_k A_k(\zeta_n, v)), \quad \text{where } y_0 = 1. \tag{2.14}$$

Letting $(\mathcal{C}, v)_X = F(v)$ and $(\mathcal{L}_m^k, v)_X = -A_k$ where $\mathcal{C}$ and $\mathcal{L}$ are guaranteed to exist by the Riesz representation theorem, it then follows from (2.11) that the residual can be expressed as:

$$||\hat{g}(y)||_{X_{\mathcal{N}}}^2 = (\mathcal{C}, \mathcal{C})_X + \sum_{k=0}^{K}\sum_{m=1}^{N} y_k u_{Nm}^{\mathcal{N}}(y) \left(2(\mathcal{C}, \mathcal{L}_m^k)_X + \sum_{k'=0}^{K}\sum_{m'=1}^{N} y_{k'} u_{Nm'}^{\mathcal{N}}(y)(\mathcal{L}_m^k, \mathcal{L}_{m'}^{k'})_X \right). \tag{2.15}$$

The quantities $(\mathcal{C},\mathcal{C})_X, (\mathcal{C},\mathcal{L}_m^k)_X$, and $(\mathcal{L}_m^k,\mathcal{L}_{m'}^{k'})_X$ are independent of y_k, hence they can be stored in the offline procedure, allowing for efficient computation of the residual in the online portion of the algorithm if necessary.

We note that often times we will actually be interested in measuring the error of some output based on the solution $s_{\mathcal{N}}(u) = L(u_{\mathcal{N}}(y)) \in \mathbb{R}$. We can still use the above posterori error estimator for these quantities as it will hold

$$|s_{\mathcal{N}}(u) - s_{\mathcal{N}}(u)| = |L(u_{\mathcal{N}}(y)) - L(u_{\mathcal{N},N}(y))| \leq ||L||_{X'_{\mathcal{N}}} ||u_{\mathcal{N}}(y) - u_{\mathcal{N},N}(y)||_X \tag{2.16}$$

where $||L||_{X'_{\mathcal{N}}}$ is a constant independent of y.

3. Multilevel. We begin this section by describing the multilevel method which considers multiple levels of mesh refinement. Let u_h denote some spatial approximation to a PDE u, Q_h the corresponding QOI, and $Q_{M,h}^{SC}$ to be the approximate QOI of the spatial approximation generated via a stochastic collocation method where M is the level of the method. The traditional goal has been to calculate the error $||Q - Q_{M,h}^{SC}||$. Using the triangle inequality it is possible to bound this error as

$$||Q - Q_{M,h}^{SC}|| \leq \underbrace{||Q - Q_h||}_{\text{Spatial Error}} + \underbrace{||Q_h - Q_{M,h}^{SC}||}_{\text{Sampling Error}} \tag{3.1}$$

here the spatial error is independent of the sampling error and can thus be considered separately. Generally we wish to achieve an error tolerance ϵ that balances the amount of work between these 2 errors so as to minimize the amount of work needed in our calculation.

Now for a multilevel scheme we let $\{h_\ell\}_{\ell=0}^L$ be a sequence of spatial discretization errors of increasing accuracy and let h_L be chosen such that

$$||Q - Q_h|| \leq \frac{\epsilon}{2}. \tag{3.2}$$

Multilevel quadrature methods are then based on an expansion of the the fine scale approximation as a sum on an initial court approximation plus a series of correction terms

$$Q_{h_L} = Q_{h_0} + \sum_{\ell=1}^{L} (Q_{h_\ell} - Q_{h_\ell - 1}) \tag{3.3}$$

Now since the spatial error is independent of the sampling error we may choose a different interpolation sample size M_ℓ for each spatial refinement level i.e.

$$Q_{\{M_\ell\},\{h_\ell\}}^{MLSC} = Q_{M_0,h_0}^{SC} + \sum_{\ell=1}^{L} (Q_{M_\ell,h_\ell}^{SC} - Q_{M_\ell,h_\ell-1}^{SC}) \tag{3.4}$$

This allows us to do the most sampling on the coarsest levels of our spatial approximation and the least amount of sampling on the finest level of our approximation resulting in a significant computational cost savings.

The total error for the multilevel method can then be written as

$$||Q - Q_{\{M_\ell\},\{h_\ell\}}^{MLSC}|| \leq \underbrace{||Q - Q_{h_L}||}_{\text{Spatial Error}} + \underbrace{||Q_{h_0} - Q_{M_0,h_0}^{SC}|| + \sum_{\ell=1}^{L} ||(Q_{h_\ell} - Q_{h_{\ell-1}}) - (Q_{M_\ell,h_\ell}^{SC} - Q_{M_\ell,h_{\ell-1}})||}_{\text{Multilevel Sampling Error}}. \tag{3.5}$$

Similar to (3.1) we have decomposed the error to depend only a spatial and sampling error. This time though the spatial error is the multilevel sampling error which quantifies the accuracy with which the correction terms $Q_{h_\ell} - Q_{h_{\ell-1}}$ are approximated through interpolation.

Now turning to our implementation the above analysis follows exactly, except we will consider the spatial refinement h_ℓ to be fixed. This is applicable to problems where where we only wish to set the refinement to a level where the physics of the underlying PDE are resolved. Rather we will replace the spatial refinement with a reduced basis outlined in section 2. Letting ϵ_ℓ^r $(\ell = 1, ..., L)$ a sequence of epsilon tolerances in our reduced basis method we can conduct our sampling at different levels of the reduced basis error. Similar to (3.5) We can then write the total error for our multilevel method as

$$||Q - Q^{MLSC}_{\{M_\ell\},\{\epsilon_\ell^r\}}|| \leq \underbrace{||Q - Q_{\epsilon_L^r}||}_{\text{Spatial Error}} + \underbrace{||Q_{\epsilon_0^r} - Q^{SC}_{M_0,\epsilon_0^r}|| + \sum_{\ell=1}^{L} ||(Q_{\epsilon_\ell^r} - Q_{\epsilon_{\ell-1}^r}) - (Q^{SC}_{M_\ell,\epsilon_\ell^r} - Q_{M_\ell,\epsilon_{\ell-1}^r})||}_{\text{Multilevel Sampling Error}}. \quad (3.6)$$

The implementation of our method will be very similar to the one proposed in [**?**] essentially we have added a reduced basis and replaced the parts dealing with mesh refinement with reduced basis refinement. We formally state our algorithm below We end this section by making a few comments of the above algorithm. In our current implementation we use the optimization procedure outlined in [**?**] in order to minimize $C(Q^{MLSC}_{\{M_\ell\},\{\epsilon_\ell^r\}})$, in the framework we are working there should be room to improve on this method. This is due to the fact that our reduced basis construction is hierarchical, therefore it is possible to look at all different levels of error tolerances and the associated reduced basis systems. This topic will be researched further in the future.

4. Numerical experiments. The goal of this section is to illustrate the computational speed up we will see via the multilevel reduced basis algorithm. Four our example we will consider the elliptic problem in one spatial dimension

$$\begin{aligned} -(a(x,\mathbf{y})u(x,\mathbf{y})')' &= 1 \quad x \in (0,1), \mathbf{y} \in \Gamma = [-1,1] \\ u(0,\mathbf{y}) &= u(1,\mathbf{y}) = 0, \end{aligned} \quad (4.1)$$

We will let $a(x,\mathbf{y}) = 4 + y_1 + 0.2\sin(\pi x)y_2 + 0.04\sin(2\pi x)y_3 + 0.008\sin(3\pi x)y_4$ and consider 4 different fixed spatial refinements $h = \frac{1}{500}, \frac{1}{1000}, \frac{1}{2000}, \frac{1}{4000}$. For each spatial refinement we will compare the cost of our methods calculated at a sequence of different error tolerances $\epsilon = 10^{-3}, 10^{-4}, 10^{-5}, 10^{-6}$ versus just a Clenshaw Curtis sparse grid method. In order to calculate sample paths of our PDE we will utilize a finite element method with piecewise linear polynomials.

We see from our results for smaller meshes it is possible that our multilevel method may actually be more expensive than the full method due the to overhead from constructing the reduced basis. It is even possible for coarse enough of a mesh and a small tolerance ϵ that the resulting reduced basis system may be more expensive to solve than the full system. On the other hand we see as our mesh becomes finer that we will begin to witness a very large cost savings from the reduced basis multilevel method. This is related to the fact that there appears to be a very weak scaling between the number of reduced basis vectors needed and the size of the mesh. In other words for a mesh size $h = \frac{1}{500}$ and tolerance $\epsilon = 10^{-6}$ we may need 8 reduced basis vectors, while for a mesh size $h = \frac{1}{4000}$ we may only need 9 reduced basis vectors.

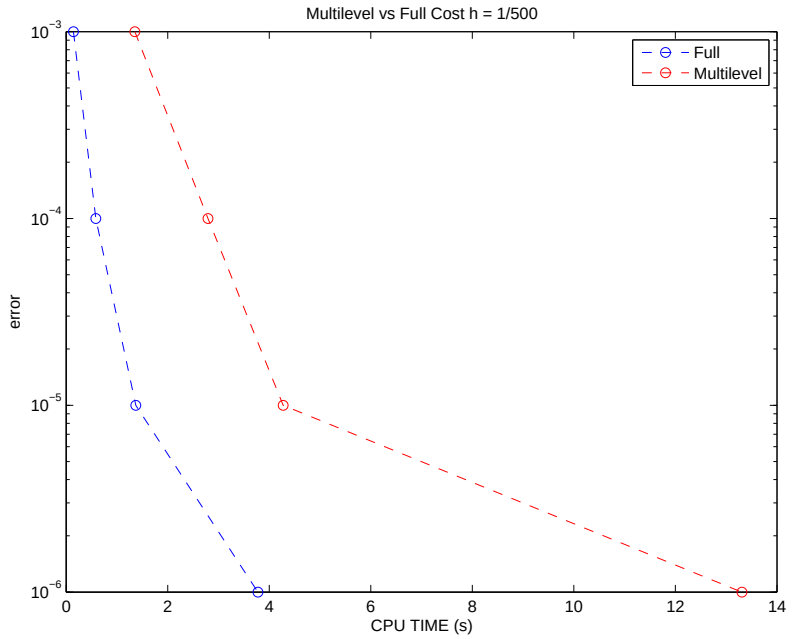

Fig. 4.1: The computational cost of the multilevel and full collocation versus CPU time for problem (4.1) $h = \frac{1}{500}$

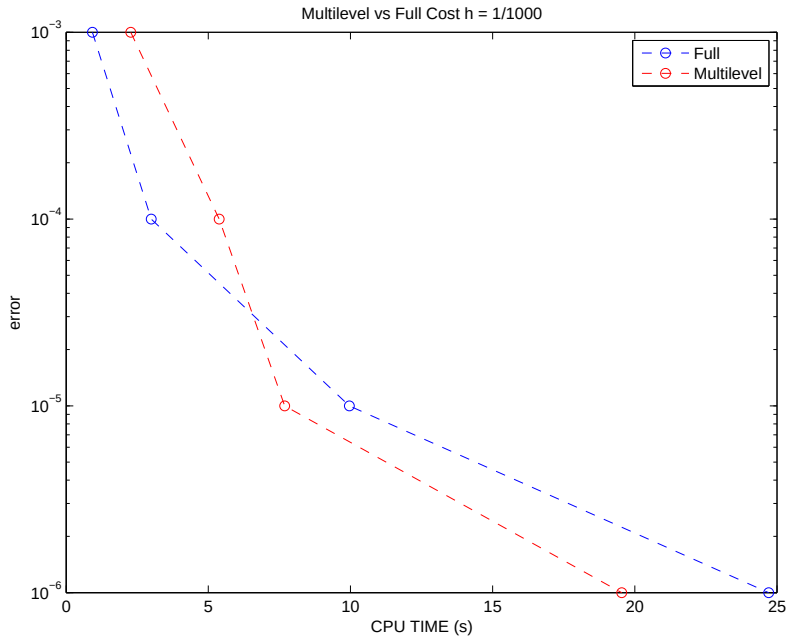

Fig. 4.2: The computational cost of the multilevel and full collocation versus CPU time for problem (4.1) $h = \frac{1}{1000}$

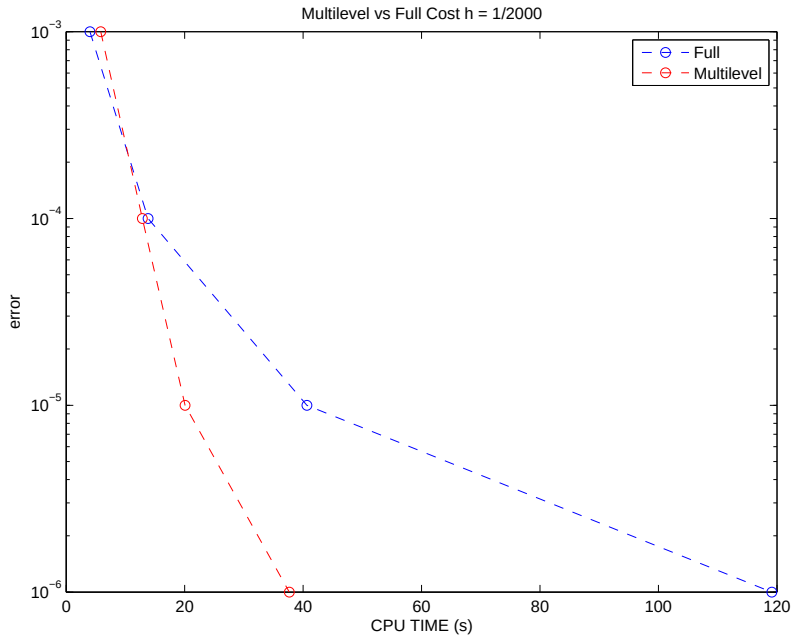

Fig. 4.3: The computational cost of the multilevel and full collocation versus CPU time for problem (4.1) $h = \frac{1}{2000}$

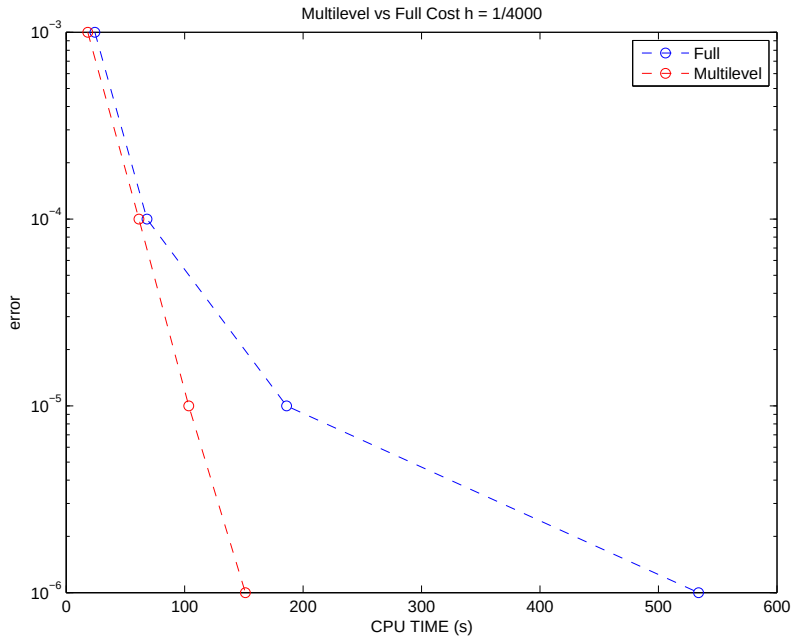

Fig. 4.4: The computational cost of the multilevel and full collocation versus CPU time for problem (4.1) $h = \frac{1}{4000}$

Algorithm 1 Multilevel Reduced Basis

Determine desired accuracy ϵ and a sequence of less accurate tolerances $\epsilon_{tol}^{j}(j = 1...J)$
procedure (Offline construction)
  Initialization: $\epsilon_{tol}^{j}(j = 1...J)$,mesh, parameters, finite element functions ϕ_i, $1 \leq i \leq \mathcal{N}$
  Pre-compute and store stiffness matrix $A_k = A_k(\phi, \phi)$, $0 \leq k \leq K$ and vector $F(\phi)$;
  Pre-compute and store Ξ_{train} and α_{LB};
  Initialize $y^1 \in \Xi_{\text{train}}, S^1 = \{y^1\}, X_1 = \{\zeta_1\}, \zeta_1 = u(y^1)/||u(y^1)||_X$
  Compute and store $(C, C)_X, (C, \mathcal{L}_1^k)_X, (\mathcal{L}_1^k, \mathcal{L}_1^{k'})_X, 0 \leq k, k' \leq K$;
  j =1
  while $\triangle_{N-1}^{u}(y) \geq \epsilon_{tol}^{J}$ **do**
    Compute $\triangle_{N-1}^{u}(y) = ||\hat{g}(y)||_X/\alpha_{LB}(y)$
    Choose $y^N = \text{argmax}_{y \in \Xi_{\text{train}}} \triangle_{N-1}^{u}(y)$
    if $\triangle_{N-1}^{u}(y) \leq \epsilon_{\text{tol}}$ **then**
      $N_{\max} = N - 1$; **Break**;
    end if
    Set $S_N = S_{N-1} \cup y^N$ and compute $u(y^N)$;
    Orthogonalize $X_N = \text{span}\{\zeta_1, ..., \zeta_{N-1}, u(y^N)\}$;
    Compute and store $A_k(\zeta_m, \zeta_n)$ and $F(\zeta_N)$;
    Set $S_N = S_{N-1} \cup y^N$ and compute $u(y^N)$;
    Compute and store $(C, \mathcal{L}_N^k)_X, (\mathcal{L}_n^k, \mathcal{L}_{n'}^{k'})_X$
    if $\epsilon_{\text{j}} \leq \triangle_{N-1}^{u}(y)$ **then**
      Store $(C, \mathcal{L}_N^k)_X, (\mathcal{L}_n^k, \mathcal{L}_{n'}^{k'})_X$
      j = j + 1
    end if
  end while
end procedure

procedure (Multilevel)
  Set desired accuracy level ϵ
  Determine initial sample size M_0
  Generate sample $\{u_{\epsilon_0^r}(x, \boldsymbol{y}^m)\}_{m=1}^{M_0}$ and compute $Q_{\{M_0\},\{\epsilon_0^r\}}^{MLSC}$
  Set the ROM error estimate $e_0^{\text{ROM}} = 1$
  while $e_L^{ROM} > \frac{\epsilon}{2}$ **do**
    L = L + 1
    Consider the next ROM basis associated with ϵ_L^r
    Determine $\{M_0, ..., M_L\}$ so that $e_L^{ROM} + e_L^{Sample} < \epsilon$ while minimizing the cost $C(Q_{\{M_\ell\},\{\epsilon_\ell^r\}}^{MLSC})$
    Generate the samples $\{u_{\epsilon_\ell^r}(y^m)\}_{m=1}^{M_\ell}$
    Update the multilevel estimate
    compute e_L^{ROM}
  end while
end procedure