Available online at www.sciencedirect.com
ScienceDirect
Comput. Methods Appl. Mech. Engrg. 196 (2007) 1030–1047
Computer methods in applied mechanics and engineering
www.elsevier.com/locate/cma

Reduced-order modeling of time-dependent PDEs with multiple parameters in the boundary data

Max D. Gunzburger [a,*,1], Janet S. Peterson [a,1], John N. Shadid [b,2]

[a] School of Computational Science, Florida State University, Tallahassee, FL 32306-4120, United States
[b] Computational Sciences Department, Sandia National Laboratories, P.O. Box 5800, MS 0316, Albuquerque, NM 87185-0316, United States

Received 28 October 2005; received in revised form 24 July 2006; accepted 2 August 2006

Abstract

The computational approximation of solutions to nonlinear partial differential equations (PDEs) such as the Navier–Stokes equations is often a formidable task. For this reason, there is significant interest in the development of very low-dimensional models that can be used to determine reasonably accurate approximations in simulation and control problems for PDEs. Many concrete tests have been reported on in the literature that show that several classes of reduced-order models (ROMs) are effective, i.e., an accurate approximate solution can be obtained using very low-dimensional ROMs. Most of these tests involved problems for which the solution depends on only a single parameter appearing in the boundary data. The extension of the techniques used in those tests to the case of multiple parameters is not a straightforward matter. Here, we present, test, and compare two methods for treating, within the context of a class of ROMs, inhomogeneous Dirichlet-type boundary conditions that contain multiple parameters. In this study, we focus on the proper orthogonal decomposition (POD) approach to ROM; however, the issues and results discussed here in the POD context apply equally well to other ROM approaches.

Keywords: Reduced-order modeling; Multiple parameters; Nonlinear partial differential equations

1. Introduction

Computing approximations of solutions of nonlinear partial differential equations (PDEs) is a computationally intensive task. For realistic simulations, many thousands or even millions of degrees of freedom are often required to obtain useful approximations. Thus, if one needs to do multiple simulations or to do a simulation in real time, the use of traditional discretization methods, e.g., finite element,[3] finite volume, or spectral methods, may not be feasible. Consequently, it is not surprising that there is intensive interest in any means by which one can reduce the cost of obtaining state solutions. That is the goal of reduced-order modeling (ROM) of the type we consider here.

ROM means different things to different people, and, as a broad class of algorithms, has been and remains a subject of active research in fields such as linear algebra, statistics, information science, ordinary and partial differential equations, physics, and engineering, as an incomplete list. Thus, we need to define what we mean by reduced-order modeling.

For approximating the solution $u(t,\vec{x})$ of a nonlinear, time dependent PDE, the type of ROMs we consider

* Corresponding author. Tel.: +1 850 6447060; fax: +1 850 6440098.
E-mail addresses: gunzburg@csit.fsu.edu (M.D. Gunzburger), peterson@csit.fsu.edu (J.S. Peterson), jnshadi@sandia.gov (J.N. Shadid).
[1] Supported by Sandia National Laboratories under contracts 233519 and 406670.
[2] This work was funded by the Department of Homeland Security Advanced Scientific Computing Program. Sandia National Laboratories is a multiprogram laboratory operated by Sandia Corporation, a Lockheed-Martin Company, for the United States Department of Energy's National Nuclear Security Administration under contract DE-AC-94AL85000.

[3] In this paper, we will only discuss ROM related to finite element methods; most of what we have to say applies as well to other high-fidelity discretization methods.

0045-7825/$ - see front matter © 2006 Elsevier B.V. All rights reserved.
doi:10.1016/j.cma.2006.08.004

proceed as follows. One chooses a reduced basis $\{\phi_i(\vec{x})\}_{i=1}^{d}$; hopefully, d is very small compared to the usual number of functions used in a finite element approximation or the number of grid points used in a finite difference approximation. Next, one seeks an approximation $u_{\text{rom}}(t,\vec{x})$ to the state u of the form

$$u_{\text{rom}}(t,\vec{x}) = \sum_{j=1}^{d} a_j(t)\phi_j(\vec{x}) \in V \equiv \text{span}\{\phi_1,\ldots,\phi_d\}.$$

Then, one determines the coefficients a_j, $j=1,\ldots,d$, by solving the state equations in the set V, e.g., one could find a Galerkin solution of the state equations in a standard way, using V for the space of approximations. The cost of such a computation would be very small if d is small.

In optimal control or optimization settings, one is faced with the need to do multiple state solves during an iterative process that determines the optimal state and controls. If one approximates the state in the reduced, d-dimensional space V and if d is small, then the cost of each iteration of the optimizer would be very small relative to that using full, high-fidelity state approximations. In a feedback control setting one often needs to compute state solutions in real time; determining approximate state equations in the low-dimensional space V could make this feasible.

The construction of the type of ROMs we consider require the solution of high-fidelity, e.g., standard finite element or finite volume, discrete state and/or sensitivity equations which are therefore very expensive to obtain. The main idea is that these expensive calculations can be done off line before a state simulation or the optimization of the design parameters or feedback control is attempted. Moreover, one hopes that the cost of obtaining a reduced basis can be amortized over many state simulations or design or control problems.

For the type of ROMs we consider, the support of the basis functions is global. Therefore, approximating the solution of the PDE using any reduced basis requires the solution of dense linear and nonlinear systems; thus, unless the dimension of a reduced basis is "small", ROMs will not be effective in practice.

The question remains about how one determines a reduced basis? There have been several approaches suggested for carrying out this task. There is the "reduced basis method" developed in, e.g., [28–31,39] which has been extensively studied in recent years (see [2,19,48]); for the most basic of this class of methods, a reduced basis is determined by computing finite element approximations of the solutions of the nonlinear PDE for a sampling of parameter values. A second approach is to use centroidal Voronoi tessellations of snapshot sets[4] to determine the reduced basis; see, e.g., [9–17,24,32]. In this approach, a clustering method is used to remove redundant information from the snapshot set, resulting in a low-dimensional reduced basis. Perhaps the most popular approach for the reduced-order modeling of nonlinear PDEs is the proper orthogonal decomposition (POD) of snapshot sets; see, e.g., [1,3–8,18,22,23,25–27,33–38,40–44,46,47,49,50] and the references cited therein. In this approach, a projection method is used to extract important information from the snapshot set, again resulting in a low-dimensional reduced basis.

In this paper, we only consider the POD approach. This keeps the exposition relatively simple. However, the issues and methods discussed in the paper apply equally well to other ROM approaches and in particular, to those mentioned in the previous paragraph. In Section 2, we provide a brief review of the POD approach to ROM.

The main focus of this paper is on how one handles, within the ROM context, boundary conditions containing multiple parameters. The great majority of tests of, e.g., POD-based ROM reported in the literature consider problems containing a single, possibly time dependent, parameter in the boundary data. These tests show that in the context of the problems addressed, POD-based ROM is often effective, i.e., an accurate ROM approximate solution can be obtained using very small dimensional POD bases. Extending the techniques used in the one-parameter case to the multiple parameter setting is not totally straightforward. In Section 3, we present two methods for handling inhomogeneous Dirichlet-type boundary conditions that involve multiple parameters. The two methods mimic, in the ROM setting, two standard approaches (which we also review in Section 3) for treating inhomogeneous essential boundary conditions in the finite element setting. The fact that the finite element functions are locally supported while reduced-order basis functions are globally supported have important consequences on how the two methods are implemented and on their relative merits.

In Section 4, we apply the two methods for handling multiple parameters in the boundary conditions to some specific nonlinear PDE problems. These problems are used not only as a test bed for comparing the two methods, but also to provide a context for a discussion of important issues encountered when the two methods are implemented.

2. POD-based model reduction

POD-based model reduction proceeds in three stages. First, one collects samples of solutions of the nonlinear PDE to produce a set of snapshots which, hopefully, will accurately represent the dynamical behavior of those solutions. One then uses the set of snapshots to determine a POD basis which, one again hopes, can accurately capture the information contained in the snapshot set using a set of much smaller cardinality. Finally, the POD basis is used to determine approximations to the solution of the PDE, in this way hopefully obtaining accurate approximations at a much lower cost than that incurred when using, e.g., a standard finite element method. Here, for the sake of completeness, we briefly review the three stages. Details about

[4] Snapshot sets are discussed in Sections 2.1 and 4.1.

POD reduced-order modeling and its variants may be found in the references cited above.

2.1. Snapshot sets

Snapshot sets consist of (approximate) solutions of the PDE corresponding to several sets of parameter values appearing in the problem specification and/or evaluated at several time instants during the evolution process. Snapshot sets are usually determined by solving a full, very large-dimensional discretized system obtained by, e.g., a finite volume or finite element discretization.

Snapshots sets must contain sufficient information to accurately represent the dynamics of solutions of the PDE. In order to do so, one usually has to determine snapshot sets containing hundreds or even thousands of elements. On the other hand, snapshot sets often contain much redundant information in the sense that one can, in principle, represent the information contained in the snapshot set with sets of much smaller cardinality. POD analysis is one means for attempting to remove redundancies in snapshot sets.

It should be apparent that snapshot sets are crucial to the effectiveness of POD reduced-order modeling, so that some care should be exercised in the determination of those sets. Unfortunately, at this time, there are no systematic and provably effective procedures available for generating snapshot sets. For example, although snapshot generation is an exercise in the design of experiments, very little use of the large body of statistics literature on that subject has been used. Instead, ad hoc procedures are invoked in the hope that they will lead to good snapshot sets. For example, in order to generate "rich transients" in the solution, impulsive forcing is commonly used, e.g., starting the evolution impulsively and/or introducing impulsive changes in the data in the middle of a simulation. Any a priori knowledge available about the types of states to be simulated using the reduced-order model can prove useful in snapshot generation. Certainly, systematizing the snapshot generation process should be a subject of substantial interest for some time to come; after all, a POD basis can only be as good as the snapshot set used to generate it.

2.2. POD bases

In the finite element context, one can view approximate solutions as vectors in Euclidean space or as functions in some finite-dimensional finite element space. The two views, of course, are related; the components of a vector are the coefficients in an expansion of a corresponding function in terms of a basis for the finite element space. Since snapshots are approximate solutions of the PDE, they can be viewed in these two ways. POD bases can be generated using either view; here, for simplicity, we choose to view them as vectors in Euclidean space.

We begin with n snapshots $\mathbf{s}_j \in \Re^N, j = 1, \ldots, n$. Here, N denotes the dimension of the finite element approximating space. Let $d \leqslant n < N$. Let S denote the $N \times n$ snapshot matrix whose columns are the snapshots $\mathbf{s}_j$, i.e.,

$$S = (\,\mathbf{s}_1 \quad \mathbf{s}_2 \quad \cdots \quad \mathbf{s}_n\,).$$

Let $S = U\Sigma V^{\mathrm{T}}$ denote the singular value decomposition of S. Then, the POD basis $\{\boldsymbol{\phi}_i\}_{i=1}^{d}$ of cardinality d is given by the first d left singular vectors of the snapshot matrix S, i.e., $\boldsymbol{\phi}_i = \mathbf{u}_i$ for $i = 1, \ldots, d$, where $\mathbf{u}_i$ denotes the ith column of U. By construction, a POD basis is orthonormal, i.e., $\boldsymbol{\phi}_i^{\mathrm{T}}\boldsymbol{\phi}_j = 0$ for $i \neq j$ and $\boldsymbol{\phi}_i^{\mathrm{T}}\boldsymbol{\phi}_i = 1$. Since $U\Sigma = SV$, it follows that the columns of U and therefore the POD basis vectors are linear combinations of the snapshots, i.e., of the columns of S.

POD is closely related to the statistical methods known as Karhunen–Loève analysis, the method of empirical orthogonal eigenfunctions, and principal component analysis.

POD bases are optimal in the following sense. Let $\{\boldsymbol{\psi}_i\}_{i=1}^{n}$ denote an arbitrary orthonormal basis for the span of the snapshot set $\{\mathbf{s}_j\}_{j=1}^{n}$. Let $P_{\psi,d}\mathbf{s}_j$ denote the projection of the snapshot $\mathbf{s}_j$ onto the d-dimensional subspace spanned by $\{\boldsymbol{\psi}_i\}_{i=1}^{d}$. Clearly, we have, for each $j = 1, \ldots, n$, $P_{\psi,d}\mathbf{s}_j = \sum_{i=1}^{d} c_{ji}\boldsymbol{\psi}_i$, where $c_{ji} = \boldsymbol{\psi}_i^{\mathrm{T}}\mathbf{s}_j$ for $i = 1, \ldots, d$. Let the error be defined by

$$\mathscr{E} = \sum_{j=1}^{n} |\mathbf{s}_j - P_{\psi,d}\mathbf{s}_j|^2.$$

Then, the minimum error is obtained when $\boldsymbol{\psi}_i = \boldsymbol{\phi}_i$ for $i = 1, \ldots, d$, i.e., when the $\boldsymbol{\psi}_i$'s are the POD basis vectors.

It is easy to show that the error of the d-dimensional POD subspace is given by

$$\mathscr{E}_{\mathrm{pod}} = \sum_{j=d+1}^{n} \sigma_j^2,$$

where $\{\sigma_j\}_{j=1}^{n}$ denote the singular values of S. If one would like the relative error to be less than a prescribed tolerance δ, i.e., if one wants $\mathscr{E}_{\mathrm{pod}} \leqslant \delta\sum_{j=1}^{n}|\mathbf{s}_j|^2$, one should choose d to be the smallest integer such that

$$\frac{\sum_{j=1}^{d}\sigma_j^2}{\sum_{j=1}^{n}\sigma_j^2} \geqslant 1 - \delta.$$

2.3. POD-based reduced-order modeling

It is more convenient to now switch to the function view of snapshots and POD basis vectors. Thus, with each POD basis vector $\boldsymbol{\phi}_i$, $i = 1, \ldots, d$, we associate the corresponding finite element function $\phi_i(\vec{x})$ determined by using the components of $\boldsymbol{\phi}_i$ as the coefficients in the expansion of ϕ_i in terms of the chosen basis for the finite element space.

We now consider how the POD basis $\{\phi_i\}_{i=1}^{d}$ is used to determine a reduced-order approximation of the PDE. For the sake of concreteness, suppose we wish to solve the problem:

given $\theta \in \Theta$, find $u(t,\vec{x}) \in U$ such that

$$N(u;\theta;v) = 0 \quad \forall v \in U, \text{ a.e. } t \in (0,T), \tag{1}$$

where U and Θ are given solution and parameter function spaces, respectively, and $(0,T)$ is a specified time interval; (1) is meant to represent a weak formulation of a system of nonlinear partial differential equations. The mapping $N: U \times \Theta \times U$ is linear in its third argument and is generally nonlinear in its first two arguments. In (1), θ denotes a set of parameters that serve to specify the problem, u denotes the desired solution, and v denotes a suitable test function. Thus, (1) implicitly defines a solution mapping $u(\theta): \Theta \to U$. The object of a computational simulation is to find approximations of this mapping.

A finite element method could be used to find approximate solutions of (1). In general, for PDEs of practical interest, the dimension of the approximating finite element space may be in the many thousands or even millions. As a result, it may be very expensive to solve the finite element discrete system for a single set θ of parameter values and it may be prohibitively expensive to solve it for many sets of parameter values. The latter is exactly one of the situations that cause the interest in ROM. So, instead of using standard finite element bases, we use POD reduced bases to find approximations of the solutions of (1).

We let $U_{\text{pod}} = \text{span}\{\phi_i\}_{i=1}^{d}$ denote the POD-reduced basis space. Then, a reduced-order solution of (1) is found by solving the problem

given $\theta \in \Theta$, find $u_{\text{pod}}(t,\vec{x}) \in U_{\text{pod}}$ such that

$$N(u_{\text{pod}};\theta;v) = 0 \quad \forall v \in U_{\text{pod}}, \text{ a.e. } t \in (0,T). \tag{2}$$

Since the reduced-order solution has the form $u_{\text{pod}} = \sum_{j=1}^{d} c_j(t)\phi_j(\vec{x})$ for some time dependent functions c_j, $j = 1,\ldots,d$, we have that (2) is equivalent to

given $\theta \in \Theta$, find $c_j(t), j = 1,\ldots,d$,

$$\text{such that } N\left(\sum_{j=1}^{d} c_j\phi_j;\theta;\phi_i\right) = 0 \quad \text{for } i = 1,2,\ldots,d,\ t \in (0,T). \tag{3}$$

Since, so far, we have only effected spatial discretization, (3) is a system of nonlinear ordinary differential equations for $c_j(t)$, $j = 1,\ldots,d$. Thus, further discretization in time is usually needed to solve (3).

POD bases often (or at least sometimes) do well at capturing the information contained in the snapshot set; in fact, they are designed to do just that. It is important to note that this does not necessarily imply that POD bases do well at approximating solutions of the PDE. We have that

$$\|u - u_{\text{pod}}\| \leqslant \|u_{\text{snapshot}} - u_{\text{pod}}\| + \|u - u_{\text{snapshot}}\|,$$

where u_{snapshot} can be viewed as the best approximation of u that can be obtained using linear combinations of the snapshots. POD reduced-order modeling often renders $\|u_{\text{snapshot}} - u_{\text{pod}}\|$ small; to make $\|u - u_{\text{snapshot}}\|$ small, one has to have "good" snapshots to begin with. The last observation is particularly relevant since most often one is interested in determining a reduced-order approximation u_{pod} for a different set of parameter values θ than those used in determining the snapshots.

3. Handling inhomogeneous essential boundary conditions

POD and other ROM techniques have been used extensively in a finite element setting to produce accurate approximations to solutions of nonlinear PDEs. However, the problems treated have usually either involved homogeneous Dirichlet boundary data or inhomogeneous Dirichlet boundary conditions with data containing a single multiplicative (possibly time-dependent) parameter. For example, [5,6] treated the time-dependent, incompressible Navier–Stokes equations with boundary conditions that included an inflow shape function containing a single multiplicative parameter which controls the strength of the inflow. In this section, we present two methods for handling problems with inhomogeneous Dirichlet boundary data that contain $K \geqslant 1$ multiplicative parameters; we refer to this problem as a multi-parameter or K-parameter problem. The two approaches apply not only to POD-based ROM, but to other settings, e.g., CVT-based ROM, as well.

The two approaches for treating boundary conditions in the ROM setting mimic the standard approaches used in finite element methodologies; we briefly review the approaches in the latter context.

3.1. Finite element approaches

For the sake of concreteness, we denote by $\{\vec{x}_\ell\}_{\ell=1}^{M+K}$ the nodes of a triangulation of the domain over which the partial differential equation is posed. We also use a nodal finite element space so that the finite element basis functions $\{v_\ell^h(\vec{x})\}_{\ell=1}^{M+K}$ satisfy $v_\ell^h(\vec{x}_m) = \delta_{\ell m}$. We assume, without loss of generality, that the first M basis functions correspond to interior nodes and that the last K basis functions correspond to nodes on the boundary.

In the first finite element approach, one proceeds as follows.

METHOD A

- The approximate solution at the time instant t_n is chosen to have the form

$$u^h(t_n,\vec{x}) = u^p(t_n,\vec{x}) + \sum_{\ell=1}^{M} a_\ell(t_n)v_\ell^h(\vec{x}),$$

  where u^p is a particular discrete solution chosen to approximately satisfy the boundary conditions and $\{a_\ell\}_{\ell=1}^{M}$ are the unknown, possibly time dependent, coefficients to be determined;
- the test functions are chosen to be $\{v_\ell^h(\vec{x})\}_{\ell=1}^{M}$ so that they *vanish on the boundary*.

Due to the local nature of the nodal basis functions, a particular solution u^p is easy to define. For example, if

the boundary condition has the form $u = g$, one can choose u^p to be

$$u^p(t_n,\vec{x}) = \sum_{\ell=M+1}^{M+K} g(t_n,\vec{x}_\ell)v_\ell^h(\vec{x}), \tag{4}$$

i.e., u^p is the boundary interpolant of the boundary condition data g. Note that for this choice, u^p is not a solution of the discretized partial differential equation.

In the second finite element approach, one proceeds as follows.

METHOD B

- The approximate solution at the time instant t_n is chosen to have the form

$$u^h(t_n,\vec{x}) = \sum_{\ell=1}^{M+K} b_\ell(t_n)v_\ell^h(\vec{x}), \tag{5}$$

  where $\{b_\ell\}_{\ell=1}^{M+K}$ are the unknown, possibly time dependent, coefficients to be determined;
- the test functions are given by $\{v_\ell^h(\vec{x})\}_{\ell=1}^{M}$ so that they vanish on the boundary;
- one adds K equations such as $b_\ell(t_n) = u^h(t_n,\vec{x}_\ell) = g(t_n,\vec{x}_\ell)$, $\ell = M+1,\ldots,M+K$, to the discretized system; for a nodal basis, these equations merely state that $b_\ell(t_n) = g(t_n,\vec{x}_\ell)$, $\ell = M+1,\ldots,M+K$.

Clearly, in the setting just described, the two approaches which respectively involve M and $M+K$ unknown coefficients are equivalent.

It will be profitable in later discussion to describe Method B in a slightly different, albeit convoluted, manner. We still use the form (5) for the approximate solution, but we start by testing against all the basis functions $\{v_\ell^h(\vec{x})\}_{\ell=1}^{M+K}$ so that *they do not all vanish on the boundary.* We then append K equations such as $b_\ell(t_n) = g(t_n,\vec{x}_\ell)$, $\ell = M+1,\ldots,M+K$. We now have $M+2K$ equations in the $M+K$ unknowns $\{b_\ell\}_{\ell=1}^{M+K}$. So, we have to *delete K equations*; clearly, we delete those that correspond to the test functions $\{v_\ell^h\}_{\ell=M+1}^{M+K}$ that do not vanish on the boundary. Doing so leads to exactly the same system as described above for Method B.

3.2. Reduced-order modeling approaches

We now want to mimic the two finite element approaches in the reduced-order setting. However, the fact that the reduced-order basis functions are global in nature causes difficulties. For the second approach, none of the POD basis functions vanish on the boundary so that we cannot separate them into two sets according to whether or not they vanish on the boundary. For the first approach, one cannot write the particular solution as a linear combination of POD basis functions.

We will use a concrete setting to describe the two methods. Suppose we use a POD-based reduced-order modeling technique to obtain an approximation $u_{\rm pod}(t_n,\vec{x})$ to the solution $u(t,\vec{x})$ of a nonlinear partial differential equation defined in a domain Ω with boundary Γ and evaluated at the time t_n. We suppose that the boundary conditions are given by

$$u(t,\vec{x}) = \begin{cases} \beta_k(t)g_k(\vec{x}) & \text{on } \Gamma_k, k = 1,\ldots,K, \\ 0 & \text{on } \Gamma - \bigcup_{k=1}^{K}\Gamma_k, \end{cases} \tag{6}$$

where $\Gamma_k \cap \Gamma_\ell = \emptyset$ if $k \neq \ell$ and $\bigcup_{k=1}^{K}\Gamma_k$ may be a portion of the boundary Γ or the entire boundary. The functions $\{g_k(\vec{x})\}_{k=1}^{K}$ are assumed given so that there are K (time-dependent) parameters $\{\beta_k\}_{k=1}^{K}$ that serve to specify the problem.

3.2.1. Method I

The first approach to treating inhomogeneous Dirichlet boundary conditions in ROM applications can be viewed as a straightforward generalization of the approach extensively used for the one-parameter case; see, e.g., [5,6]. It also, in some sense, mimics the first approach, i.e., Method A, discussed above in the context of standard, nodal finite element methods. Here, one uses reduced-order basis functions that satisfy homogeneous boundary data. The reduced-order solution is then written as a linear combination of these basis vectors plus another linear combination of particular solutions of the PDE that is specifically chosen to satisfy the desired inhomogeneous boundary conditions.

METHOD I

(i) Choose K vectors $\vec{\alpha}_k \in \mathbb{R}^K$, $k = 1,\ldots,K$; for $i = 1,\ldots,K$, let $\alpha_{k,i}$ denote the components of $\vec{\alpha}_k$;
(ii) generate K particular solutions u_k^p, $k = 1,\ldots,K$, where $u_k^p(\vec{x}) = \alpha_{k,i}g_i(\vec{x})$ on Γ_i, $i = 1,\ldots,K$;
(iii) generate N snapshots $s_j(\vec{x})$, $j = 1,\ldots,N$, satisfying $s_j = \sigma_{j,i}g_i(\vec{x})$ on Γ_i, $i = 1,\ldots,K$, for some $\vec{\sigma}_j \in \mathbb{R}^K$, $j = 1,\ldots,N$;
(iv) form modified snapshots $\tilde{s}_j$, $j = 1,\ldots,N$, that satisfy homogeneous boundary conditions on all of Γ by subtracting from each snapshot s_j the linear combination of particular solutions u_k^p, $k = 1,\ldots,K$, that have the same boundary data as does s_j;
(v) generate a POD reduced-order basis $\tilde{\phi}_i$, $i = 1,\ldots,d$, from the modified snapshots $\tilde{s}_j$, $j = 1,\ldots,N$; the resulting reduced-basis functions satisfy homogeneous boundary conditions on all of Γ;
(vi) at each time level t_n, determine the appropriate linear combination $\sum_{k=1}^{K}\eta_{n,k}u_k^p$ of the particular solutions $u_k^p(\vec{x})$ that satisfy the given boundary conditions $\beta_k(t_n)g_k(\vec{x})$ on Γ_k, $k = 1,\ldots,K$;
(vii) at each time level t_n, set

$$u_{\rm rom}(t_n,\vec{x}) = \sum_{k=1}^{K}\eta_{n,k}u_k^p(\vec{x}) + \sum_{i=1}^{d}a_{n,i}\tilde{\phi}_i(\vec{x}) \tag{7}$$

and solve the discrete weak problem tested against each $\tilde{\phi}_i$, $i = 1,\ldots,d$, to determine $a_{n,i}$, $i = 1,\ldots,d$.

We leave the discussion of how one chooses the $\vec{\sigma}_j$'s that are used to generate the snapshots and how one chooses the $\vec{\alpha}_k$'s and $u_k^p(\vec{x})$'s that are used for forcing the modified snapshots to satisfy homogeneous boundary conditions and for the formation of u_{rom} until we consider, in Section 4, some computational examples.[5] These choices can have a significant effect on the quality of the reduced-order solution. We also leave to that section details about how one implements step (vi) in practice. As discussed in Section 2.1, the snapshots are usually generated at different time levels for a few choices of the parameters σ_j. The POD reduced-order basis can be generated as described in Section 2.2.[6]

3.2.2. Method II

The second approach mimics Method B of Section 3.1 for handling inhomogeneous boundary conditions in the finite element case. Here, we set $u_{\text{rom}}(t_n,\vec{x}) = \sum_{i=1}^{m} b_{n,i}\phi_i$, where the POD basis functions ϕ_i, $i = 1,\ldots,m$, are globally supported and satisfy inhomogeneous boundary data.[7] We cannot satisfy the boundary conditions for our problem by simply relating a few of the coefficients $b_{n,i}$ to the boundary data because of the global character of the basis functions; this is unlike the case for nodal-based finite element functions. However, we can still add K equations to satisfy the K boundary conditions. The difficulty encountered is that we can no longer test our discrete weak problem against the m basis functions ϕ_i because we would obtain an overdetermined system of equations, i.e., we would have $m + K$ equations in m unknowns. The problem is that not only would this require a non-square reduced system of equations to be solved, with for example a least-squares type approach, but it would also not enforce the particular boundary conditions or the discretized PDE system.

We recall (see Method B of Section 3.1) that in the nodal finite element setting, we only test against functions which vanish on the boundary. Consequently, in the ROM setting, we want to test against $d = m - K$ linearly independent functions that vanish on the boundary and that are linear combinations of the original m basis functions $\{\phi_i\}_{i=1}^m$. The QR factorization algorithm is useful in this situation, as will be explained below. Note that we choose to describe the algorithm in terms of generating a POD reduced basis of cardinality $m = (d + K)$ instead of the d-dimensional basis set used in Method I. This is because when we compare the two approaches we want to compare results where the same number d of test functions are used in the discrete weak problem.[8]

METHOD II

(i) Generate N snapshots $s_j(\vec{x})$, $j = 1,\ldots,N$, satisfying $s_j = \sigma_{j,i} g_i(\vec{x})$ on Γ_i, $i = 1,\ldots,K$, for some $\vec{\sigma}_j \in \mathbb{R}^K$, $j = 1,\ldots,N$;
(ii) generate a POD reduced-order basis ϕ_i, $i = 1,\ldots,m$, from the snapshots s_j, $j = 1,\ldots,N$; the resulting reduced-basis functions satisfy inhomogeneous boundary conditions;
(iii) use the QR decomposition algorithm to determine the linear combinations ψ_ℓ, $\ell = 1,\ldots,d = m - K$ of the POD basis functions $\{\phi_i\}_{i=1}^m$ that vanish on the boundary;
(iv) at each time level t_n, set

$$u_{\text{rom}}(t_n,\vec{x}) = \sum_{i=1}^{m} b_{n,i}\phi_i(\vec{x}) \tag{8}$$

and solve the m-dimensional system formed by using, in the discrete weak formulation, the test functions ψ_i, $i = 1,\ldots,d = m - K$ plus the additional K equations

$$u_{\text{rom}}(t_n,\vec{x}_{\gamma_k}) = \sum_{i=1}^{m} b_{n,i}\phi_i(\vec{x}_{\gamma_k}) = \beta_k(t_n) g_k(\vec{x}_{\gamma_k}) \quad \text{for } k = 1,\ldots,K,$$

where $\vec{x}_{\gamma_k}$ is any point on Γ_k for which $g_k(\vec{x}_{\gamma_k}) \neq 0$.

To clarify step (iii), let $\vec{x}_{\gamma_k}$ be a point on Γ_k, $k = 1,\ldots,K$, and let B be the $m \times K$ matrix with entries given by

$$B_{ik} = \phi_i(\vec{x}_{\gamma_k}), \quad i = 1,\ldots,m, \;\; k = 1,\ldots,K.$$

We now want to determine $d = m - K$ vectors in $\mathbb{R}^d$ that are linearly independent and orthogonal to the span of columns of B; this can, of course, be done by performing a QR factorization of B; the last $d = m - K$ columns of the $m \times m$ orthogonal matrix Q determine the coefficients of the

[5] It is usually the case that once the $\vec{\alpha}_k$'s are chosen, the $u_k^p(\vec{x})$'s are computed as corresponding steady state solutions of the finite element system.

[6] Method I is a generalization of the standard approach in the ROM literature (see, e.g., [5,6] and the references cited therein) for handling inhomogeneous Dirichlet boundary data containing a single multiplicative parameter. For this case, we have $K = 1$ so that the inhomogeneous boundary condition is simply given by $u(\vec{x},t) = \beta(t)g(\vec{x})$ on Γ_1. Method I then reduces to the following procedure.

(i) Generate u^p which satisfies $u^p = \alpha g(\vec{x})$ on Γ for some choice of α;
(ii) generate snapshots s_j, $j = 1,\ldots,N$, that satisfy $u = \sigma_j g(\vec{x})$ for some σ_j, $j = 1,\ldots,N$;
(iii) form the modified snapshots $\tilde{s}_j$, $j = 1,\ldots,N$, satisfying homogeneous boundary data by setting $\tilde{s}_j = s_j - \frac{\sigma_j}{\alpha}u^p$, $j = 1,\ldots,N$;
(iv) generate a reduced order basis $\tilde{\phi}_i$, $i = 1,\ldots,d$, from the modified snapshots $\tilde{s}_j$, $j = 1,\ldots,N$;
(v) set $u_{\text{rom}}(\vec{x},t^n) = \frac{\beta(t^n)}{\alpha}u^p + \sum_{i=1}^{d}\mu_{n,i}\tilde{\phi}_i$ and solve the appropriate discrete weak problem tested against each $\tilde{\phi}_\ell$, $\ell = 1,\ldots,d$.

[7] The POD basis functions ϕ_i, $i = 1,\ldots,m$, do not satisfy the boundary conditions that are used to generate the snapshots nor are they solutions of the discretized finite element equations. This is entirely analogous with finite element basis functions which also do not satisfy the finite element equations or the boundary data of the problem being solved.

[8] Recall that for Method I, in addition to the d POD basis functions, we also have to determine K particular solutions so that for both Method I and Method II the respective reduced-order solutions (7) and (8) are linear combination of $m = d + K$ functions.

linear combinations ψ_ℓ, $\ell = 1,\ldots,d = m-K$, of the basis functions $\{\phi_i\}_{i=1}^m$ that we use as test functions. Specifically, for $\ell = 1,\ldots,d$, we set $\psi_\ell = \sum_{i=1}^m Q_{i,\ell+K}\phi_i$. Note that, due to the orthogonality of the columns of B with respect to the last d columns of Q, we have for $\ell = 1,\ldots,d = m-K$ and $k = 1,\ldots,K$ that

$$\psi_\ell(\vec{x}_{\gamma_k}) = \sum_{i=1}^m Q_{i,\ell+K}\phi_i(\vec{x}_{\gamma_k}) = \sum_{i=1}^m Q_{i,\ell+K}B_{ik} = 0$$

so that the linear combinations $\{\psi_\ell\}_{\ell=1}^d$ do indeed vanish on the boundary. Note also that the reduced-order system formed in step (iv) is given by

$$N_n\left(\sum_{i=1}^m b_{n,i}\phi_i;\theta;\psi_\ell\right) = 0 \quad \text{for } \ell = 1,\ldots,d = m-K$$

$$\sum_{i=1}^m b_{n,i}\phi_i(\vec{x}_{\gamma_k}) = \beta_k(t_n)g_k(\vec{x}_{\gamma_k}) \quad \text{for } k = 1,\ldots,K,$$

where $N_n(\cdot;\cdot)$ denotes the weak form of the nonlinear partial differential equation after discretization in time is effected.

4. Computational experiments

In this section we present the results of some computational experiments. We first consider as a simple prototype example a time dependent, nonlinear parabolic problem where the boundary data depend on either one, two, or four time-dependent parameters. This example is used to compare the two approaches described in Section 3. A second example is taken from an application which models air flow in a large public building; in this example, the flow is governed by the time-dependent Navier–Stokes equations and the boundary data involve six parameters. For the second example, we only present results for Method II.

4.1. Example I

Consider the following nonlinear, parabolic partial differential equation

$$\frac{\partial u}{\partial t} - \Delta u + f(u) = 0 \quad \text{for } \vec{x} \in \Omega = (0,1)\times(0,1), \;\; 0 < t \leqslant T \tag{9}$$

for the scalar-valued function $u(t,\vec{x})$, where $u(0,\vec{x}) = 0$, T is a given terminal time, and $f(\cdot)$ is a given nonlinear function; specifically, we present computational results for $f(u) = u^2$ and $f(u) = e^u$. The boundary condition will take the form (6); the specific form is described below.

A weak formulation of this problem is to seek a $u \in L^2(0,T;H^1(\Omega))$ satisfying, for almost every $t \in (0,1)$,

$$\int_\Omega u_t v\,d\vec{x} + \int_\Omega \nabla u\cdot\nabla v\,d\vec{x} + \int_\Omega f(u)v\,d\vec{x} = 0 \quad \forall v \in H_0^1(\Omega),$$

where $u(0,\vec{x}) = 0$ and $u(t,\vec{x})$ satisfies the given boundary conditions. Here, $H^1(\Omega)$ denotes the standard Sobolev space of functions belonging to $L^2(\Omega)$ that possess one weak derivative belonging to $L^2(\Omega)$ and $H_0^1(\Omega) = \{w \in H^1(\Omega) : w|_\Gamma = 0\}$. The space $L^2(0,T;H^1(\Omega))$ consists of functions whose $H^1(\Omega)$-norm in space is square integrable with respect to time.

For the results presented here, we use a finite element method to discretize in space and the backward Euler method to discretize in time. Specifically, let $S^h \subset H^1(\Omega)$ denote the finite element space consisting of continuous, piecewise quadratic polynomials with respect to a triangulation of $\Omega = (0,1)\times(0,1)$. Let $S_0^h = S^h \cap H_0^1(\Omega)$. Then, the approximation $u_n^h \in S^h$ to the solution $u(t_n,\vec{x})$ of (9) is determined by solving, for $n = 1,2,\ldots$, the system

$$\frac{1}{\Delta t}\int_\Omega u_n^h v^h\,d\vec{x} + \int_\Omega \nabla u_n^h\cdot\nabla v^h\,d\vec{x} + \int_\Omega f(u_n^h)v^h\,d\vec{x} = \frac{1}{\Delta t}\int_\Omega u_{n-1}^h v^h\,d\vec{x} \tag{10}$$

for all $v^h \in S_0^h$, where $u_0^h = 0$ and appropriate discretized boundary conditions are imposed. The discrete system (10) is equivalent to a system of nonlinear equations that can be solved, e.g., by Newton's method.

We use a uniform grid of 128 triangles and 225 nodes and a fixed time step $\delta t = 0.01$ for generating particular solutions, snapshot functions used in determining the POD bases and, later, for computing the finite element approximations used to compare with the reduced-order solutions. ROM bases are generated using standard POD techniques.

4.1.1. Determining particular solutions and snapshot sets

Particular solutions of the finite element discrete system are used in two steps of Method I of Section 3.2.1. First, they are used in step (iv) to modify the snapshots so that they satisfy homogeneous boundary conditions and, second, they are used in step (vi) to force the reduced-order approximation to satisfy the given boundary conditions. We use the same particular solutions for both purposes. In the current context, we generate, for $k = 1,\ldots,K$, the particular solution $u_k^p(\vec{x})$ as steady state solutions of the finite element discrete system with the boundary condition (6), where we choose the parameter functions $\beta_\ell(t) = \alpha_{k,\ell}$ with $\alpha_{k,\ell} = \delta_{kl}$, $\ell = 1,\ldots,K$; here δ_{ij} denotes the Kronecker delta function. Thus, with respect to steps (i) and (ii) of Method I, the particular solution $u_k^p(\vec{x})$ is determined from the boundary data $\vec{\alpha}_k = (0\cdots010\cdots0)^{\mathrm{T}}$, where the 1 appears in the kth component.

To generate the snapshot set for both Methods I and II of Sections 3.2.1 and 3.2.2, we proceed as follows.

Snapshot generation

For $\ell = 1,\ldots,K$,

(i) choose the boundary parameter functions $\beta_k(t) = \sigma_{\ell,k}$, where $\sigma_{\ell,\ell} = 1$ and $\sigma_{\ell,k} = 0$ for $k = 1,\ldots,K$, $k \neq \ell$;

(ii) for $\ell = 1$, set the initial condition equal to zero and for $\ell > 1$, set the initial condition to the steady state solution obtained at the end of step (vi) for the previous value of ℓ;
(iii) solve the discrete finite element system until it reaches steady state and add the solutions at every time step to the snapshot set;
(iv) change the boundary parameter functions $\beta_k(t) = \sigma_{\ell,k}$ so that now $\sigma_{\ell,k} = 1$ for $k = 1, \ldots, K$;
(v) set the initial condition to the steady state solution obtained at the end of step (iii);
(vi) solve the discrete finite element system until it reaches steady state and add the solutions at every time step to the snapshot set.

Note that during the snapshot generation process described by this procedure, the boundary data $\sigma_{\ell,k}$ experiences several jump discontinuities. This follows the common practice of "jolting" the simulation during the snapshot generation process; such jolts generate "rich transient behavior" that is thought of as being desirable to include in the snapshot set.

For Method I, we need to modify the snapshots so that they have homogeneous boundary conditions. We denote by $\{s_j\}_{j=1}^N$ the snapshot functions generated by the above algorithm; we now denote the boundary data used to generate these snapshots by $\sigma_{\ell(j),k}$, where $\ell(j)$ denotes the value of ℓ in use when the jth snapshot was evaluated. This explicitly highlights the fact that the boundary data depends on time. We then set

$$\tilde{s}_j(t,x) = s_j(t,x) - \sum_{k=1}^{K} \sigma_{\ell(j),k} u_k^p(t,x) \quad \text{for } j = 1, \ldots, N.$$

By construction, the functions $\{\tilde{s}_j\}_{j=1}^N$ satisfy homogeneous boundary conditions.

To highlight the arbitrariness in the processes just described, we purposely did not motivate the choices made for the boundary parameter functions $\{\beta_k(t)\}_{k=1}^K$ used in the generation of particular solutions and snapshots sets. We will comment on those choices in Section 4.2.

4.1.2. The reduced-order models

To facilitate comparisons, we provide some details about the reduced-order models that result from the two approaches for handling inhomogeneous boundary conditions. The POD reduced-order bases are generated in a standard manner based on the modified snapshot set $\{\tilde{s}_j\}_{j=1}^N$ for Method I and the unmodified snapshot set $\{s_j\}_{j=1}^N$ for Method II. We now want to use the POD reduced-order bases to find solutions of (9) for parameter functions $\beta_k(t)$, $k = 1, \ldots, K$, appearing in the boundary condition (6) that are different from the ones used to generate particular solutions and snapshots sets.

The POD basis $\{\tilde{\phi}_i\}_{i=1}^d$ used in Method I satisfies homogeneous boundary conditions. Thus, the POD reduced-order approximation $u_{\text{rom}}(t_n,x)$ is given as in step (vi) of Method I, where at each time level n, the $\{\eta_{n,k}\}_{k=1}^K$ are determined so that the boundary conditions are satisfied. For the specific choices made in Section 4.1.1 for $\{u_k^p\}_{k=1}^K$, we have that at the time level n,

$$u_{\text{rom}}^n(\vec{x}) = u^{p,n}(\vec{x}) + w^n(\vec{x}), \tag{11}$$

where

$$u^{p,n}(\vec{x}) = \sum_{k=1}^{K} \beta_k(t_n) u_k^p(\vec{x}) \quad \text{and} \quad w^n(\vec{x}) = \sum_{i=1}^{d} a_{n,i} \tilde{\phi}_i(\vec{x}). \tag{12}$$

Then, at each time level n, the coefficients $a_{n,i}$ are determined by the Newton iteration: starting with $w^{n,0} = u_{\text{rom}}^{n-1} - u^{p,n-1}$, for $q = 1, 2, \ldots$, determine the Newton iterates $w^{n,q}$ by solving the linear systems

$$\begin{aligned}
&\frac{1}{\Delta t}\int_\Omega w^{n,q}\tilde{\phi}_i \,\mathrm{d}\vec{x} + \int_\Omega \nabla w^{n,q}\cdot\nabla\tilde{\phi}_i \,\mathrm{d}\vec{x} + \int_\Omega f'(u^{n,q-1}) w^{n,q}\tilde{\phi}_i \,\mathrm{d}\vec{x} \\
&\quad = \frac{1}{\Delta t}\int_\Omega u^{n-1}\tilde{\phi}_i \,\mathrm{d}\vec{x} - \frac{1}{\Delta t}\int_\Omega u^{p,n}\tilde{\phi}_i \,\mathrm{d}\vec{x} - \int_\Omega \nabla u^{p,n}\cdot\nabla\tilde{\phi}_i \,\mathrm{d}\vec{x} \\
&\qquad - \int_\Omega f'(u^{n,q-1}) u^{p,n}\tilde{\phi}_i \,\mathrm{d}\vec{x} + \int_\Omega f'(u^{n,q-1}) u^{n,q-1}\tilde{\phi}_i \,\mathrm{d}\vec{x} \\
&\qquad - \int_\Omega f(u^{n,q-1})\tilde{\phi}_i \,\mathrm{d}\vec{x},
\end{aligned} \tag{13}$$

where $u^{n,q-1} = u^{p,n} + w^{n,q-1}$. If $w^{n,\tilde{q}}$ denotes the converged solution of the above iteration, we set $u_{\text{rom}}^n = u^{p,n} + w^{n,\tilde{q}}$ and advance to the next time step.

The POD basis $\{\phi_i\}_{i=1}^m$ used in Method II satisfies inhomogeneous boundary conditions. The reduced-order solution is now simply given by

$$u_{\text{rom}}^n(\vec{x}) = \sum_{i=1}^{m} b_{n,i}\phi_i(\vec{x}). \tag{14}$$

At each time level n, we need to determine the functions $\{\psi_\ell\}_{\ell=1}^d$ (which are linear combinations of the basis functions $\{\phi_i\}_{i=1}^m$) as described in step (iii) of Method II. Then, at each time level n, the coefficients $b_{n,i}$ are determined by the Newton iteration: starting with $u^{n,0} = u_{\text{rom}}^{n-1}$, for $q = 1, 2, \ldots$, determine the Newton iterates $u^{n,q}$ by solving the linear systems

$$\begin{cases}
\dfrac{1}{\Delta t}\displaystyle\int_\Omega u^{n,q}\psi_\ell \,\mathrm{d}\vec{x} + \int_\Omega \nabla u^{n,q}\cdot\nabla\psi_\ell \,\mathrm{d}\vec{x} + \int_\Omega f'(u^{n,q-1}) u^{n,q}\psi_\ell \,\mathrm{d}\vec{x} \\
\quad = \displaystyle\int_\Omega f'(u^{n,q-1}) u^{n,q-1}\psi_\ell \,\mathrm{d}\vec{x} - \int_\Omega f(u^{n,q-1})\psi_\ell \,\mathrm{d}\vec{x} \\
\quad\quad + \dfrac{1}{\Delta t}\displaystyle\int_\Omega u^{n-1}\psi_\ell \,\mathrm{d}\vec{x} \quad \text{for } \ell = 1, \ldots, d \\
\displaystyle\sum_{i=1}^{m} b_{n,i}\phi_i(\vec{x}_{\gamma_k}) = \beta_k(t_n) g(\vec{x}_{\gamma_k}) \quad \text{for } k = 1, \ldots, K,
\end{cases} \tag{15}$$

where $\vec{x}_{\gamma_k}$ denotes any point on Γ_k for which $g(\vec{x}_{\gamma_k}) \neq 0$. If $u^{n,\tilde{q}}$ denotes the converged solution of the above iteration, we set $u_{\text{rom}}^n = u^{n,\tilde{q}}$ and advance to the next time step.

4.1.3. Computational experiments

We consider the partial differential equation (9) with the boundary condition (6) given by one of the following:

one-parameter problem:

$$u(t,\vec{x}) = \beta_1(t) 4x_1(1-x_1) \quad \text{on } \Gamma,$$

two-parameter problem:

$$u(t,\vec{x}) = \begin{cases} \beta_1(t) 4x_1(1-x_1) & \text{if } x_2 = 0 \text{ or } 1, \\ \beta_2(t) 4x_2(1-x_2) & \text{if } x_1 = 0 \text{ or } 1, \end{cases} \tag{16}$$

four-parameter problem:

$$u(t,\vec{x}) = \begin{cases} \beta_1(t) 4x_1(1-x_1) & \text{if } x_2 = 1, \\ \beta_2(t) 4x_1(1-x_1) & \text{if } x_2 = 0, \\ \beta_3(t) 4x_2(1-x_2) & \text{if } x_1 = 0, \\ \beta_4(t) 4x_2(1-x_2) & \text{if } x_1 = 1, \end{cases}$$

where $\vec{x} = (x_1, x_2)$. Particular solutions and snapshot sets are determined using the processes discussed in Section 4.1.1. The number of snapshots determined in this manner are 86, 156, and 300 for the one, two, and four-parameter problems, respectively. Of course, the respective number of particular solutions determined is $K = 1$, 2, and 4. For each problem, two different POD bases $\{\tilde{\phi}_i(\vec{x})\}_{i=1}^{16}$ and $\{\phi_i(\vec{x})\}_{i=1}^{16+K}$ corresponding to Methods I and II, respectively, are determined from the snapshot set and, in the case of Method I, the particular solutions $\{u_k^p\}_{k=1}^{K}$ are also used to modify the snapshots so that the resulting POD basis functions satisfy homogeneous boundary conditions. Note that POD bases are nested, i.e., a POD basis of cardinality d consists of the first d basis functions of the POD basis of cardinality $d+1$, so that we can use subsets of a POD basis of cardinality d to define reduced-order models of any dimension $\leqslant d$. For Method II, the boundary points $\vec{x}_{\gamma_k}$ at which the boundary conditions are enforced through the second equation of (15) are chosen to be the midpoints of the respective boundary segment Γ_k, e.g., in the four-parameter case, we enforce the first boundary condition at the point $(0.5, 1)$ so that we have that $\sum_{i=1}^{m} b_{n,i}\phi_i(0.5,1) = \beta_k(t_n)$ $g(0.5) = \beta_k(t_n)$.

We recall that the decay of the singular values of the snapshot matrix give an indication of how well POD bases approximate the information contained in the snapshot set. For the case of $f(u) = u^2$ the largest singular values of the snapshot matrix are plotted in Fig. 1 for both the snapshot set used for Method II and for the modified snapshot set satisfying homogeneous boundary conditions used for Method I. The figure presents results for the one, two, and four-parameter problems. Note the rapid decay of the singular values which tells us that small-dimensional POD bases should do well at capturing the information contained in the respective snapshot sets. However, we again caution that the singular values tell us nothing about how well the snapshot set itself can approximate the information contained in our "truth" model, i.e., in the full finite element model.

We now want to test the effectiveness of Methods I and II in approximating solutions of the finite element system. To do so, we choose parameter functions $\beta_i(t)$, $i = 1, \ldots, K$, and compare the solutions obtained using the finite element model with those obtained using POD reduced-order models. Specifically, we determine, at each time level,

$$\mathscr{E}(t_n) = \frac{\|u_h^n - u_{\text{rom}}^n\|_0}{\max\limits_{0 \leqslant t \leqslant 1} \|u_h^n\|_0}, \tag{17}$$

where $u_h^n(\vec{x})$ and $u_{\text{rom}}^n(\vec{x})$ denote the finite element and reduced-order solutions at time level n and $\|\cdot\|_0$ denotes the $L^2(\Omega)$ norm. For the finite element model, we use the same grid and time step as we used for the generation of particular solutions and snapshot sets.

The specific choices we use for the boundary parameter functions in (16) are as follows:

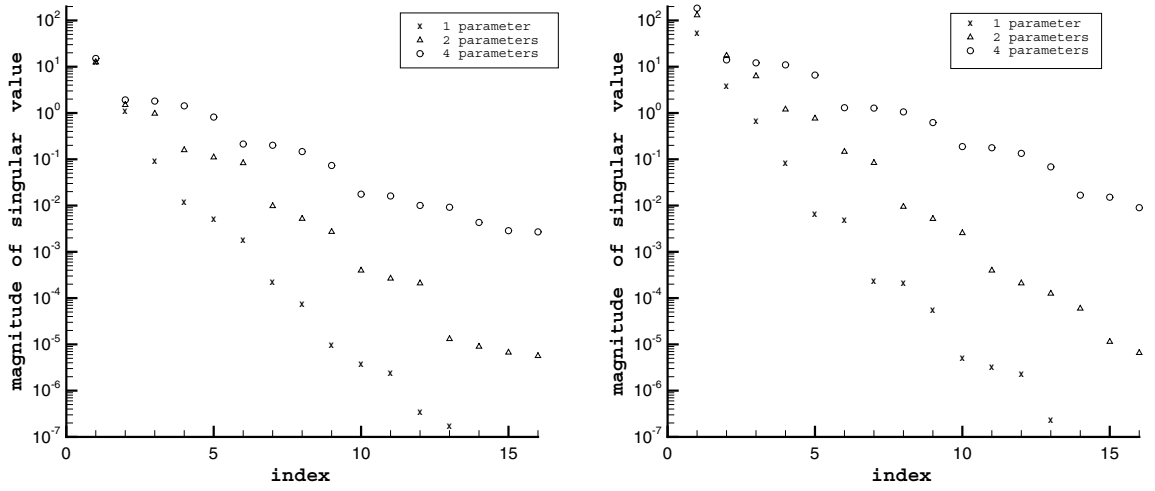

Fig. 1. The singular values of the modified snapshot set used for Method I (left) and the unmodified snapshot set used for Method II (right) for $f(u) = u^2$.

$$\begin{cases} \text{one-parameter problem} & \beta_1(t) = \begin{cases} 2t, & \text{for } 0 \leqslant t \leqslant 0.5, \\ 2(1-t) & \text{for } 0.5 \leqslant t \leqslant 1, \end{cases} \\ \text{two-parameter problem} & \begin{cases} \beta_1(t) = \begin{cases} 2t, & \text{for } 0 \leqslant t \leqslant 0.5, \\ 2(1-t) & \text{for } 0.5 \leqslant t \leqslant 1, \end{cases} \\ \beta_2(t) = |\sin 2\pi t| & \text{for } 0 \leqslant t \leqslant 1, \end{cases} \\ \text{four-parameter problem} & \begin{cases} \beta_1(t) = \begin{cases} 2t, & \text{for } 0 \leqslant t \leqslant 0.5, \\ 2(1-t) & \text{for } 0.5 \leqslant t \leqslant 1, \end{cases} \\ \beta_2(t) = 4(t-t^2) & \text{for } 0 \leqslant t \leqslant 1, \\ \beta_3(t) = |\sin 2\pi t| & \text{for } 0 \leqslant t \leqslant 1, \\ \beta_4(t) = |\sin 4\pi t| & \text{for } 0 \leqslant t \leqslant 1. \end{cases} \end{cases} \tag{18}$$

Note that these choices for the parameter functions are very different from the ones made to determine particular solutions and snapshot sets.

For each of the K-parameter problems, $K = 1$, 2, and 4, reduced-order solutions are determined using 2, 4, 8, and 16 modified POD basis vectors for Method I and $2 + K$, $4 + K$, $8 + K$, and $16 + K$ unmodified POD basis vectors for Method II; of course, for Method I, we also have that the reduced-order solutions involve linear combinations of K particular solutions. The error function $\mathscr{E}(t)$ is displayed in Figs. 2–4 for the one, two, and four-parameter problems, respectively.

For the nonlinearity $f(u) = e^u$, we proceed exactly as described above for the $f(u) = u^2$ case. This resulted in 288 snapshot functions. The error function $\mathscr{E}(t)$ for the four-parameter problem is displayed in Fig. 5 for both Methods I and II.

As can be seen from the figures, the two methods yield very comparable results, the errors at any reported time decrease as the number of basis vectors is increased, and a relatively small number of basis vectors provides a reasonably accurate approximation to the solution of the finite element model. The results for the one-parameter problem are similar to the ones reported in, e.g., [5,6].

4.2. Implementation issues and comparison of approaches

For the example considered in Section 4.1, there was little difference in the ROM results obtained through the two methods for handling boundary conditions containing multiple parameters. However, there are several issues that arise when one implements these methods that can affect the choice of which method one chooses to use in practice. Here, we discuss several of these issues.

4.2.1. Particular solutions in Method I

For Method I, one must determine the particular solutions $u_k^p(\vec{x})$, $k = 1, \ldots, K$, in the weighted sum $u^{p,n}(\vec{x}) = \sum_{k=1}^{K} \eta_{n,k} u_k^p(\vec{x})$; the coefficients $\eta_{n,k}$, $k = 1, \ldots, K$, are chosen so that $u^{p,n}(\vec{x})$ satisfies given boundary conditions at the time level t^n; see Section 3.2.1. Steps (i) and (ii) of Method I show how the particular solutions $u_k^p(\vec{x})$, $k = 1, \ldots, K$, are determined. One begins by choosing K vectors $\vec{\alpha}_k \in \mathbb{R}^K$ which are used to set the boundary conditions for $u_k^p(\vec{x})$. For example, for each $k = 1, \ldots, K$, we set, in the boundary condition (6), $\beta_\ell(t) = \vec{\alpha}_{k,\ell}$ for $\ell = 1, \ldots, K$ and $u_k^p(\vec{x})$ is determined by solving for the steady state finite element approximation of the solution of the PDE.

Clearly, choosing the parameter vectors $\{\vec{\alpha}_k\}_{k=1}^K$ is the fundamental step in determining the particular solutions. In Section 4.1.1, we chose $\alpha_{k,\ell} = \delta_{k\ell}$ for $k, \ell = 1, \ldots, K$, i.e., $\vec{\alpha}_k = \vec{e}_k$, where $\vec{e}_k$ is the kth unit vector in $\mathbb{R}^K$. This particular choice simplifies the determination of the weights $\eta_{n,k}$, $k = 1, \ldots, K$. These weights are normally determined by solving the $K \times K$ linear system

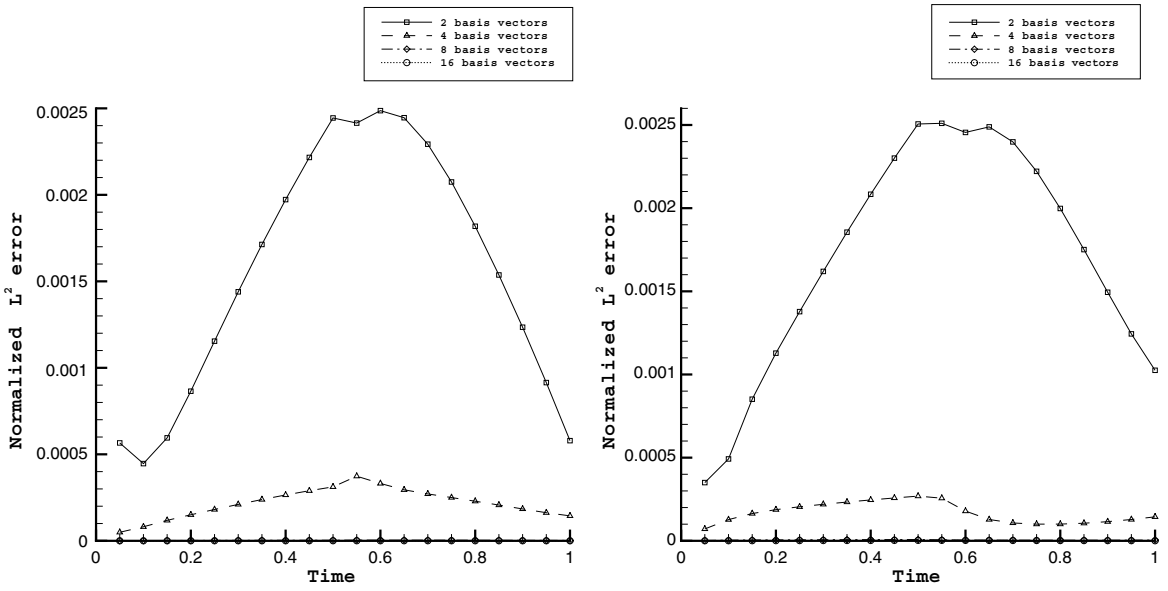

Fig. 2. The error function $\mathscr{E}(t)$ for the one-parameter problem for different numbers of POD basis vectors used in the ROM for $f(u) = u^2$; left: Method I; right: Method II.

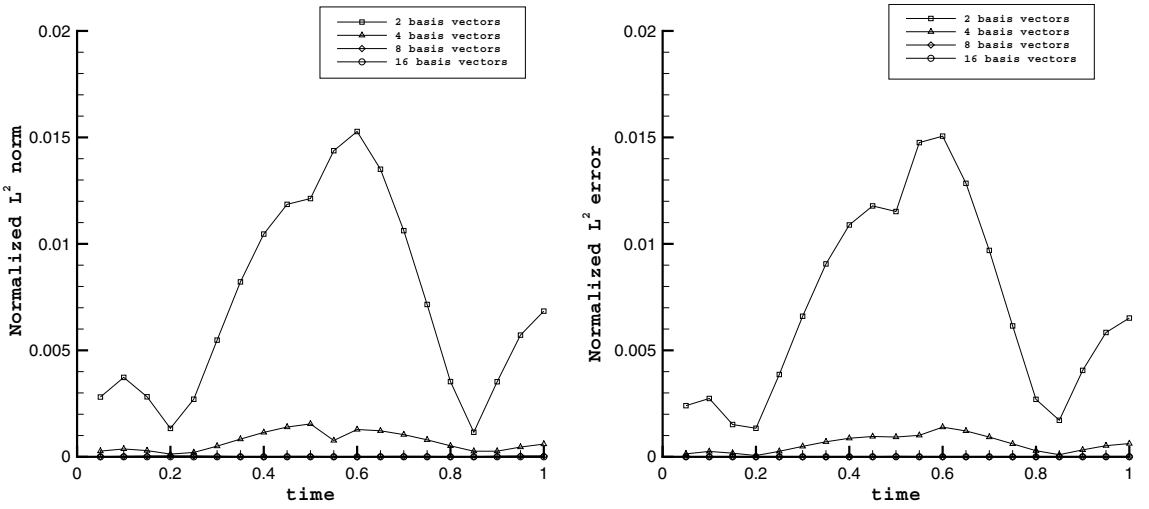

Fig. 3. The error function $\mathscr{E}(t)$ for the two-parameter problem for different numbers of POD basis vectors used in the ROM for $f(u) = u^2$; left: Method I; right: Method II.

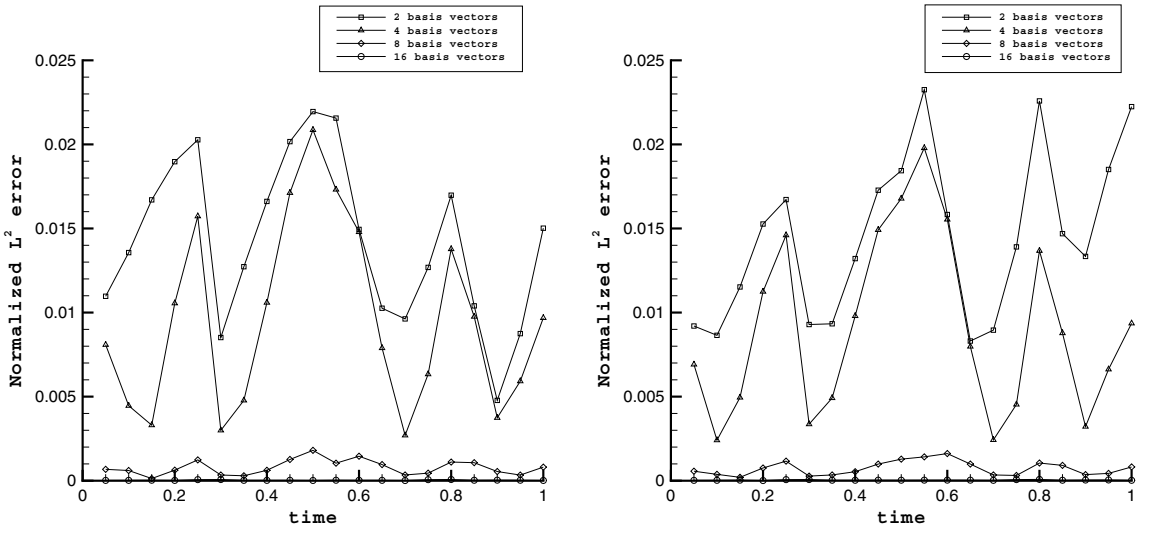

Fig. 4. The error function $\mathscr{E}(t)$ for the four-parameter problem for different numbers of POD basis vectors used in the ROM for $f(u) = u^2$; left: Method I; right: Method II.

$$\sum_{k=1}^{K} \alpha_{k,\ell}\eta_{n,k} = \beta_\ell(t^n) \quad \text{for } \ell = 1,\ldots,K. \tag{19}$$

With the choice $\alpha_{k,\ell} = \delta_{k\ell}$, this linear system is immediately solvable to yield $\eta_{n,k} = \beta_k(t^n)$ for $k = 1,\ldots,K$. This is an especially useful choice whenever the boundary conditions are time-dependent, i.e., whenever β_ℓ is a function of time. In this case, the linear system (19) must be solved at every time step;[9] with the choice $\vec{\alpha}_k = \vec{e}_k$, these solutions are trivially obtainable.

Thus, it seems that the choice $\vec{\alpha}_k = \vec{e}_k$ is a good one; it simplifies the application of boundary conditions and, judging by the results of Section 4.1.3, it also leads to effective ROMs. However, this choice for the parameter points is not always feasible and, at other times, does not always yield good ROM approximations. An example of the latter is the example of Section 4.3. Thus, we now examine other choices for the K parameter points $\{\vec{\alpha}_k\}_{k=1}^K$ to use in step (i) and (ii) of Method I. In particular, we want to see if the accuracy of the ROM is sensitive to the choice of $\vec{\alpha}_k$'s. Note that we use the particular solutions u_k^p in two ways: they are used in applying boundary conditions to the reduced-order approximation and also to modify the snapshots so that they satisfy homogeneous boundary conditions.

For Example I, we assume that the functions $\beta_k(t)g_k(\vec{x})$ can range over the interval $[0,1]$ as is true for the specific case of (16) and (18). In fact, that is why particular solutions and snapshots were generated using parameter

[9] If β_ℓ being independent of time, then so are the $\eta_{n,k}$'s and they may be solved for before the time stepping process is started.

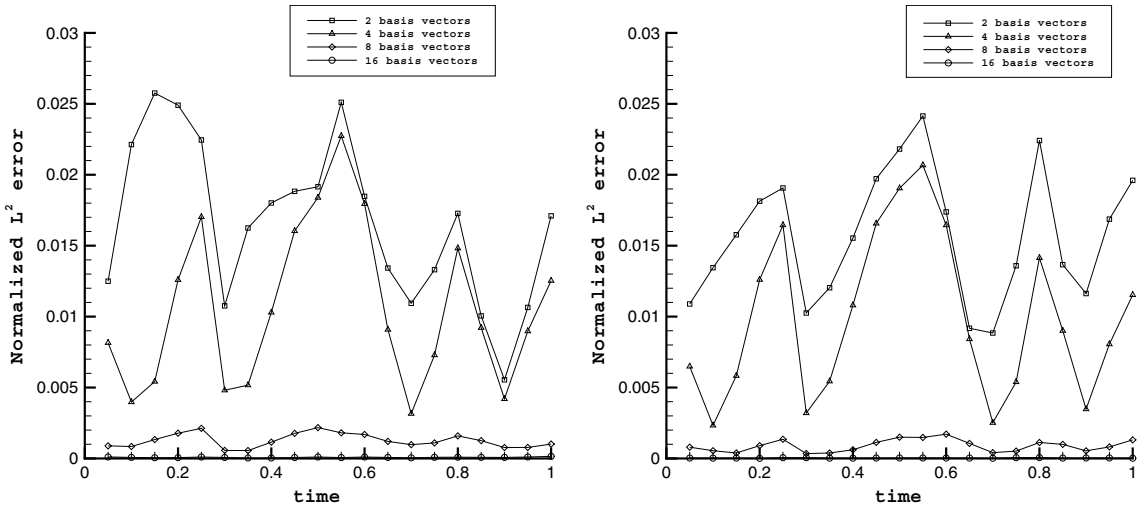

Fig. 5. The error function $\mathscr{E}(t)$ for the four-parameter problem for different numbers of POD basis vectors used in the ROM for $f(u) = e^u$; left: Method I; right: Method II.

Table 1
Choices for $\vec{\alpha}_k$ used to determine particular solutions u_k^p for the four-parameter problem

k	Coordinate				Monte Carlo			
	$(\vec{\alpha}_k)_1$	$(\vec{\alpha}_k)_2$	$(\vec{\alpha}_k)_3$	$(\vec{\alpha}_k)_4$	$(\vec{\alpha}_k)_1$	$(\vec{\alpha}_k)_2$	$(\vec{\alpha}_k)_3$	$(\vec{\alpha}_k)_4$
1	1.0	0.0	0.0	0.0	0.2184	0.9563	0.8295	0.5617
2	0.0	1.0	0.0	0.0	0.4153	0.0661	0.2576	0.1010
3	0.0	0.0	1.0	0.0	0.0438	0.6340	0.06172	0.4495
4	0.0	0.0	0.0	1.0	0.4013	0.7547	0.7973	0.0018
	CVT				LCVT			
	$(\vec{\alpha}_k)_1$	$(\vec{\alpha}_k)_2$	$(\vec{\alpha}_k)_3$	$(\vec{\alpha}_k)_4$	$(\vec{\alpha}_k)_1$	$(\vec{\alpha}_k)_2$	$(\vec{\alpha}_k)_3$	$(\vec{\alpha}_k)_4$
1	0.4166	0.5002	0.7304	0.7661	0.125	0.375	0.625	0.625
2	0.7658	0.4996	0.2694	0.5825	0.875	0.875	0.375	0.875
3	0.2342	0.5007	0.2692	0.4167	0.375	0.125	0.125	0.125
4	0.5831	0.4997	0.7311	0.2349	0.625	0.625	0.875	0.375
	Halton				Hammersley			
	$(\vec{\alpha}_k)_1$	$(\vec{\alpha}_k)_2$	$(\vec{\alpha}_k)_3$	$(\vec{\alpha}_k)_4$	$(\vec{\alpha}_k)_1$	$(\vec{\alpha}_k)_2$	$(\vec{\alpha}_k)_3$	$(\vec{\alpha}_k)_4$
1	0.5	0.3333	0.2	0.1429	0.0	0.5	0.3333	0.2
2	0.25	0.6667	0.4	0.2857	0.25	0.25	0.6667	0.4
3	0.75	0.1111	0.6	0.4286	0.5	0.75	0.1111	0.6
4	0.125	0.4444	0.8	0.5714	0.75	0.125	0.4444	0.8

points in the unit hypercube.[10] Thus, the task of choosing K parameter points $\{\vec{\alpha}_k\}_{k=1}^K$ reduces to sampling K points in the unit hypercube in $\mathbb{R}^K$. For the four-parameter problem, we have several well-known point sampling methods for this task; we use Monte Carlo, Halton [20], Hammersley [21], centroidal Voronoi tessellation (CVT) [9,45], and Latin hypercube centroidal Voronoi (LCVT) [45] techniques. The specific sets of points determined by each sampling method are given in Table 1; "coordinates" refers to the choice $\vec{\alpha}_k = \vec{e}_k$.

Note that we usually choose the particular solutions to be steady state solutions of the finite element system for the corresponding choice of parameter points. This is not the only choice one can make; in fact, one can choose the particular solution u_k^p to be the solution of the finite element system at any time step. To see the possible effect of such a choice, we also choose, for Example I, u_k^p to be the solution corresponding to $\vec{\alpha}_k = \vec{e}_k$ evaluated at the fifth time step which is well before a steady state is achieved.

Another choice for $u^{p,n}(\vec{x})$ that does not require any parameter sampling is to use the finite element function that vanishes at all interior nodes; at boundary nodes, $u^{p,n}(\vec{x})$ satisfies the given boundary condition.

Thus, we have eight sets of parameter points $\{\vec{\alpha}_k\}_{k=1}^K$. We do not provide results for the Halton points because at some time level, the Newton iterations failed to converge using the same time step as that used for the other sampling methods. This already points out that the choice of

[10] If the boundary data could possibly range over, say, the interval [0,3], we would have, e.g., chosen $\vec{\alpha}_k = 3\vec{e}_k$ instead of $\vec{\alpha}_k = \vec{e}_k$ when generating particular solutions.

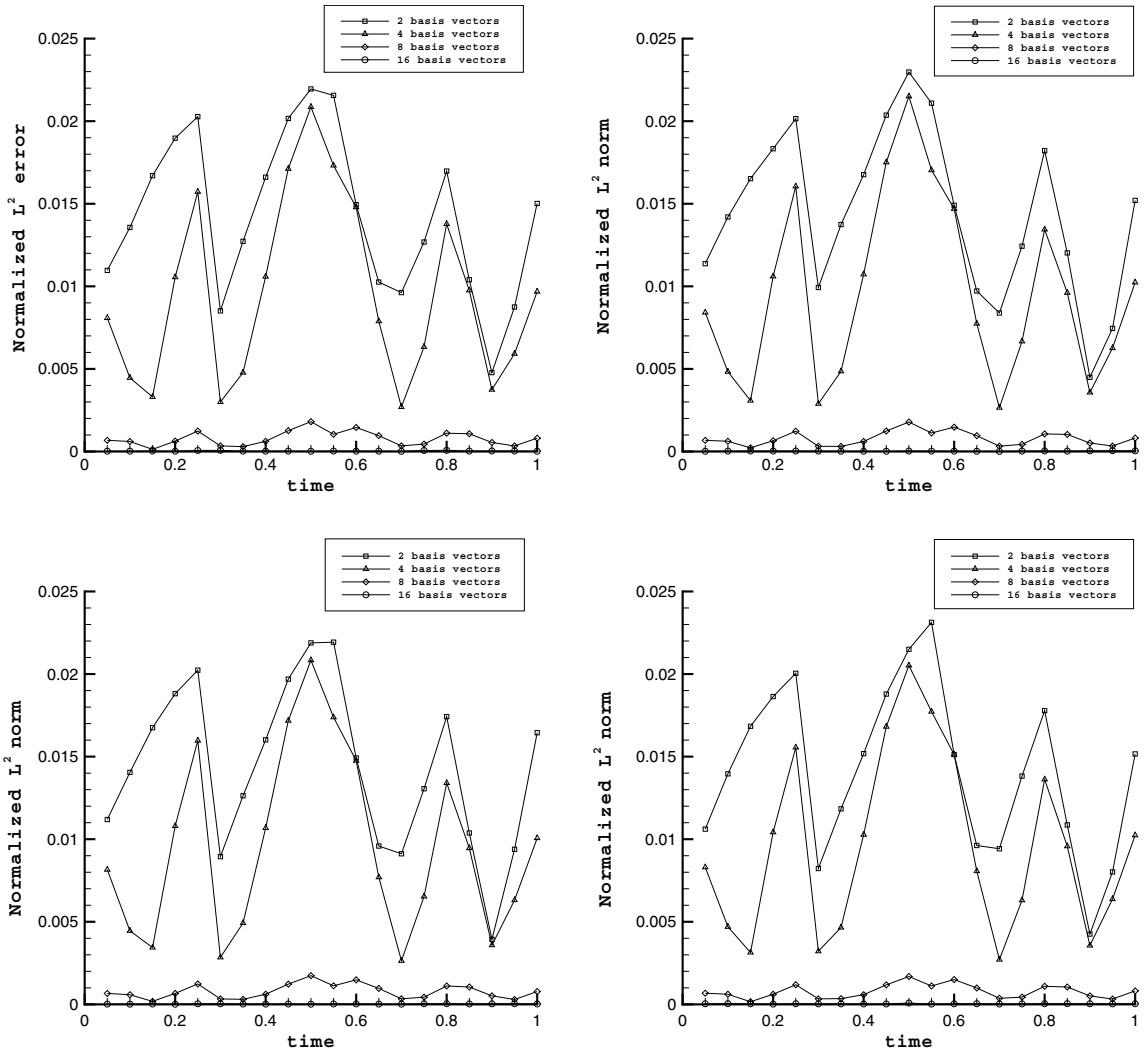

Fig. 6. Comparison of errors in ROM solutions (when compared to full finite element solutions) for the four-parameter problem for $f(u) = u^2$ using Method I with different choices of u^p. Reading from left to right and top to bottom, u^p was determined using $\{\vec{\alpha}_k\}_{k=1}^K$ to be the coordinate points, Monte Carlo points, CVT points, and LCVT points.

sampling method can affect a ROM calculation. The errors for the other seven choices are given in Figs. 6 and 7. Fig. 6 is for the coordinate, Monte Carlo, CVT, and LCVT point sets. We note that for all dimensions of the reduced-order space, the results using these four methods are very similar. Fig. 7 is for the Hammersley method and the last two methods that use particular solutions that are not steady states and particular solutions that vanish at interior nodes. For these three choices, the accuracy of the ROM solutions for a small number of POD vectors is noticeably worse than for the first four choices; for a larger number of POD vectors, all seven approaches seem to perform about as well.

If we ignore the Halton choice for the parameter points, we conclude from Figs. 6 and 7 that the worst errors resulted from the choice of u^p that is zero everywhere except on the boundary. Four choices (coordinate, Monte Carlo, CVT, and LCVT) yielded nearly identical results. However, it is dangerous to draw too many inferences from this single example problem. In fact, it is important to realize that one has no a priori knowledge of which choice of u^p is appropriate for a specific problem. This appears to be a significant disadvantage of Method I compared to Method II which does not require the determination of particular solutions.

4.2.2. QR factorizations in Method II

For Method II, a *QR* factorization must be performed to determine the linear combination of basis vectors that are used as test functions. Even if the boundary conditions are time dependent, only a single *QR* factorization is needed since it is used to determine the linear combination of basis functions that vanish on the boundary. Consequently, the *QR* factorization can be determined before the time stepping procedure is started. For the results reported here, the LAPACK routines DGEQRT and DORGQR were used. Since we perform a single factorization on a matrix of size $m \times K$ (see Section (3.2.2)) and m and K

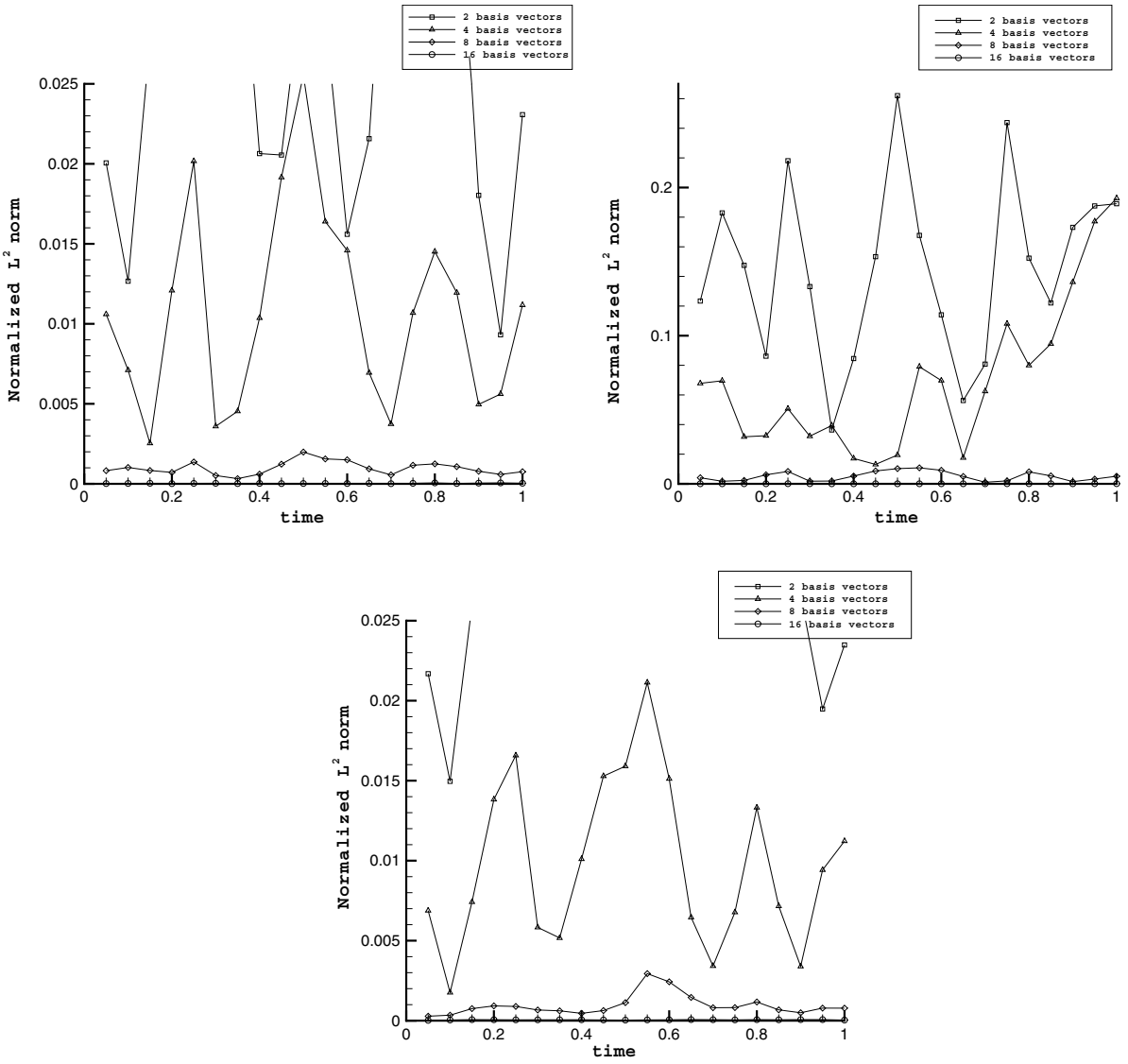

Fig. 7. Comparison of errors in ROM solutions (when compared to full finite element solutions) for the four-parameter problem for $f(u) = u^2$ using Method I with different choices of u^p. For the top-left plot, u^p was determined using $\{\vec{\alpha}_k\}_{k=1}^K$ to be the Hammersley points. For the top-right plot, u^p was chosen to be zero at every interior node and satisfying the desired boundary data. For the bottom plot, u^p was not chosen to be a steady state solution of the finite element system.

are usually very small, this factorization is not costly to effect.

The need to do QR factorizations in Method II should be contrasted to the need to determine particular solutions in Method I; the latter requires, in general, solutions of linear systems to make sure the ROM solution satisfies the given boundary conditions. If the boundary conditions are time dependent, then a different linear system must be solved at every time step. For Method II, the required QR factorization is only done once at the beginning of the calculations. Moreover, the effectiveness of Method I is sensitive to the choice of parameters used in the determination of the particular solutions. Thus, Method II, which does not involve such parameter choices or multiple factorizations, is likely to be more efficient and robust.

4.2.3. The Newton systems

The implementation of Method I results in the Newton system (13) which is more complicated than the Newton system (15) for Method II so that, in this respect, Method II is preferable. The operators on the left-hand sides of these two systems are, of course, the same. However, while the right-hand side of (15) looks very much like an ordinary finite element Newton system (with of course, POD test and trial functions used instead of finite element ones), the right-hand side of (13) involves additional terms. These result from the fact that, for Method I, the ROM solution is the sum of a particular solution and a linear combination of the basis vectors while for Method II, the ROM solution is merely expressed in terms of the latter. For Method II, the Newton system (15) also contains additional equations

that enforce the boundary conditions; however, these equations are simple.

One should also note that Method II is simpler in another respect; there is no need to modify the snapshots so that they satisfy homogeneous boundary conditions.

4.2.4. Initial conditions

A final issue to be considered in implementing a ROM is how one imposes an initial condition. In the examples presented in Section 4.1, the data in the initial condition in the nonlinear PDE problem was zero. Consequently, the initial data for the ROM solutions is simply chosen to vanish.

If the initial data for the given nonlinear PDE problem is nonzero, then one can obtain an initial condition for the ROM problem by projecting the given data onto the reduced space. Recall that for Method I, the ROM solution can be written as $u_{\rm rom}(t^n,\vec{x}) = u^{p,n} + \sum_{j=1}^{d} a_{n,j}\tilde{\phi}_j(\vec{x})$, where $u^{p,n}$ is a linear combination of particular solutions determined so that $u_{\rm rom}$ satisfies the boundary conditions at time t^n. Consequently, to determine an initial condition for the ROM, we need to determine $a_{0,j}$, $j = 1,\ldots,d$. If u_0^h denotes the initial condition for the full finite element problem and we are using Method I, then we first set $\tilde{u}_0^h = u_0^h - u^{p,0}$ so that $\tilde{u}_0^h$ satisfies homogeneous boundary data. Then, to obtain the initial conditions for $a_{0,j}$, $j = 1,\ldots,d$, for the ROM problem, we project $\tilde{u}_0^h$ onto the space spanned by the modified POD basis functions $\{\tilde{\phi}_i\}_{i=1}^{d}$, i.e., we solve the problem

$$\sum_{j=1}^{d}\left(\tilde{\phi}_j,\tilde{\phi}_i\right)a_{0,j} = (\tilde{u}_0^h,\tilde{\phi}_i) \quad \text{for } i = 1\ldots,d,$$

where $(\cdot,\cdot)$ denotes the standard L^2 inner product.

For Method II, the ROM solution has the form $u_{\rm rom}(t_n,\vec{x}) = \sum_{j=1}^{m} b_{n,j}\phi_i(\vec{x})$. Then, to determine the initial conditions for $b_{0,j}$, $j = 1,\ldots,m$, we need only project u_0^h onto the $m = (d+K)$-dimensional space determined by the basis vectors $\{\phi_j\}_{j=1}^{m}$, i.e., we simply solve the problem

$$\sum_{j=1}^{m}(\phi_j,\phi_i)b_{0,j} = (u_0^h,\phi_i) \quad \text{for } i = 1,\ldots,m.$$

Thus, with respect to implementing inhomogeneous initial conditions, Method II seems to be preferable.

For Method I, an alternative procedure for defining the initial conditions for the reduced-order problem is possible. We could simply choose $u^{p,0} = u_0^h$ which immediately leads to the initial data $a_{0,j} = 0$, $j = 1,\ldots,d$, for the coefficients determining $u_{\rm rom}$. At the other time levels, we choose the particular solution $u^{p,n}$ as before. The effectiveness of this as well as the projection procedure has not been fully tested. In particular, the effect of defining the particular solution at $t = 0$ in a different way than at other times has not been studied.

4.3. Example II

In this section, we present an example which arises from modeling the airflow in a large public building. We choose to present results for this problem using only Method II of Section 3.2.2 in which we add additional equations to satisfy the boundary data. The reason for not presenting results for Method I of Section 3.2.1 (in which linear combinations of particular solutions are used to satisfy the boundary conditions) is that for more complicated problems such as the one considered here, the ROM solution is very sensitive to the choice of particular solutions.

For this example, we solve the time dependent incompressible Navier–Stokes equations given by

$$\begin{cases} \dfrac{\partial\vec{u}}{\partial t} - \nu\Delta\vec{u} + \vec{u}\cdot\nabla\vec{u} + \nabla p = 0 & \text{in } \Omega\times(0,T] \\ \nabla\cdot\vec{u} = 0 & \text{in } (0,T]\times\Omega \\ \vec{u}(0,\vec{x}) = \vec{u}_0 & \text{in } (0,T]\times\Omega \end{cases} \tag{20}$$

for the velocity $\vec{u}$ and pressure p; here the Reynolds number $1/\nu$ is chosen to be 100. The physical domain for this example is illustrated in Fig. 8; along the boundary of the flow domain, there are six sets of inlet/outlet orifices Γ_i, $i = 1,\ldots,6$ and a main outlet orifice. The remainder of the boundary of the flow domain is a solid wall. We enforce a zero stress outflow boundary condition at the main outlet orifice indicated in Fig. 8 and homogeneous zero velocity boundary conditions along the solid wall.

At the six sets of inlet/outlet orifices Γ_i, $i = 1,\ldots,6$, we impose the following boundary conditions:

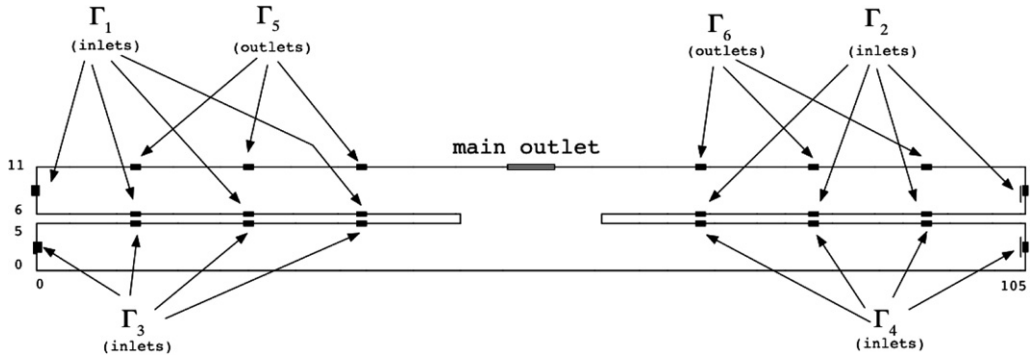

Fig. 8. Physical domain for Example II; a single parameter β_i determines the boundary data at each of the six sets of orifices Γ_i, $i = 1,\ldots,6$.

$$\begin{cases} \Gamma_1(\text{inlets}) & x_1 = 0, \quad 8 \leqslant x_2 \leqslant 9, \quad u = .48\beta_1(x_2 - 8)(9 - x_2), \\ & a_i \leqslant x_1 \leqslant b_i, \quad x_2 = 6, \quad v = .48\beta_1(x_1 - a_i)(b_i - x_1), \\ \Gamma_2(\text{inlets}) & x_1 = 105, \quad 8 \leqslant x_2 \leqslant 9, \quad u = -.5\beta_2(x_2 - 8)(9 - x_2), \\ & c_i \leqslant x_1 \leqslant d_i, \quad x_2 = 6, \quad v = .5\beta_2(x_1 - c_i)(d_i - x_1), \\ \Gamma_3(\text{inlets}) & x_1 = 0, \quad 2 \leqslant x_2 \leqslant 3, \quad u = .44\beta_3(x_2 - 2)(3 - x_2), \\ & a_i \leqslant x_1 \leqslant b_i, \quad x_2 = 5, \quad v = -.44\beta_3(x_1 - a_i)(b_i - x_1), \\ \Gamma_4(\text{inlets}) & x_1 = 105, \quad 2 \leqslant x_2 \leqslant 3, \quad u = -.352\beta_4(x_2 - 2)(3 - x_2), \\ & c_i \leqslant x_1 \leqslant d_i, \quad x_2 = 6, \quad v = -.352\beta_4(x_1 - c_i)(d_i - x_1), \\ \Gamma_5(\text{outlets}) & a_i \leqslant x_1 \leqslant b_i, \quad x_2 = 11, \quad u = .612\beta_5(x_1 - a_i)(b_i - x_1), \\ \Gamma_6(\text{outlets}) & c_i \leqslant x_1 \leqslant d_i, \quad x_2 = 11, \quad u = .896\beta_6(x_1 - c_i)(d_i - x_1), \end{cases} \tag{21}$$

where $(a_i, b_i) \in \{(10,11),(22,23),(34,35)\}$ and $(c_i, d_i) \in \{(70,71),(82,83),(94,95)\}$, $i = 1,2,3$.

Approximate solutions of the Navier–Stokes equations are obtained using a standard Taylor-Hood finite element method to effect spatial discretization, i.e., continuous piecewise linear functions on triangles are used to approximate the pressure and continuous piecewise quadratic functions on the same triangles are used to approximate the components of the velocity. The backward Euler approximation is used to effect temporal discretization. A uniform grid consisting of 8520 triangles and resulting in 35,730 unknowns is used as is a uniform time step.

A total of 509 snapshots were generated in the following manner:

- generate 10 LCVT points, $\vec{\alpha}_k$, $k = 1,\ldots,10$, in $\mathbb{R}^6$;
- in (21), set $\beta_i = 1$, $i = 1,\ldots,6$, set the initial velocity to zero, and generate snapshots that are finite element approximations to the Navier–Stokes equations evaluated at different time levels;
- for $k = 1,\ldots,10$, set $\beta_i = (\vec{\alpha}_k)_i$, $i = 1,\ldots,6$, set the initial velocity to the steady state solution obtained when $\beta_i = 1$, $i = 1,\ldots,6$, and generate snapshots which are finite element approximations to Navier–Stokes equations evaluated at different time levels.

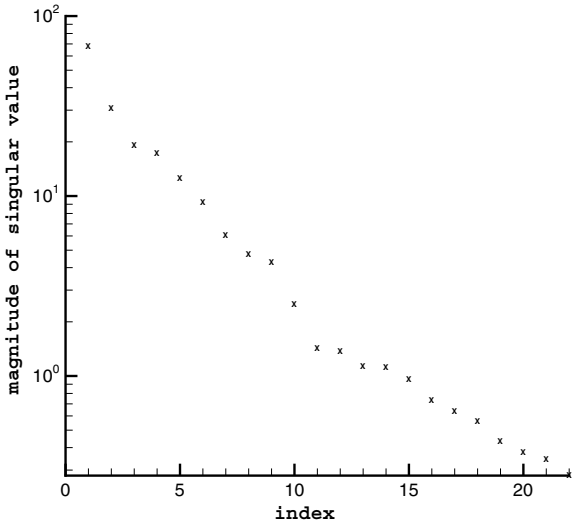

Fig. 9. The largest singular values of the snapshot matrix for Example II.

A snapshot was generated at every time step for the first twenty steps and then every 10 time steps until steady state was achieved.

From the snapshots, a total of 22 basis POD vectors were generated; the corresponding singular values of the snapshot matrix are plotted in Fig. 9. Again we see that the singular values decay rapidly indicating that just a few POD vectors can capture most of the information contained in the snapshot set.

Using Method II, POD bases are used to determine a reduced-order solution of the Navier–Stokes system for randomly chosen values of the coefficients β_i, $i = 1,\ldots,6$, appearing in (21); specifically, the values

$$\beta_1 = 0.20, \quad \beta_2 = 0.72, \quad \beta_3 = 0.48, \\ \beta_4 = 0.83, \quad \beta_5 = 0.64, \quad \beta_6 = 0.33 \tag{22}$$

are used. Note that these values of β_i are totally different from those used to generate the snapshot set. With $K = 6$

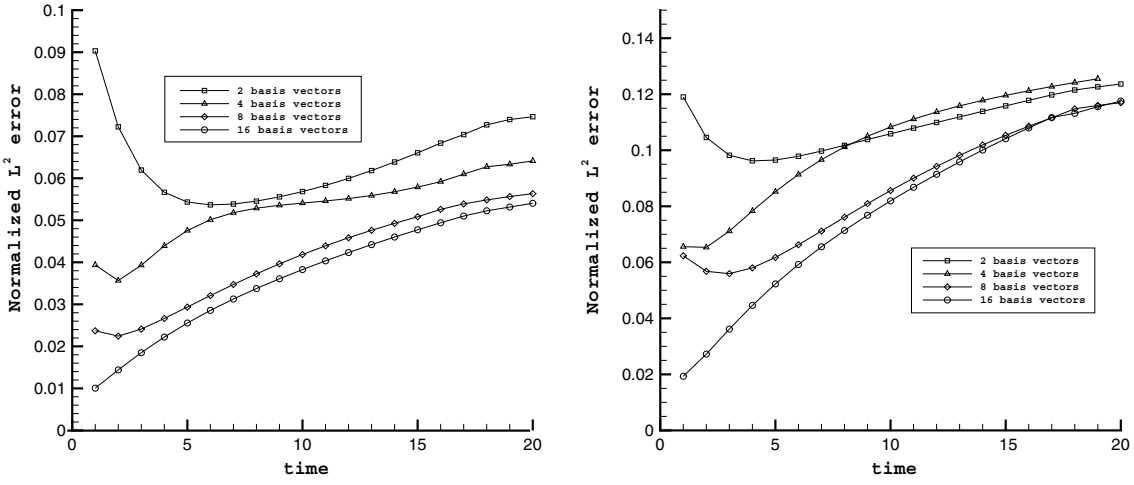

Fig. 10. Error in the ROM solution (compared to the full finite element solution) for Example II. The left and right plots respectively depict the errors for the horizontal and vertical components of the velocity.

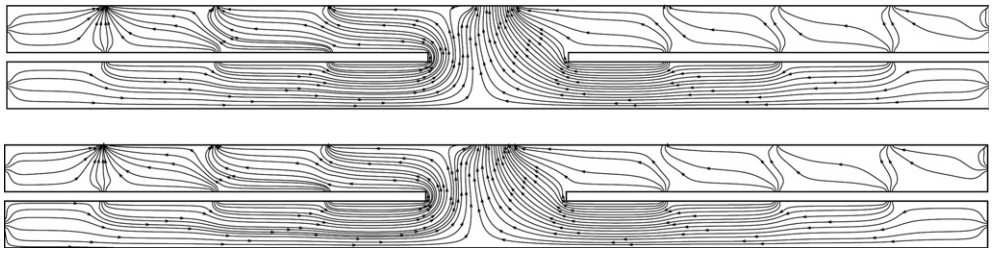

Fig. 11. The streamlines of the full finite element (top) and ROM (bottom) solutions for Example II.

parameters, the test functions used in the discretized weak form of the Navier–Stokes equations are chosen to be 2, 4, 8, and 16 appropriate linear combinations of the first 8, 10, 14, and 22 POD basis vectors, respectively. The linear combinations are chosen as described in Section 3.2.2. The results of the POD-based ROM are compared to the full finite element approximation of (20) based on the same parameter values (22). The errors for the horizontal and vertical components of the velocity are graphed in Fig. 10. In addition, Fig. 11 gives a comparison of the streamlines for the flow at a fixed time instant generated from the full finite element approximation with 35,730 unknowns and the ROM approximation using 14 unknowns. We again see that a reduced-order approximation of very small dimension produces a reasonably accurate approximation to the full finite element solution. To decrease the error further, a more extensive set of snapshots would need to be used.

5. Conclusions

In this paper we have presented a new technique for handling multiple parametric constraints in the form of a set of Dirichlet boundary conditions for a PDE system. This technique is based on developing a specific linear combinations of the global POD basis vectors that can be used to enforce, in the reduced-order model, the K boundary conditions for the PDEs. Through the use of a reduced-space QR factorization, the method then projects the PDE system onto an subspace of dimension $m - K$ (where m is the dimension of the reduced basis) that is orthogonal to the subspace used to enforce the boundary conditions. The resulting ROM strictly enforces the boundary conditions and produces a square system of equations that can be used to determine ROM approximations. In our computational examples, we have demonstrated the method on a one, two and four parameter scalar PDE problem and a six parameter Navier–Stokes problem. These initial results appear very promising and further investigation of the method appears warranted.

References

[1] N. Aubry, W. Lian, E. Titi, Preserving symmetries in the proper orthogonal decomposition, SIAM J. Sci. Comput. 14 (1993) 483–505.
[2] M. Barrault, Y. Maday, N. Nguyen, A. Patera, An 'empirical interpolation' method: application to efficient reduced-basis discretization of partial differential equations, C. R. Acad. Sci. Paris, Ser., I 339 (2004) 667–672.
[3] G. Berkooz, P. Holmes, J. Lumley, The proper orthogonal decomposition in the analysis of turbulent flows, Ann. Rev. Fluid Mech. 25 (1993) 539–575.
[4] G. Berkooz, E. Titi, Galerkin projections and the proper orthogonal decomposition for equivariant equations, Phys. Lett. A 174 (1993) 94–102.
[5] J. Burkardt, M. Gunzburger, H. Lee, Centroidal Voronoi tessellation-based reduced-order modeling of complex systems, SIAM J. Sci. Comput., in press.
[6] J. Burkardt, M. Gunzburger, H. Lee, POD and CVT-based reduced-order modeling of Navier–Stokes flows, Comput. Methods Appl. Mech. Engrg., in press, doi:10.1016/j.cma.2006.04.004.
[7] E. Christensen, M. Brons, J. Sorensen, Evaluation of proper orthogonal decomposition-based decomposition techniques applied to parameter-dependent nonturbulent flows, SIAM J. Sci. Comput. 21 (2000) 1419–1434.
[8] A. Deane, I. Kevrekidis, G. Karniadakis, S. Orszag, Low-dimensional models for complex geometry flows: applications to grooved channels and circular cylinders, Phys. Fluids A 3 (1991) 2337–2354.
[9] Q. Du, V. Faber, M. Gunzburger, Centroidal Voronoi tessellations: applications and algorithms, SIAM Rev. 41 (1999) 637–676.
[10] Q. Du, M. Gunzburger, Model reduction by proper orthogonal decomposition coupled with centroidal Voronoi tessellation, in: Proc. Fluids Engineering Division Summer Meeting, FEDSM2002-31051, ASME, 2002.
[11] Q. Du, M. Gunzburger, Grid generation and optimization based on centroidal Voronoi tessellations, Appl. Math. Comput. 133 (2002) 591–607.
[12] Q. Du, M. Gunzburger, Centroidal Voronoi tessellation based proper orthogonal decomposition analysis, in: Proc. 8th Conference on Control of Distributed Parameter Systems, Birkhauser, Basel, 2002, pp. 137–150.
[13] Q. Du, M. Gunzburger, L. Ju, Meshfree, probabilistic determination of points sets and support regions for meshless computing, Comput. Methods Appl. Mech. Engrg. 191 (2002) 1349–1366.
[14] Q. Du, M. Gunzburger, L. Ju, Constrained CVTs on general surfaces, SIAM J. Sci. Comput. 24 (2003) 1488–1506.
[15] Q. Du, M. Gunzburger, L. Ju, Voronoi-based finite volume methods, optimal Voronoi meshes, and PDEs on the sphere, Comput. Methods Appl. Mech. Engrg. 192 (2003) 3933–3957.
[16] A. Faulds, Centroidal Voronoi decompositions, algorithms and applications, M.S. thesis, Department of Mathematics, Penn State University, 2002.
[17] A. Faulds, B. King, Sensor location in feedback control of partial differential equation systems, in: Proc. 2000 IEEE CCA/CACSD, IEEE, Washington, 2000, pp. 536–541.
[18] M. Graham, I. Kevrekidis, Pattern analysis and model reduction: some alternative approaches to the Karhunen–Lóeve decomposition, Comput. Chem. Engrg. 20 (1996) 495–506.
[19] M. Grepl, A. Patera, A posteriori error bounds for reduced-basis approximations of parameterized parabolic partial differential equations, ESAIM: M2AN 39 (2005) 157–181.

[20] J. Halton, On the efficiency of certain quasi-random sequences of points in evaluating multi-dimensional integrals, Numer. Math. 2 (1960) 84–90.
[21] J. Hammersley, Monte Carlo methods for solving multivariable problems, Proc. New York Acad. Sci. 86 (1960) 844–874.
[22] P. Holmes, J. Lumley, G. Berkooz, Turbulence, Coherent Structures, Dynamical Systems and Symmetry, Cambridge University, Cambridge, 1996.
[23] P. Holmes, J. Lumley, G. Berkooz, J. Mattingly, R. Wittenberg, Low-dimensional models of coherent structures in turbulence, Phys. Rep. 287 (1997) 337–384.
[24] L. Ju, Q. Du, M. Gunzburger, Probabilistic methods for centroidal Voronoi tessellations and their parallel implementations, J. Parallel Comput. 28 (2002) 1477–1500.
[25] K. Kunisch, S. Volkwein, Control of Burger's equation by a reduced order approach using proper orthogonal decomposition, JOTA 102 (1999) 345–371.
[26] K. Kunisch, S. Volkwein, Galerkin proper orthogonal decomposition methods for parabolic problems, Spezialforschungsbereich F003 Optimierung und Kontrolle, Projektbereich Kontinuierliche Optimierung und Kontrolle, Bericht Nr. 153, Graz, 1999.
[27] J. Lumley, Stochastic Tools in Turbulence, Academic, New York, 1971.
[28] D. Nagy, Modal representation of geometrically nonlinear behavior by the finite element method, Comput. Struct. 10 (1979) 693.
[29] A. Noor, Recent advances in reduction methods for nonlinear problems, Comput. Struct. 13 (1981) 31–44.
[30] A. Noor, C. Anderson, J. Peters, Reduced basis technique for collapse analysis of shells, AAIA J. 19 (1981) 393.
[31] A. Noor, J. Peters, Tracking post-linit-paths with reduced basis technique, Comput. Methods Appl. Mech. Engrg. 28 (1981) 217.
[32] A. Okabe, B. Boots, K. Sugihara, S. Chiu, Spatial Tessellations: Concepts and Applications of Voronoi Diagrams, Wiley, Chichester, 2000.
[33] H. Park, D. Cho, Low dimensional modeling of flow reactors, Int. J. Heat Mass Trans. 39 (1996) 3311–3323.
[34] H. Park, J. Chung, A sequential method of solving inverse natural convection problems, Inverse Prob. 18 (2002) 529–546.
[35] H. Park, Y. Jang, Control of Burgers equation by means of mode reduction, Int. J. Engrg. Sci. 38 (2000) 785–805.
[36] H. Park, J. Lee, Solution of an inverse heat transfer problem by means of empirical reduction of modes, Z. Angew. Math. Phys. 51 (2000) 17–38.
[37] H. Park, W. Lee, An efficient method of solving the Navier–Stokes equations for flow control, Int. J. Numer. Methods Engrg. 41 (1998) 1133–1151.
[38] H. Park, W. Lee, A new numerical method for the boundary optimal control problems of the heat conduction equation, Int. J. Numer. Methods Engrg. 53 (2002) 1593–1613.
[39] J. Peterson, The reduced basis method for incompressible viscous flow calculations, SIAM J. Sci. Stat. Comput. 10 (1989) 777.
[40] M. Rathinam, L. Petzold, A new look at proper orthogonal decomposition, in press.
[41] M. Rathinam, L. Petzold, Dynamic iteration using reduced order models: a method for simulation of large scale modular systems, in press.
[42] S. Ravindran, Proper orthogonal decomposition in optimal control of fluids, Int. J. Numer. Methods Fluids 34 (2000) 425–448.
[43] S. Ravindran, Reduced-order adaptive controllers for fluid flows using POD, SIAM J. Sci. Comput. 15 (2000) 457–478.
[44] J. Rodríguez, L. Sirovich, Low-dimensional dynamics for the complex Ginzburg–Landau equations, Physica D 43 (1990) 77–86.
[45] Y. Saka, M. Gunzburger, J. Burkardt, Latinized, improved LHS, and CVT point sets in hypercubes, in press.
[46] L. Sirovich, Turbulence the dynamics of coherent, structures, I–III, Quart. J. Appl. Math. 45 (1987) 561–590.
[47] N. Smaoui, D. Armbruster, Symmetry and the Karhunen–Loève analysis, SIAM J. Sci. Comput. 18 (1997) 1526–1532.
[48] K. Veroy, A. Patera, Certified real-time solution of the parameterized steady incompressible Navier–Stokes equations: rigorous reduced-basis a posteriori error bounds, Int. J. Numer. Methods Fluids 47 (2005) 773–788.
[49] S. Volkwein, Optimal control of a phase field model using the proper orthogonal decomposition, ZAMM 81 (2001) 83–97.
[50] S. Volkwein, Proper orthogonal decomposition and singular value decomposition, Spezialforschungsbereich F003 Optimierung und Kontrolle, Projektbereich Kontinuierliche Optimierung und Kontrolle, Bericht Nr. 153, Graz, 1999.