Comput. Methods Appl. Mech. Engrg. 272 (2014) 251–270

Contents lists available at ScienceDirect

Comput. Methods Appl. Mech. Engrg.

journal homepage: www.elsevier.com/locate/cma

Stabilization of projection-based reduced order models for linear time-invariant systems via optimization-based eigenvalue reassignment

Irina Kalashnikova [a,*], Bart van Bloemen Waanders [b], Srinivasan Arunajatesan [c], Matthew Barone [c]

[a] Computational Mathematics Department, Sandia National Laboratories, P.O. Box 5800, MS 1320, Albuquerque, NM 87185, USA
[b] Optimization and Uncertainty Quantification Department, Sandia National Laboratories, P.O. Box 5800, MS 1318, Albuquerque, NM 87185, USA
[c] Aerosciences Department, Sandia National Laboratories, P.O. Box 5800, MS 1124, Albuquerque, NM 87185, USA

ARTICLE INFO

Article history:
Received 10 September 2013
Received in revised form 7 January 2014
Accepted 8 January 2014
Available online 18 January 2014

Keywords:
Reduced order model (ROM)
Proper orthogonal decomposition (POD)/ Galerkin projection
Linear time-invariant (LTI) system
Stability
Pole placement
Constrained nonlinear least-squares

ABSTRACT

A new approach for stabilizing unstable reduced order models (ROMs) for linear time-invariant (LTI) systems through an *a posteriori* post-processing step applied to the algebraic ROM system is developed. The key idea is to modify the unstable eigenvalues of the ROM system by moving these eigenvalues into the stable half of the complex plane. It is demonstrated that this modification to the ROM system eigenvalues can be accomplished using full state feedback (a.k.a. pole placement) algorithms from control theory. This approach ensures that the modified ROM is stable provided the system's unstable poles are controllable and observable; however, the accuracy of the stabilized ROM is not guaranteed. To remedy this difficulty and guarantee an accurate stabilized ROM, a constrained nonlinear least-squares optimization problem for the stabilized ROM eigenvalues in which the error in the ROM output is minimized is formulated. This optimization problem is small and therefore computationally inexpensive to solve. Performance of the proposed algorithms is evaluated on two test cases for which ROMs constructed via the proper orthogonal decomposition (POD)/Galerkin method suffer from instabilities.

Published by Elsevier B.V.

1. Introduction

As computing power has increased, so has the complexity of multi-physics models. Simultaneously, there has been a continuing push to incorporate uncertainty quantification (UQ) into high-fidelity simulations. Unfortunately, integrating UQ techniques into high-fidelity simulation codes can present an intractable computational burden due to the high-dimensional systems that arise, as well as the need to run these simulations many times in order to explore a space of design parameters or uncertain inputs.

Reduced order modeling is a promising tool that can enable not only UQ, but also on-the-spot decision-making, optimization and/or control. A reduced order model (ROM) is a surrogate model constructed from a full order (high-fidelity) model (FOM) that retains the essential physics and dynamics of the FOM, but has a much lower computational cost. Numerous approaches to construct ROMs exist, from simply running a numerical simulation on a coarser mesh, to surrogates obtained from data-fitting (e.g., Kriging interpolation). More commonly, however, the term "reduced order model" refers to a

* Corresponding author. Tel.: +1 5058441097.
E-mail address: ikalash@sandia.gov (I. Kalashnikova).

http://dx.doi.org/10.1016/j.cma.2014.01.011
0045-7825/Published by Elsevier B.V.

projection-based reduced order model, the subject of the present work. The basic idea of projection-based reduced order modeling is to project the state of a large dimensional space onto a small dimensional subspace that contains the essential dynamics of the system. Examples of projection-based model reduction approaches include proper orthogonal decomposition (POD) [13,14,9], balanced proper orthogonal decomposition (BPOD) [19,11], balanced truncation [16,5], the reduced basis method [15,32], and Krylov-based techniques [31].

In order for a ROM to serve as a viable mathematical model of a physical system of interest, it is important that it preserves certain crucial properties of the original system. Particularly important is that the ROM maintains numerical stability of its underlying physical system, as stability is a prerequisite for the ROM's accuracy and convergence. Some projection-based model reduction techniques give rise to ROMs with an *a priori* stability guarantee. One example of such a method is balanced truncation [16,5]. Unfortunately, the computational cost of this method, which requires the computation and simultaneous diagonalization of infinite controllability and observability Gramians, makes balanced truncation computationally intractable for systems of very large dimensions (i.e., systems with more than 10,000 degrees of freedom (dofs) [12]). Among the most popular model reduction techniques that are computationally tractable for very large systems are the POD method [13,14,9] and the BPOD method [19,11]. In general, these methods lack an *a priori* stability guarantee. In [18], Amsallem et al. suggest that POD and BPOD ROMs constructed for linear time-invariant (LTI) systems in descriptor form tend to possess better numerical stability properties than POD ROMs constructed for LTI systems in non-descriptor form. Although heuristics such as these exist, it is in general unknown *a priori* if a ROM constructed using POD or BPOD will preserve the stability properties of the high-fidelity system from which the model was constructed. Hence, a ROM might be stable for a given number of modes, but unstable for other choices of basis size; see [10,3,4] for examples of this for POD models of compressible flow.

A literature search reveals that approaches for developing stability-preserving projection-based ROMs based on POD and BPOD fall into roughly three categories, overviewed briefly below.

The first category of methods derives (*a priori*) a stability-preserving model reduction framework, often specific to a particular equation set. In [12], Rowley et al. show that Galerkin projection preserves the stability of an equilibrium point at the origin if the ROM is constructed in an "energy-based" inner product. In [3,4], Barone et al. demonstrate that a symmetry transformation leads to a stable formulation for a Galerkin ROM for the linearized compressible Euler equations [3,4] and non-linear compressible Navier–Stokes equations [17] with solid wall and far-field boundary conditions. In [1], Serre et al. propose applying the stabilizing projection developed by Barone et al. in [3,4] to a skew-symmetric system constructed by augmenting a given linear system with its adjoint system. This approach yields a ROM that is stable at finite time even if the solution energy of the full-order model is growing. In [35,40], Sirisup et al. develop a method for correcting long-term unstable behavior for POD models using a spectral viscosity (SV) diffusion convolution operator. The advantage of approaches such as these is they are physics-based, and guarantee *a priori* a stable ROM; the downside is that they can be difficult to implement, as access to the high-fidelity code and/or the governing partial differential equations (PDEs) is often required.

A second category of methods is aimed to remedy the so-called "mode truncation instability". These methods [36–38,23,41], motivated by the observation that higher order modes can give rise to nonphysical instabilities in the ROM system, are often physics-based and minimally intrusive to the ROM. In [23], a ROM stabilization methodology that achieves improved accuracy and stability through the use of a new set of basis functions representing the small, energy-dissipation scales of turbulent flows is derived by Balajewicz et al. The stabilization of ROMs using shift modes and residual modes was proposed in [37,38] by Noack et al. and Bergmann et al. respectively. Other authors, e.g., Terragni et al. [41], have demonstrated that the stability and performance of a ROM can be improved by adapting the POD manifold to the local dynamics.

The third category of approaches are those which stabilize an unstable ROM through a post-processing (*a posteriori*) stabilization step applied to an unstable algebraic ROM system. Ideally, the stabilization only minimally alters the ROM physics, so that the ROM's accuracy is not sacrificed. In [2], Amsallem et al. propose a method for stabilizing projection-based linear ROMs through the solution of a small-scale convex optimization problem. In [22], a set of linear constraints for the left-projection matrix, given the right-projection matrix, are derived by Bond et al. to yield a projection framework that is guaranteed to generate a stable ROM. In [20], Zhu et al. derive some large eddy simulation (LES) closure models for POD ROMs for the incompressible Navier–Stokes equations, and demonstrate numerically that the inclusion of these LES terms yields a ROM with increased numerical stability. In [39], Couplet et al. propose methods for correcting the behavior of a low-order POD-Galerkin system through a coefficient calibration/minimization. A nice feature of these and similar approaches is that they are easy to implement: often the stabilization step can be applied in a "black-box" fashion to an algebraic ROM system that has already been constructed. However, the approaches can give rise to inconsistencies between the ROM and FOM physics, thereby limiting the accuracy of the ROM.

The present work proposes and develops a *new* ROM stabilization method for LTI systems that falls into the second category of methods described above. This approach can be used to stabilize ROMs constructed using *any* choice of reduced basis (e.g., POD [8], balanced truncation [16,5], proper generalized decomposition [42], among others). The key idea, motivated by the concept of full state feedback (a.k.a. pole placement) in control theory, is to change the unstable eigenvalues of a system matrix by pushing them into the stable half of the complex plane. The eigenvalues of a ROM system matrix can be modified by applying directly full state feedback (a.k.a. pole placement) algorithms from control theory [6,7,25], that is, by adding to the ROM system a linear feedback control term, and solving for the feedback matrix such that the stabilized ROM system has a desired set of eigenvalues. However, this approach can change the ROM physics, thereby making the ROM

inaccurate. To alleviate this difficulty, an alternative algorithm is developed in which a constrained nonlinear least-squares optimization problem that minimizes the error in the ROM output (thereby maximizing the accuracy of the ROM) is formulated. The said optimization problem is small, with at most as many dofs as the number of dofs in the ROM, and therefore computationally inexpensive to solve.

The remainder of this paper is organized as follows. Galerkin projection-based model reduction for LTI systems is reviewed in Section 2. Section 3 presents the two ROM stabilization algorithms described above. The first employs full state feedback (a.k.a. pole placement) algorithms from control theory to change an unstable ROM's unstable eigenvalues (Section 3.1); the second solves a constrained nonlinear least-squares optimization problem for the ROM eigenvalues in which the ROM output error is minimized, and changes the eigenvalues directly in the ROM system using the eigenvalue decomposition (Section 3.2). The performance of these eigenvalue reassignment algorithms is evaluated on two benchmarks in Section 4: the international space station (ISS) problem (Section 4.1) and a involving a model of an electrostatically actuated beam (Section 4.2). For both test cases, the ROMs are constructed via the POD/Galerkin method and suffer from instabilities. The numerical results reveal the superiority of the second stabilization algorithm over the first, and demonstrate that the second stabilization algorithm delivers a stable and accurate ROM. Conclusions are offered in Section 5.

2. Projection-based model reduction for LTI systems

In this section, projection-based model reduction applied to LTI systems is reviewed briefly. A system is called time-invariant if the output response for a given input does not depend on when that input is applied [6,7]. In constructing a projection-based reduced order model, the basic idea is to project the state space of a large dimension onto a small dimensional subspace that contains the essential dynamics of the system. Consider an LTI FOM:

$$\begin{aligned} \dot{\mathbf{x}}(t) &= \mathbf{A}\mathbf{x}(t) + \mathbf{B}\mathbf{u}(t) \\ \mathbf{y}(t) &= \mathbf{C}\mathbf{x}(t), \end{aligned} \tag{1}$$

where "·" indicates differentiation with respect to time, i.e., $\dot{\mathbf{x}} \equiv \frac{\partial \mathbf{x}}{\partial t}$; $\mathbf{x}(t) \in \mathbb{R}^N$ is the full order state vector; $\mathbf{u}(t) \in \mathbb{R}^P$ is the vector of control variables; $\mathbf{y}(t) \in \mathbb{R}^Q$ is the output. The matrices $\mathbf{A} \in \mathbb{R}^{N\times N}, \mathbf{B} \in \mathbb{R}^{N\times P}$ and $\mathbf{C} \in \mathbb{R}^{Q\times N}$ are constant matrices (in particular, they are not a function of time t). A system of the form (1) would arise, for instance, by discretizing a linear set of PDEs in space using a discretization scheme, e.g., the finite element method.

The general approach to Galerkin projection-based model reduction consists of two steps:

Step 1: Calculation of a reduced basis of order M, with $M \ll N$.
Step 2: Projection of the governing system (1) onto the reduced basis in some inner product.

In the present work, it will be assumed the projection is done at the level of the discrete Eqs. (1) and in the L^2 inner product, defined by

$$(\mathbf{u}, \mathbf{v}) \equiv \mathbf{u}^T\mathbf{v}, \tag{2}$$

for $\mathbf{u}, \mathbf{v} \in \mathbb{R}^N$. To simplify the presentation, it will also be assumed that the ROM is constructed using a Galerkin projection, where the solution is approximated by and projected onto the same reduced basis. It is emphasized that the approaches developed in this work are not restricted to ROMs constructed using Galerkin projection; a more general Petrov–Galerkin projection can be employed.

Let $\mathbf{\Phi}_M \in \mathbb{R}^{N\times M}$ denote a reduced basis for (1), respectively. Assume this matrix has full column rank, and is orthonormal in the inner product (2), so that $\mathbf{\Phi}_M^T\mathbf{\Phi}_M = \mathbf{I}_M$, where $\mathbf{I}_M$ denotes the $M \times M$ identity matrix. First, the solution to the FOM system (1) is approximated as:

$$\mathbf{x}_N(t) \approx \mathbf{\Phi}_M\mathbf{x}_M(t), \tag{3}$$

where $\mathbf{x}_M(t)$ denotes the ROM solution (to be determined in solving the ROM). Substituting (3) into (1) and projecting the resulting system onto the reduced basis $\mathbf{\Phi}_M$, the following is obtained:

$$\begin{aligned} \dot{\mathbf{x}}_M(t) &= \mathbf{A}_M\mathbf{x}_M(t) + \mathbf{B}_M\mathbf{u}(t) \\ \mathbf{y}_M(t) &= \mathbf{C}_M\mathbf{x}_M(t), \end{aligned} \tag{4}$$

where $\mathbf{y}_M(t)$ is a reduced approximation of the output, and

$$\mathbf{A}_M = \mathbf{\Phi}_M^T\mathbf{A}\mathbf{\Phi}_M \in \mathbb{R}^{M\times M}, \quad \mathbf{B}_M = \mathbf{\Phi}_M^T\mathbf{B} \in \mathbb{R}^{M\times P}, \quad \mathbf{C}_M = \mathbf{C}\mathbf{\Phi}_M \in \mathbb{R}^{Q\times M}. \tag{5}$$

The dynamical system (4) is the ROM LTI system. It is small ($M \times M$ with $M \ll N$), and describes accurately the dynamics of the full order system (1) for some set of conditions. The ROM solution $\mathbf{x}_M(t)$ is obtained by advancing (4) forward in time using a time-integration scheme. Since the FOM considered here is linear, the projection terms in (5) are not time-dependent. Hence, these terms can be pre-computed and stored in the offline stage of the model reduction – in particular,

they need not be re-computed at each time step of the online time-integration stage of the ROM. The reduced basis $\mathbf{\Phi}_M$ can be calculated using a number of approaches, e.g., POD [13,14,9], BPOD [19,11], balanced truncation [16,5], goal-oriented methods [10], or the reduced basis method [15,32].

3. ROM stabilization via eigenvalue reassignment

One problem that can arise in projection-based model reduction and addressed herein is ROM instability. In the present work, the term "stability" refers to Lyapunov stability, defined below.

Definition 3.1. (Lyapunov-Stability [33]): An LTI system (1) is stable in the sense of Lyapunov if and only if all the eigenvalues of $\mathbf{A}$ have real parts less than or equal to zero, and those with real parts equal to zero are non-repeated.

For popular model reduction techniques such as POD and BPOD, a ROM is not guaranteed to preserve the stability properties of the FOM from which it was constructed. This is because orthogonal and bi-orthogonal projections do not in general preserve stability. Hence, for some number of modes M, the ROM system matrix $\mathbf{A}_M$ may be unstable even though the FOM system matrix $\mathbf{A}$ is stable. This issue is particularly problematic for strongly stiff systems, and can arise in computational fluid dynamics applications (e.g., high Reynolds number 3D turbulent flow problems, compressible flow problems [10,3,4]), as well as computational structural dynamics applications (e.g., the second order Lagrangian systems considered in this paper).

In the following subsections, two algorithms are proposed for stabilizing (4) by modifying the unstable eigenvalues of $\mathbf{A}_M$ through a "black-box" post-processing step applied to the given (unstable) ROM system, meaning they can be used to stabilize ROMs from *any* application area. It will be assumed from this point onward that the matrix $\mathbf{A}$ defining the FOM system (1) is stable. Algorithm 2 is the primary contribution of this paper. Algorithm 1 is provided, as it served as a strategic foundation for the final development (Algorithm 2). It is given here not only for the sake of completeness, but also because it is shown in Section 3.3 that Algorithm 2 can be seen as a variant of Algorithm 1.

3.1. Algorithm 1: ROM stabilization via full state feedback (a.k.a. pole placement)

The first ROM stabilization algorithm is motivated by the observation that (4) is an LTI system, and, as such, can be stabilized using full state feedback, or pole placement, methods from control theory [6,7,25]. The general approach of stabilizing an LTI system using full state feedback is reviewed below.

Consider the open loop ROM LTI system (4), where it is assumed $\mathbf{u}(t)$ is given, so that $\mathbf{B}_M\mathbf{u}(t)$ represents, for instance, a given source for the equations. The objective of full state feedback (pole placement) is to redesign the dynamics of the system (4) through feedback of the state. If $\mathbf{A}_M$ is unstable, it is desired to redesign the system such that it is stable. Towards this end, the open-loop system (4) is transformed into a closed-loop system, and a feedback controller that positions the closed loop eigenvalues of the system is developed. The first step is to select a control matrix $\mathbf{B}_C \in \mathbb{R}^{M\times J}$ for some integer J, and modify the system (4) by adding to it the control $\mathbf{B}_C\mathbf{u}_C(t)$:

$$\begin{aligned}\dot{\mathbf{x}}_M(t) &= \mathbf{A}_M\mathbf{x}_M(t) + \mathbf{B}_M\mathbf{u}(t) + \mathbf{B}_C\mathbf{u}_C(t)\\ \mathbf{y}_M(t) &= \mathbf{C}_M\mathbf{x}_M(t).\end{aligned} \tag{6}$$

Here, $\mathbf{u}_C(t) \in \mathbb{R}^J$ is a control that will be designed to modify the dynamics of the original system (4) such that it is stable. For an LTI system representing some physical dynamics, $\mathbf{B}_C$ is typically selected to represent a physical control that can be imposed on the system, e.g., actuation applied to a boundary of a fluid domain. Next, a linear control law of the form $\mathbf{u}_C(t) = -\mathbf{K}_C\mathbf{x}_M(t)$ is assumed, where $\mathbf{K}_C \in \mathbb{R}^{J\times M}$ is the control matrix, to be determined. Substituting this law into (6) and rearranging, the following is obtained:

$$\begin{aligned}\dot{\mathbf{x}}_M(t) &= (\mathbf{A}_M - \mathbf{B}_C\mathbf{K}_C)\mathbf{x}_M(t) + \mathbf{B}_M\mathbf{u}(t)\\ \mathbf{y}_M(t) &= \mathbf{C}_M\mathbf{x}_M(t).\end{aligned} \tag{7}$$

The system (7) is a system of the form (4) but with $\mathbf{A}_M$ replaced by $\tilde{\mathbf{A}}_M$, where

$$\tilde{\mathbf{A}}_M \equiv \mathbf{A}_M - \mathbf{B}_C\mathbf{K}_C. \tag{8}$$

The reader can observe that if it is possible to compute the control matrix $\mathbf{K}_C$ such that $\tilde{\mathbf{A}}_M$ is stable, the ROM system (6) will be stable.

In order to formulate a well-posed ROM stabilization algorithm based on the approach outlined above, a number of questions need to be addressed:

1. How should the control matrix $\mathbf{B}_C$ be selected? Typically, when applying pole placement algorithms, a *physical* system is stabilized using a *physical* controller. In this case, the controller matrix $\mathbf{B}_C$ is added at the level of the algebraic system (6). In this context, what does $\mathbf{B}_C$ mean? What should it mean?

2. What eigenvalues should the stabilized ROM matrix $\tilde{\mathbf{A}}_M$ (8) be prescribed to have? It is clear that the eigenvalues should lie in the stable half of the complex plane, but what physical values should they have?
3. Does the solution $\mathbf{K}_C$ to the pole placement problem exist?
4. How has the stabilization affected the accuracy of the ROM? By modifying the ROM system (4), inconsistencies between the FOM and ROM physics have been introduced.

In this subsection, only question 3, the existence question, will be addressed. Answering this question gives rise to a preliminary ROM stabilization algorithm, referred to as "Algorithm 1". The remaining questions are addressed through the formulation of "Algorithm 2", described in Section 3.2.

Before formulating an algorithm which guarantees the existence of the solution to the pole placement problem described above, it is useful to recall the following theorem.

Theorem 3.1.1. (quoted from [6]): If the pair $(\mathbf{A}_M, \mathbf{B}_C)$ is controllable,[1] there exists a feedback $\mathbf{u}_C(t) = -\mathbf{K}_C\mathbf{x}_M$ such that the eigenvalues of $\tilde{\mathbf{A}}_M$ (8) can be arbitrarily assigned.

In general, the pair $(\mathbf{A}_M, \mathbf{B}_C)$ may not be controllable. However, it is possible to apply Theorem 3.1.1 by working in the controllable and observable[2] subspaces of $\mathbf{A}_M$ and $\mathbf{B}_C$, which can be isolated through the Kalman decomposition. A detailed discussion of the Kalman decomposition can be found in classical control theory texts, e.g., [6,7]. The key result of the Kalman theorem is that the state space can be decomposed into four parts: a part that is reachable and observable, a part that is reachable but not observable, a part that is not reachable but observable and a part that is neither reachable nor observable. The procedure is summarized in Algorithm 1.

Algorithm 1

- Pick a control matrix $\mathbf{B}_C$, e.g., $\mathbf{B}_C = \mathbf{1}_M$.
- Given $\mathbf{B}_C$, use the Kalman decomposition to isolate the controllable and observable parts of $\mathbf{A}_M$ and $\mathbf{B}_C$, call them $\mathbf{A}_M^{co} = \mathbf{U}\mathbf{A}_M\mathbf{U}^T$ and $\mathbf{B}_C^{co} = \mathbf{U}\mathbf{B}_C$ respectively.
- Compute the eigenvalues $\lambda_1^{co}, \ldots, \lambda_{M^{co}}^{co}$ of $\mathbf{A}_M^{co}$.
- Reassign the unstable eigenvalues of $\mathbf{A}_M^{co}$ to make them stable, e.g., for $k = 1$ to M^{co}, set

$$\lambda_k = \min\{Re(\lambda_k^{co}), -Re(\lambda_k^{co})\} + i \cdot Im(\lambda_k^{co}), \tag{11}$$

where $Re(z)$ and $Im(z)$ denote respectively the real and imaginary parts of a complex number $z \in \mathbb{C}$, and $i \equiv \sqrt{-1}$.
- Compute $\mathbf{K}_C$ such that $\mathbf{A}_M^{co} - \mathbf{K}_C\mathbf{B}_C^{co}$ has these eigenvalues using full state feedback (a.k.a pole placement) algorithms from control theory.
- Set $\mathbf{A}_M = \mathbf{U}^T(\mathbf{A}_M^{co} - \mathbf{K}_C\mathbf{B}_C^{co})\mathbf{U}$.

Typically in full state feedback, the matrix $\mathbf{B}_C$ represents a physical control that would be applied to a physical system of the form (4) so as to stabilize this system. The situation of interest here is not entirely comparable, as it has been assumed that the *physical* system underlying (4) *is* stable (and hence does not need stabilization via full state feedback); it is the *algebraic* ROM system (4) that is unstable, and hence the matrix $\mathbf{B}_C$ is added to the system at the algebraic level. This scenario complicates the interpretation of (and therefore the choice of) $\mathbf{B}_C$. In general, it can be argued that the choice of $\mathbf{B}_C$ does not matter provided the unstable eigenvalues of $\mathbf{A}_M$ are controllable and observable given the choice of $\mathbf{B}_C$. In the numerical example studied below (Section 4.1), $\mathbf{B}_C$ is selected to be a vector of all ones.

It remains to provide some discussion of approaches for selecting the eigenvalues of the stabilized matrix $\tilde{\mathbf{A}}_M$. One possible choice is to replace the real parts of the unstable eigenvalues of $\mathbf{A}_M$ with their negatives (11), or some negative scaled multiple of these values. Another option is to try to match the eigenvalues of the stabilized ROM matrix $\tilde{\mathbf{A}}_M$ with the eigenvalues of the FOM matrix $\mathbf{A}$ (provided the computational resources to compute the FOM eigenvalues are available, which may not be the case for very large systems). In general, the eigenvalues of a stable ROM will lie on or near the manifold of the eigenvalues of the FOM from which the ROM was constructed. This is illustrated in Fig. 1, which shows the eigenvalue manifold of the FOM matrix $\mathbf{A}$ and a ROM matrix $\mathbf{A}_M$ for an $M = 20$ mode ROM constructed via balanced truncation [16,5] for a variant of the inter-

[1] An LTI system (1) is controllable (a.k.a. reachable) if for any $\mathbf{x}_0, \mathbf{x}_f \in \mathbb{R}^N$, there exists a $T > 0$ and $\mathbf{u} : [0, T] \to \mathbb{R}$ such that the corresponding solution satisfies $\mathbf{x}(0) = \mathbf{x}_0$ and $\mathbf{x}(T) = \mathbf{x}_f$ [6,7]. To test for controllability of a linear system, it is sufficient to check the rank of the controllability matrix

$$\mathbf{W}_c \equiv (\mathbf{B}, \quad \mathbf{AB}, \quad \ldots, \quad \mathbf{A}^{N-1}\mathbf{B}). \tag{9}$$

The LTI system (1) is controllable if and only if the controllability matrix (9) is invertible [7,6].

[2] An LTI system (1) is observable if for any $T > 0$ it is possible to determine the state of the system $\mathbf{x}(T)$ through measurements of $\mathbf{y}(t)$ and $\mathbf{u}(t)$ on the interval $[0, T]$ [6,7]. To test for observability of a linear system, it is sufficient to check the rank of the observability matrix

$$\mathbf{W}_o^T \equiv (\mathbf{C}, \quad \mathbf{CA}, \quad \ldots, \quad \mathbf{CA}^{N-1}). \tag{10}$$

The LTI system (1) is observable if and only if the observability matrix (10) is full rank [6,7].

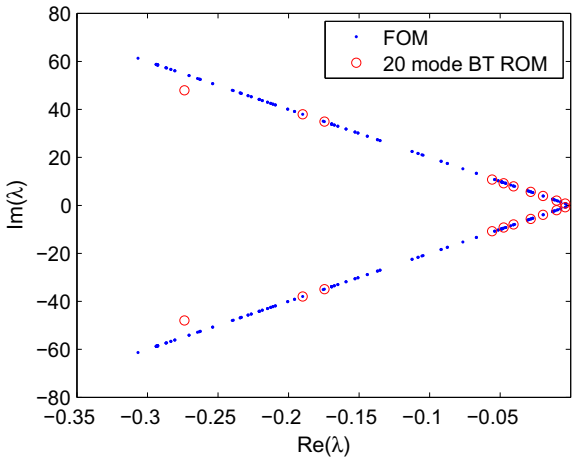


Fig. 1. Eigenvalue manifold of FOM matrix $\mathbf{A}$ and ROM matrix $\mathbf{A}_M$ for an $M = 20$ mode ROM constructed via balanced truncation for a variant of the ISS benchmark (Section 4.1).

national space station (ISS) benchmark (Section 4.1). In fact, if $M = N$ in a ROM, that is, a ROM is constructed with a full basis of the space $\mathbb{R}^N$, $\mathbf{A}_M \sim \mathbf{A}$ (as can be seen from (5)), so that $\mathbf{A}_M$ will have the same eigenvalues as $\mathbf{A}$.

3.2. Algorithm 2: ROM stabilization through solution of constrained nonlinear least squares optimization problem

The primary downside of Algorithm 1 (Section 3.1) is it is unclear *a priori* how a particular choice of the control matrix $\mathbf{B}_C$ and stabilized eigenvalues will affect the accuracy of the resulting stabilized ROM. This problem is remedied in the present section through the development of a new algorithm, "Algorithm 2". In this algorithm, the eigenvalues of the stabilized matrix $\tilde{\mathbf{A}}_M$ are determined such that the ROM output solution deviates minimally from the FOM output solution. Hence, questions 2 and 4 in Section 3.1 are addressed explicitly. As will be clear shortly, Algorithm 2 does not require the selection of a control matrix $\mathbf{B}_C$ (question 1).

Consider the ROM LTI system (4). Note that it is possible to work out analytically in closed form the exact solution to this system. The reader may verify that the solution to this system is given by

$$\mathbf{x}_M(t) = \exp(t\mathbf{A}_M)\mathbf{x}_M(0) + \int_0^t \exp\{(t-\tau)\mathbf{A}_M\}\mathbf{B}_M u(\tau)d\tau. \tag{12}$$

In Eq. (12), $\exp(\cdot)$ denotes the matrix exponential. It is worthwhile to note that this quantity is not an issue to compute, as the ROM system matrix $\mathbf{A}_M$ is small. Given the solution for the ROM state vector (12), the ROM output is given by

$$\mathbf{y}_M(t) = \mathbf{C}_M\left[\exp(t\mathbf{A}_M)\mathbf{x}_M(0) + \int_0^t \exp\{(t-\tau)\mathbf{A}_M\}\mathbf{B}_M u(\tau)d\tau\right]. \tag{13}$$

The existence of an analytical solution to the ROM LTI system (4) motivates the formulation of the following optimization problem, to be solved for the eigenvalues of the stabilized ROM system:

$$\begin{aligned} &\min_{\lambda_i^u} \quad \sum_{k=1}^{K} ||\mathbf{y}^k - \mathbf{y}_M^k||_2^2. \\ &\text{s.t.} \quad Re(\lambda_i^u) < 0, i = 1, \ldots, L \end{aligned} \tag{14}$$

The optimization is over the unstable eigenvalues of the original ROM system matrix $\mathbf{A}_M$, denoted by λ_i^u, for $i = 1, .., L$ where $L \leqslant M$ is the number of unstable eigenvalues of $\mathbf{A}_M$. The shorthand $\mathbf{y}^k$ denotes the FOM output at time t_k, i.e., $\mathbf{y}^k \equiv \mathbf{y}(t_k)$. In a model reduction approach based on an empirical basis computed from a set of snapshots of the high-fidelity solution, e.g., the POD or BPOD method, these values are available at the snapshot times. The shorthand $\mathbf{y}_M^k$ denotes the ROM output at time t_k, i.e., $\mathbf{y}_M^K \equiv \mathbf{y}_M(t_k)$. It is given by the formula (13). The constraint in (14) ensures that the stabilized ROM eigenvalues are in the stable half of the complex plane Here $Re(z)$ denotes the real part of a complex number $z \in \mathbb{C}$. Eq. (14) is a constrained nonlinear least-squares optimization problem with inequality constraints.

Remark that the optimization problem (14) is small: there are at most M dofs, and solving the problem does not require operating on any matrices that are of size $\mathcal{O}(N)$. This optimization problem can be solved using standard algorithms for constrained optimization, e.g., an SQP algorithm with line search globalization, BFGS for Hessian approximations, and an interior point method to handle the inequality constraints [30].

An interesting question that arises is whether the solution to the optimization problem (14) is unique. A sufficient condition for a minimization problem of the form

$$\min_x f(x), \tag{15}$$

where $x \in \mathbb{R}^n$ is a real vector and $f : \mathbb{R}^n \to \mathbb{R}$ is a smooth function, to have a unique solution is for f to be convex [30]. In this case, any stationary point of f is a global minimizer of f, and hence a local minimizer of f will be the global minimizer of f. It is straightforward to show that the objective function in (14) is not necessarily convex. Since convexity is a sufficient but not a necessary condition for uniqueness of the solution to (14), the optimization problem could have a unique solution, but this scenario is not guaranteed. The numerical tests performed in Section 4 suggest that the optimization problem (14) has in general multiple solutions.

It turns out that it is convenient to implement and solve the optimization problem (14) in the "characteristic variables", defined by $\mathbf{z}_M(t) = \mathbf{S}_M^{-1}\mathbf{x}_M(t)$, where $\mathbf{S}_M^{-1}$ is the matrix that diagonalizes $\mathbf{A}_M$, i.e., $\mathbf{A}_M = \mathbf{S}_M\mathbf{D}_M\mathbf{S}_M^{-1}$. The steps of the stabilization are detailed in Algorithm 2. Note that, although it is assumed here $\mathbf{A}_M$ is diagonalizable, the extension to non-diagonalizable $\mathbf{A}_M$ is straightforward. In this case, the eigenvalue decomposition in Algorithm 2 (16) is replaced with the Jordan decomposition.

Algorithm 2

- Diagonalize the ROM matrix $\mathbf{A}_M$:

$$\mathbf{A}_M = \mathbf{S}_M\mathbf{D}_M\mathbf{S}_M^{-1}. \tag{16}$$

- Initialize a diagonal $M \times M$ matrix $\tilde{\mathbf{D}}_M$.
- Set $j = 1$.
- **for** $i = 1$ to M
  - **if** $Re(D_M(i,i)) < 0$
    - Set $\tilde{D}_M(i,i) = D_M(i,i)$.
  - **else**
    - Set $\tilde{D}_M(i,i) = \lambda_j^u$.
  - **endif**
- **endfor**
- Increment $j \leftarrow j + 1$.
- Solve the optimization problem (14) for the eigenvalues $\{\lambda_j^u\}$ with $\mathbf{y}_M(t)$ given by

$$\mathbf{y}_M(t) = \mathbf{C}_M\left[\mathbf{S}_M \exp(t\tilde{\mathbf{D}}_M)\mathbf{S}_M^{-1}\mathbf{x}_M(0) + \int_0^t \mathbf{S}_M \exp\{(t-\tau)\tilde{\mathbf{D}}_M\}\mathbf{S}_M^{-1}\mathbf{B}_M u(\tau)d\tau\right], \tag{17}$$

using an optimization algorithm.

- Evaluate $\tilde{\mathbf{D}}_M$ at the solution of the optimization problem (14).
- The stabilized LTI ROM system is now given by

$$\begin{aligned}\dot{\mathbf{x}}_M(t) &= \tilde{\mathbf{A}}_M\mathbf{x}_M(t) + \mathbf{B}_M\mathbf{u}(t)\\ \mathbf{y}_M(t) &= \mathbf{C}_M\mathbf{x}_M(t),\end{aligned} \tag{18}$$

where $\tilde{\mathbf{A}}_M = \mathbf{S}_M\tilde{\mathbf{D}}_M\mathbf{S}_M^{-1}$.

3.3. Connection between Algorithm 1 and Algorithm 2

One notable difference between Algorithms 1 and 2 is that, unlike the former algorithm, the latter algorithm does not employ directly full state feedback (a.k.a. pole placement) routines from control theory to solve for the stabilized ROM matrix $\tilde{\mathbf{A}}_M$. However, it turns out that it is possible to show that Algorithm 2 is equivalent to Algorithm 1 for a specific choice of control matrices $\mathbf{B}_C$ and $\mathbf{K}_C$.

Suppose $\mathbf{A}_M$ has $L \leqslant M$ unstable eigenvalues λ_k^u, each with corresponding eigenvector $\mathbf{s}_k^u$. Let $\tilde{\lambda}_k^u$ denote the stabilized value of λ_k^u, obtained by solving the optimization problem (14). The reader can verify that $\tilde{\mathbf{A}}_M$ in (18) is equivalent to

$$\tilde{\mathbf{A}}_M = \mathbf{A}_M - \mathbf{B}_C\mathbf{K}_C, \tag{19}$$

where

$$\mathbf{B}_C = (\mathbf{s}_1^u, \quad \ldots, \quad \mathbf{s}_L^u) \in \mathbb{R}^{M\times L} \tag{20}$$

$$\mathbf{K}_C = \begin{pmatrix} \lambda_1^u - \tilde{\lambda}_1^u & 0 & 0 & \cdots & 0 \\ 0 & \lambda_2^u - \tilde{\lambda}_2^u & 0 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & \lambda_L^u - \tilde{\lambda}_L^u & 0 \end{pmatrix} \mathbf{S}_M^{-1} \in \mathbb{R}^{L\times M}. \tag{21}$$

4. Numerical experiments

The performance of the ROM stabilization algorithms described in Section 3 is now assessed on two benchmarks: the international space station (ISS) benchmark (Section 4.1 benchmark involving a one-dimensional model of an electrostatically actuated beam (Section 4.2). Although the applications considered in this section come from the field of structural mechanics, the ROM stabilization algorithms developed in this work can potentially be used to build stable ROMs for *any* application. For both test cases, the reduced basis $\mathbf{\Phi}_M$ is constructed using the POD, and the projection step is a Galerkin projection in the L^2 inner product. Discussed in detail in Lumley [8] and Holmes et al. [9], POD is a mathematical procedure in which, given an ensemble of data and an inner product, an empirical basis is constructed. This basis, the POD basis, is optimal in the sense that it describes more energy (on average) of the ensemble in the chosen inner product than any other linear basis of the same dimension M. For a discussion of the details of the POD algorithm, the reader is referred to [8,9].

Typically, the size of a reduced POD basis, namely M, is calculated using an energy criterion. That is, M is selected such that the reduced basis $\mathbf{\Phi}_M$ captures some fixed percentage of the snapshot energy, e.g., 95% or 99% (see [8,9]). For the problems considered here, M is chosen to be the smallest integer such that: (1) the basis $\mathbf{\Phi}_M$ captures at least 99% of the snapshot energy, (2) the resulting POD/Galerkin ROM has at least one unstable eigenvalue, and (3) the POD/Galerkin ROM goes unstable during the time horizon considered. This strategy of choosing M is a natural one given the objective of this paper: to evaluate the ROM stabilization algorithms developed in Section 3.

For both test cases considered, the error in the ROM output relative to the FOM output is calculated and reported. This error is denoted $\mathcal{E}_{rel}$ and computed according to the following formula:

$$\mathcal{E}_{rel} = \sqrt{\frac{\sum_{k=1}^{K} ||\mathbf{y}^k - \mathbf{y}_M^k||_2^2}{\sum_{k=1}^{K} ||\mathbf{y}^k||_2^2}}. \tag{22}$$

Here, $\mathbf{y}^k \equiv \mathbf{y}(t_k)$ denotes the snapshot FOM output at time t_k, and $\mathbf{y}_M^k \equiv \mathbf{y}_M(t_k)$ denotes the ROM output at time t_k.

For the ISS example (Section 4.1) the performance of Algorithm 1 and the performance of Algorithm 2 are evaluated. This comparison is intended to highlight the superiority of Algorithm 2 over Algorithm 1. For the sake of brevity, results for only Algorithm 2 (established in the context of the ISS example as the superior algorithm) are shown for the electrostatically actuated beam example (Section 4.2).

The `place` function in the MATLAB control toolbox [34] is used to solve the pole placement problem at the heart of Algorithm 1. To solve the constrained nonlinear least squares optimization at the heart of Algorithm 2 (14), the `fmincon` function in the MATLAB optimization toolbox [29,30] is employed. The `Algorithm` option required by this function is set to `interior-point` with exact (analytic) Jacobians. An analytic expression for the Jacobian of the objective function for the specific case of $\mathbf{u}(t) = \mathbf{0}$ and one output of interest in (14) can be found in Section A.1 of the Appendix. Deriving and implementing an analytic Jacobian is recommended over using finite difference Jacobians calculated within the MATLAB optimization toolbox. Since analytic Jacobians are exact, they are accurate. In contrast, finite difference Jacobians can be inaccurate for some problems as a result of an arbitrary selection of the finite difference increment. Moreover, the solution of the optimization problem (14) is much faster with exact Jacobian due to fewer required function evaluations. With exact Jacobians, the number of function evaluations per optimization step is constant. In particular, it does not grow with L, the number of eigenvalues reassigned by the optimization algorithm. The default `fmincon` settings for this method are used, which can be found in [29].

Note that the `fmincon` function will compute only real solutions to an optimization problem. In general the eigenvalues of the matrix $\mathbf{A}_M$ may be complex, however. To allow the `fmincon` algorithm to compute complex eigenvalue solutions of the ROM stabilization optimization problem (14), a complex-valued functional form for λ_j^u may be assumed. In this approach, λ_j^u in line 9 of Algorithm 2 is replaced with

$$\lambda_j^u \leftarrow \lambda_j^{ur} + \boldsymbol{i} \cdot \lambda_j^{uc} \in \mathbb{C}, \quad \lambda_j^{ur}, \lambda_j^{uc} \in \mathbb{R}, \tag{23}$$

(where $i \equiv \sqrt{-1}$) and (14) is solved for $\lambda_j^{ur}, \lambda_j^{uc} \in \mathbb{R}$ subject to the constraint that $\lambda_j^{ur} < 0$. Since complex eigenvalues of $\mathbf{A}_M$ occur in complex-conjugate pairs, if λ_j^u has the form (23), then λ_{j+1} in Algorithm 2 must have the form

$$\lambda_{j+1}^u = \lambda_{j+1}^{ur} - \boldsymbol{i} \cdot \lambda_{j+1}^{uc} \in \mathbb{C}, \quad \lambda_{j+1}^{ur}, \lambda_{j+1}^{uc} \in \mathbb{R}. \tag{24}$$

It follows that the approach of assuming complex-conjugate pair solutions to (14) does not give rise to more dofs than the default approach of solving for real solutions to this problem. In fact, the former approach has fewer constraints.

The numerical results section includes comparisons of the following CPU times for both problems considered:

- The CPU time required for the time-integration of the FOM.
- The CPU time required for the offline (snapshot collection, loading of system matrices/snaptions, calculation of the POD basis, Galerkin projection, and numerical solution of the optimization problem (14)) stage of the POD/Galerkin ROMs.
- The CPU time required for the online (time-integration) stage of the POD/Galerkin ROMs.

All computations are performed in serial using MATLAB's linear algebra capabilities on a Linux workstation with 6 Intel Xeon 2.93 GHz CPUs. Note that the FOM CPU times do not include the time to discretize the relevant PDEs using the finite

element method and assemble the global system matrix. This is due to the fact that the matrices defining the FOM were downloaded from a model reduction benchmark repository, and access to the high-fidelity code that generated these matrices is not available to the authors.

In general, ROMs are employed for many-query and/or real-time analysis. In these contexts, it is critical that the online time-integration stage of the ROM has a low computational cost and fast run-time. Although the offline construction of the reduced order model, which includes the collection of snapshots, the construction of the POD basis, the Galerkin projection, and the solution of the optimization problem (14), can be computationally intensive, this step is done only *one* time when the ROM is constructed. The cost of this computation does not affect the run-time of the online step of the model reduction, the step relevant to analysis using the ROM. Nonetheless, it may be of interest how many times the ROM would need to be run (online) to compensate the cost of the (offline) pre-processing step. For this reason, estimates of the number of online ROM runs that would be required to offset the offline ROM cost are given for each example considered following the CPU time data (Tables 5 and 9).

4.1. International space station problem

The first numerical example considered here involves a structural model of the Russian service module component of the international space station (ISS) [21]. This service module is a large flexible structure whose dynamics can be described using a linearized form of the equations of motion (a second order PDE system). Written in first order LTI form, the model consists of a system of the form (1) with $N = 270$. The matrices $\mathbf{A}$, $\mathbf{B}$ and $\mathbf{C}$ defining (1) are downloaded from the ROM benchmark repository [24]. The matrix $\mathbf{A}$ is sparse, as it comes from a finite element discretization. A single output is considered, corresponding to the first row of the matrix $\mathbf{C}$. Since this problem is unforced $\mathbf{u}(t) = \mathbf{0}$, the solution behavior as $t \to \infty$ depends only on the real parts of the eigenvalues of the system matrix $\mathbf{A}$. It is verified that the FOM system is stable: the maximum real part of the eigenvalues of $\mathbf{A}$ is -0.0031. The FOM will be reduced using the POD/Galerkin method [13,14,9].

To generate the snapshots from which a POD basis will be constructed, the full order model (1) is solved using a backward Euler time integration scheme with an initial condition of $\mathbf{x}_N(0) = \mathbf{1}_N$ ($N \times 1$ vector of all ones) and no input ($\mathbf{u}(t) = \mathbf{0}$). A total of $K = 2000$ snapshots are collected, every $dt_{snap} = 5 \times 10^{-5}$, until time $t = 0.1$. These snapshots are used to compute a POD basis of size $M = 20$, and a POD/Galerkin ROM of size $M = 20$ is constructed using this basis. For this problem, the $M = 20$ mode POD/Galerkin ROM is found to be unstable with four unstable eigenvalues. This basis captures essentially 100% of the snapshot energy, and the value $M = 20$ is the smallest basis size such that the ROM exhibits an instability. The numerical values of the unstable eigenvalues are: $\lambda_1^u = 242.5$, $\lambda_2^u = 32.90 + 26.99i$, $\lambda_3^u = 32.90 - 26.99i$, $\lambda_4^u = 2.712$. Fig. 2 shows the FOM output $\mathbf{y}(t)$ (in red) compared to the unstabilized ROM output (in blue). The unstabilized ROM output diverges from the FOM output around time $t = 0.05$ and approaches $-\infty$ as $t \to \infty$ due to the ROM instability. The relative error $\mathcal{E}_{rel}$ in the unstabilized ROM output (22) is 1737.9.

The $M = 20$ mode POD/Galerkin ROM for the ISS problem is stabilized first by Algorithm 1, then by Algorithm 2. These results illustrate the superiority of Algorithm 2 over Algorithm 1.

4.1.1. Stabilization via Algorithm 1

First, the $M = 20$ mode unstable POD/Galerkin ROM is stabilized using Algorithm 1 with the control matrix $\mathbf{B}_C$ selected to be an $M \times 1$ vector of all ones: $\mathbf{B}_C = \mathbf{1}_M$. The next step in the stabilization is to select the desired eigenvalues of the stabilized

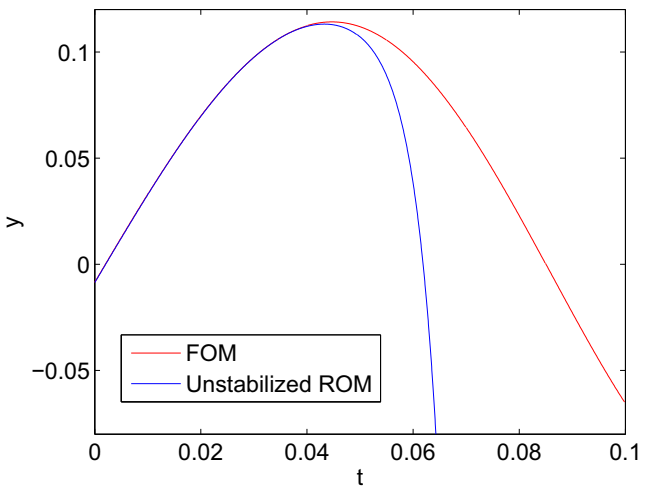


Fig. 2. Outputs for $M = 20$ unstabilized POD/Galerkin ROM vs. FOM output for ISS problem.

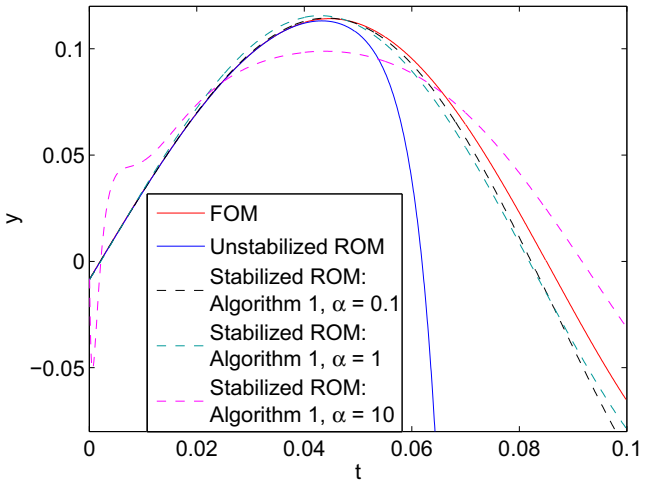


Fig. 3. Outputs for $M = 20$ POD/Galerkin ROMs stabilized via Algorithm 1 vs. FOM output for ISS problem.

ROM matrix $\tilde{\mathbf{A}}_M$. Let λ_k^u for $k = 1, \ldots, 4$ denote the unstable eigenvalues for $\mathbf{A}_M$, and let $\tilde{\lambda}_k^u$ denotes the corresponding eigenvalues of $\tilde{\mathbf{A}}_M$ (that is, the values λ_k^u will be replaced within the stabilization algorithm). Here, the following functional form for $\tilde{\lambda}_k^u$ will be considered:

$$\tilde{\lambda}_k^u = -\alpha \cdot Re(\lambda_k^u) + i \cdot Im(\lambda_k^u), \alpha > 0, \tag{25}$$

for $k = 1, \ldots, 4$, where $Re(z)$ and $Im(z)$ denote respectively the real and imaginary parts of a complex number $z \in \mathbb{C}$ and $i \equiv \sqrt{-1}$. The transformation (25) flips the sign of the real part of an unstable eigenvalue of $\mathbf{A}_M$ (thereby making it stable), and scales this value by a positive constant α. Three choices of the parameter α in (25) will be tested here:

- $\alpha = 0.1$.
- $\alpha = 1$.
- $\alpha = 10$.

The objective is to study the error in the stabilized ROM for several choices of $\tilde{\lambda}_i^u$. The choices are admittedly ad hoc, as there is no clear guideline for what the eigenvalues of $\tilde{\mathbf{A}}_M$ should be. Note that as α is increased, the eigenvalues $\tilde{\lambda}_k^u$ are pushed further into the left (stable) half of the complex plane.

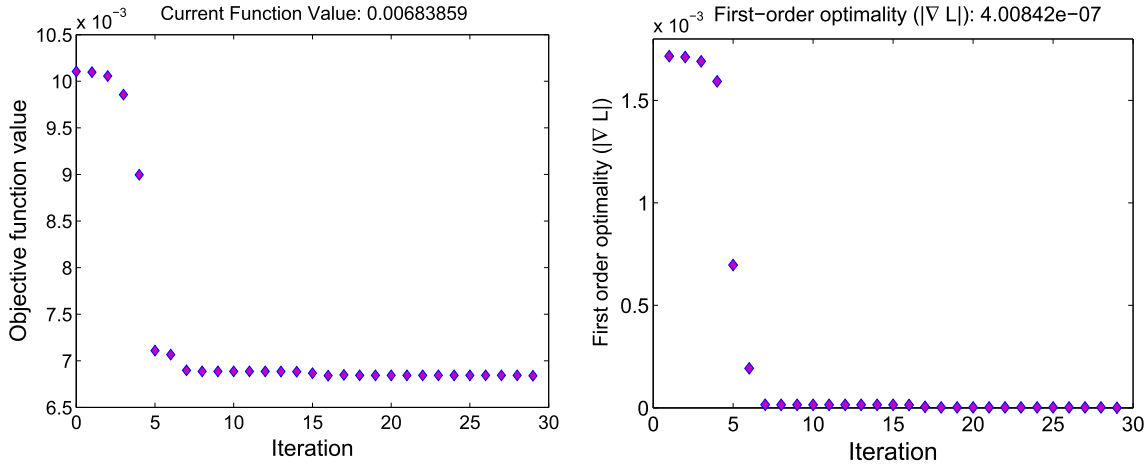
Fig. 3 shows the outputs computed by the three stabilized ROMs obtained using Algorithm 1. To solve the pole placement problem at the heart of this algorithm, the `place` command in the MATLAB control toolbox [34] was employed. The relative errors in the stabilized ROM outputs are given in Table 1. All three ROMs are stable (by construction). The ROM stabilized by

Table 1
Relative errors in $M = 20$ POD/Galerkin ROM for ISS problem stabilized via Algorithm 1.

ROM	$\mathcal{E}_{rel}$
Unstabilized	1737.8
ROM stabilized via Algorithm 1 with $\alpha = 0.1$	1.51×10^{-2}
ROM stabilized via Algorithm 1 with $\alpha = 1$	1.16×10^{-2}
ROM stabilized via Algorithm 1 with $\alpha = 10$	2.26×10^{-2}

Table 2
Performance of `fmincon` interior point method for Algorithm 2 applied to ISS problem.

	Algorithm 2 with Option 1 (real eigenvalues)	Algorithm 2 with Option 2 (complex-conjugate eigenvalues)
# Upper bound constraints	4	3
# Optimization iterations	29	27
# Function evaluations	30	30
First order optimality at convergence ($\|\nabla L\|$)	4.00×10^{-7}	5.51×10^{-7}



(a) Function value (b) First order optimality ($|\nabla L|$)

Fig. 4. Performance of interior point algorithm for Algorithm 2 with Option 1 (real eigenvalues) as a function of iteration number (ISS problem).

Algorithm 1 with $\alpha = 1$ is slightly more accurate than the ROM stabilized by Algorithm 1 with $\alpha = 0.1$. This may lead the reader to conjecture that the accuracy of the stabilized ROM will improve as the eigenvalues are pushed further and further into the left half of the complex plane. However, the ROM stabilized by Algorithm 1 with $\alpha = 10$ results demonstrate that this is not the case: the ROM with its eigenvalues pushed the most into the left half of the complex plane is the least accurate.

The numerical results presented here show that Algorithm 1 works in the sense that it will stabilize an unstable ROM. Unfortunately, the accuracy of a ROM stabilized using this algorithm is in general unknown before the ROM is stabilized and the ROM output is computed. Moreover, for some choices of $\tilde{\lambda}_i^u$ the accuracy may be unacceptable.

4.1.2. Stabilization via Algorithm 2

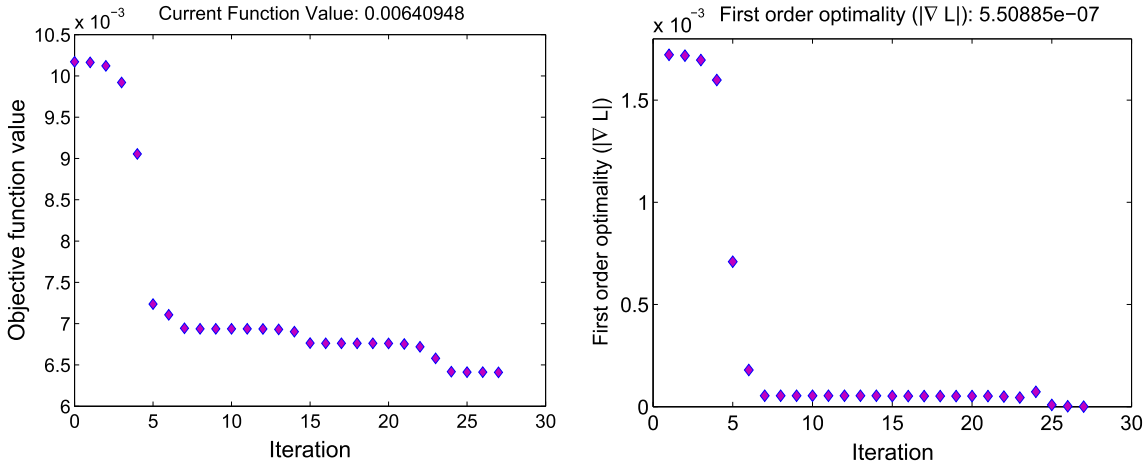
The $M = 20$ POD/Galerkin ROM for the ISS benchmark is now stabilized using Algorithm 2. Let λ_k^u for $k = 1, \ldots, 4$ denote the four unstable eigenvalues of $\mathbf{A}_M$. Two options for the eigenvalue solutions to the optimization problem (14) are considered:

- *Option 1:* Solve for $\lambda_i^u \in \mathbb{R}$ subject to the constraint that $\lambda_i^u < 0$ for $i = 1, \ldots, 4$.
- *Option 2:* Solve for $\lambda_1, \lambda_2^{ur}, \lambda_2^{uc}, \lambda_4 \in \mathbb{R}$ subject to the constraint that $\lambda_1, \lambda_2^{ur}, \lambda_4 < 0$ and set $\lambda_2^u = \lambda_2^{ur} + i\lambda_2^{uc}$, $\lambda_3^u = \lambda_2^{ur} - i\lambda_2^{uc}$ (that is, λ_3^u is set to be the complex-conjugate of $\lambda_2^u : \lambda_3^u = \bar{\lambda_2}^u$).

Per the discussion at the beginning of Section 4, Option 2 is more general than Option 1 and has fewer inequality constraints. The optimization problem (14) at the heart of Algorithm 2 is solved using the `fmincon` function in MATLAB's optimization toolbox. The `Algorithm` option required by this function is set to `interior-point`, and an initial guess of -1 for all the variables is used. For functional forms of the eigenvalues given by both Option 1 and Option 2, the optimization algorithm converges to a local minimum solution in less than 30 optimization iterations and 30 function evaluations. Table 2 shows some key information about the convergence of the optimization algorithm. The reader may observe that fewer iterations and function evaluations are required with Option 2 than with Option 1, which has more constraints. Figs. 4 and 5 illustrate further the performance of the optimization algorithm for Option 1 and Option 2 respectively. For both options, the optimality conditions are satisfied to the specified tolerance at the value of the optimal solution.[3]

An interesting question that arises is how the numbers in Table 2 change with M, the reduced basis size. Numerical experiments reveal that it is not necessarily the case that as M increases, more optimization iterations and function evaluations are required to obtain the solution to the optimization problem (14). The performance of the interior point method depends on a number of factors, including: (1) the number of optimization dofs (i.e., the number of unstable eigenvalues of a ROM), (2) the number of upper bound constraints, (3) the character of the objective function, (4) the proximity of the initial guess to the optimal solution, and (5) the tolerances used in the optimization algorithm; not M, the reduced basis size, directly. Some additional performance results of the `fmincon` interior point method for Algorithm 2 applied to the ISS problem for different (larger) values of M are given in Appendix A.2 (Tables 10,11). For the ISS problem, the ROM does in general become more unstable with increasing M, but more optimization iterations are not always required (Table 10).

[3] For a constrained optimization problem such as (14), the first order optimality conditions require that the gradient of Lagrangian of the objective function $L(\lambda_1^u, \ldots, \lambda_L^u)$ be equal to zero, i.e., $\frac{\partial L}{\partial \lambda_k^u} = 0$ for all $k = 1, \ldots, L$ where $L < M$ is the number of eigenvalues of $\mathbf{A}_M$ stabilized by Algorithm 2. A detailed discussion of this and other optimality conditions for the problem (14) can be found in [29,30].



(a) Function value (b) First order optimality ($|\nabla L|$)

Fig. 5. Performance of interior point algorithm for Algorithm 2 with Option 2 (complex-conjugate eigenvalues) as a function of iteration number (ISS problem).

Table 3
Original (unstable) eigenvalues of $\mathbf{A}_M$ for $M = 20$ mode POD/Galerkin ROM and new stable eigenvalues computed using Algorithm 2 (ISS problem).

	Original unstable $\mathbf{A}_M$	Algorithm 2 with Option 1 (real eigenvalues)	Algorithm 2 with Option 2 (complex-conjugate eigenvalues)
λ_1^u	2.42×10^2	-1.32	-1.98
λ_2^u	$3.29 \times 10^1 + 2.70 \times 10^1 i$	-2.12×10^{-2}	$-6.47 \times 10^{-3} + 1.42 \times 10^1 i$
λ_3^u	$3.29 \times 10^1 - 2.70 \times 10^1 i$	-2.13×10^{-2}	$-6.47 \times 10^{-3} - 1.42 \times 10^1 i$
λ_4^u	2.71	-1.33×10^{-4}	-1.38×10^{-4}

Table 4
Relative errors in $M = 20$ POD/Galerkin ROM for ISS problem stabilized via Algorithm 2.

ROM	$\mathcal{E}_{rel}$
Unstabilized	1.74×10^3
ROM stabilized via Algorithm 2 with Option 1 (real eigenvalues)	2.59×10^{-2}
ROM stabilized via Algorithm 2 with Option 2 (complex-conjugate eigenvalues)	2.52×10^{-2}

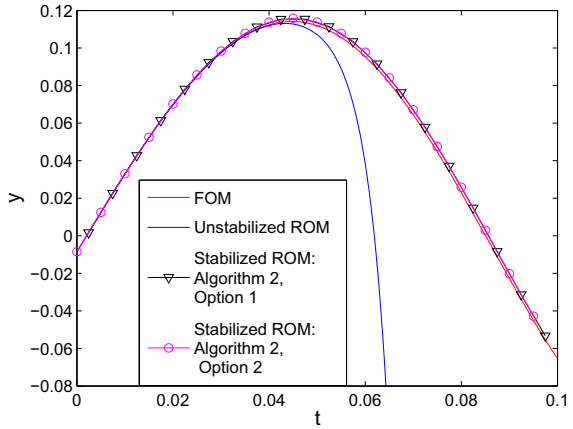


Fig. 6. Outputs for $M = 20$ POD/Galerkin ROMs stabilized via Algorithm 2 vs. FOM output for ISS problem.

The eigenvalue solutions to the optimization problem (14) with both Option 1 and Option 2 are given in Table 3, compared with the values of the original unstable eigenvalues of $\mathbf{A}_M$. It is interesting to observe that the eigenvalues computed by the optimization algorithm with Option 1 are very different in their numerical values than those computed by the optimization algorithm with Option 2. Both are local minimizers of the optimization function (14). As discussed in Section 3.2, the optimization value is not guaranteed to be unique.

Table 4 gives the error in the ROM algorithm relative to the FOM output for an $M = 20$ POD/Galerkin ROM stabilized via Algorithm 2 with Option 1 and Option 2 for the ISS problem. Both options give a ROM with a relative error between 2.5% and 2.6%. This is a significant improvement in accuracy compared to the same ROM stabilized via Algorithm 1 (Table 1). Most importantly, in contrast to Algorithm 1, Algorithm 2 guarantees some level of accuracy in the stabilized ROM, as it minimizes the error in the ROM output by construction. Recall that the accuracy of a ROM stabilized via Algorithm 1 is unknown *a priori*, and it may require some trial and error to obtain a stabilized ROM with an acceptable error (Section 4.1.1).

Fig. 6 shows the output computed from ROMs stabilized using Algorithm 2. The reader may observe that the stabilized ROM outputs are in much better agreement with the FOM output than the ROMs stabilized using Algorithm 1 (Fig. 3).

Table 5 summarizes the CPU times for the time-integration step of the FOM, in addition to the CPU times for the offline and online stages of the $M = 20$ POD/Galerkin ISS ROM. The reader can observe by examining Table 5 that the $M = 20$ online stage of the POD/Galerkin ROM requires approximately 45 times less CPU time than the time-integration stage of the FOM. To offset the total preprocess time of the ROM (the time required to run the FOM to collect snapshots, calculate the POD basis, perform the Galerkin projection, and solve the optimization problem (14)), the ROM would need to be run approximately 53 times. It is worthwhile to note that that the optimization step of the model reduction, which consists of the solution of the optimization problem (14) is very fast: it takes less than a minute to complete.

4.2. Electrostatically actuated beam problem

The second numerical example is that of an electrostatically actuated beam. The purpose of this second example is to verify the proposed ROM stabilization approach for a different application and to demonstrate the methodology presented in this paper on a larger-scale problem which has a forcing term ($\mathbf{B}_M\mathbf{u}(t) \neq \mathbf{0}$). Applications for this model include microelectromechanical systems (MEMS) devices such as electromechanical radio frequency (RF) filters [26]. Given a simple enough shape, these devices can be modeled as one-dimensional beams embedded in two or three dimensional space. The beam considered here is supported on both sides, and has two dofs: the deflection perpendicular to the beam (the flexural displacement), and the rotation in the deformation plane (the flexural rotation). The equations of motion are determined from a Lagrangian formulation. It is assumed that the beam deflection is small, so that geometric nonlinearities can be neglected. The resulting linear PDEs are discretized using the finite element method following the approach presented in [27,26]. The result of this discretization is a second order linear semi-discrete system of the form:

$$\begin{aligned} \mathbf{M}\ddot{\mathbf{x}}(t) + \mathbf{E}\dot{\mathbf{x}}(t) + \mathbf{K}\mathbf{x}(t) &= \mathbf{B}\mathbf{u}(t) \\ \mathbf{y}(t) &= \mathbf{C}\mathbf{x}(t), \end{aligned} \tag{26}$$

where $\ddot{\mathbf{x}} \equiv \frac{\partial^2 \mathbf{x}}{\partial t^2}$. The input matrix $\mathbf{B}$ corresponds to a loading of the middle node of the domain, and $\mathbf{y}(t)$ is the flexural displacement at the middle node of the domain. The damping matrix $\mathbf{E}$ is taken to be a linear combination of the mass matrix $\mathbf{M}$ and the stiffness matrix $\mathbf{K}$:

$$\mathbf{E} = c_M\mathbf{M} + c_K\mathbf{K}, \tag{27}$$

with $c_M = 10^2$ and $c_K = 10^{-2}$. Letting $\dot{\tilde{\mathbf{x}}}(t) \equiv \mathbf{x}(t)$, the second order system (26) can be written as the following first order system:

$$\begin{aligned} &\begin{pmatrix} \mathbf{E} & \mathbf{M} \\ \mathbf{I} & \mathbf{0} \end{pmatrix} \begin{pmatrix} \dot{\mathbf{x}}(t) \\ \dot{\tilde{\mathbf{x}}}(t) \end{pmatrix} + \begin{pmatrix} \mathbf{K} & \mathbf{0} \\ \mathbf{0} & -\mathbf{I} \end{pmatrix} \begin{pmatrix} \mathbf{x}(t) \\ \tilde{\mathbf{x}}(t) \end{pmatrix} = \begin{pmatrix} \mathbf{B} \\ \mathbf{0} \end{pmatrix} \mathbf{u}(t) \\ &\mathbf{y}(t) = \begin{pmatrix} \mathbf{C} & \mathbf{0} \end{pmatrix} \begin{pmatrix} \mathbf{x}(t) \\ \tilde{\mathbf{x}}(t) \end{pmatrix}, \end{aligned} \tag{28}$$

or

$$\begin{aligned} \dot{\mathbf{z}}(t) &= \mathbf{A}\mathbf{z}(t) + \tilde{\mathbf{B}}\mathbf{u}(t) \\ \mathbf{y}(t) &= \tilde{\mathbf{C}}\mathbf{z}(t), \end{aligned} \tag{29}$$

where $\mathbf{z}(t) \equiv \begin{pmatrix} \mathbf{x}(t) \\ \tilde{\mathbf{x}}(t) \end{pmatrix} \in \mathbb{R}^{2N}$ and

$$\mathbf{A} \equiv \begin{pmatrix} \mathbf{0} & \mathbf{I} \\ -\mathbf{M}^{-1}\mathbf{K} & -\mathbf{M}^{-1}\mathbf{E} \end{pmatrix}, \quad \tilde{\mathbf{B}} \equiv \begin{pmatrix} \mathbf{0} \\ \mathbf{M}^{-1}\mathbf{B} \end{pmatrix}, \quad \tilde{\mathbf{C}} \equiv \begin{pmatrix} \mathbf{C} & \mathbf{0} \end{pmatrix}. \tag{30}$$

The matrices $\mathbf{M}$ and $\mathbf{K}$ in (26) are downloaded from the Oberwolfach model reduction benchmark collection [28]. These matrices are sparse, as they come from a finite element discretization. These global matrices are then disassembled into their local counterparts, and reassembled to yield a discretization of any desired size. In the full order model for which results are

reported here, $N = 5000$, so (29) has 10,000 dofs. It is verified that the full order system is stable: the maximum real part of the eigenvalues of $\mathbf{A}$ is -0.0016. As for the ISS example, for FOM (29) will be reduced using the POD/Galerkin method [13,14,9]. It is worthwhile to note that, unlike for the ISS example, the matrix $\mathbf{A}$ that defines the system (29) for the electrostatically actuated beam test case is *not* sparse. In particular, it is straightforward to see from (29) that this matrix is of the form $\mathbf{A} = (\mathbf{A}_1, \quad \mathbf{A}_2)^T$ where $\mathbf{A}_1 \in \mathbb{R}^{N\times N}$ is sparse, but $\mathbf{A}_2 \in \mathbb{R}^{N\times N}$ is dense. This example tests therefore the performance of Algorithm 2 on a problem defined by a dense matrix $\mathbf{A}$.

To generate the snapshots from which POD bases are constructed, the full order model (29) is solved using a backward Euler time integration scheme with an initial condition of $\mathbf{z}(0) = \mathbf{0}$ and an input corresponding to a periodic on/off switching, i.e.,

$$\mathbf{u}(t) = \begin{cases} 1, & 0.005 < t < 0.01,\ 0.015 < t < 0.02,\ 0.03 < t < 0.035 \\ 0, & \text{otherwise.} \end{cases} \tag{31}$$

A total of $K_{max} = 1000$ snapshots are collected, every $dt_{snap} = 5 \times 10^{-5}$ s, until time $t = 0.05$ s. From these snapshots, an $M = 17$ mode POD/Galerkin ROM is constructed. The ROM is found to be unstable, with four unstable eigenvalues. These eigenvalues have the following numerical values: $\lambda_1^u = 16,053$, $\lambda_2^u = 48.985$, $\lambda_3^u = 12.650$, $\lambda_4^u = 0.05202$. The basis size $M = 17$ is selected since this is the smallest integer for which the ROM exhibits an instability. It captures effectively 100% of the snapshot energy. Fig. 7 shows the FOM output $\mathbf{y}(t)$ (in red) compared to the unstabilized ROM output (in blue). The relative error in the unstabilized ROM output (22) evaluates to `NaN` ("not a number") on a finite precision arithmetic machine due to overflow caused by the ROM instability. The $M = 17$ mode POD/Galerkin ROM is stabilized by Algorithm 2. Algorithm 1 is not considered for the sake of brevity, and since the superiority of Algorithm 2 has been established in Section 4.1.

4.2.1. Stabilization via Algorithm 2

The $M = 17$ POD/Galerkin ROM for the electrostatically actuated beam benchmark is stabilized using Algorithm 2. The four unstable eigenvalues of $\mathbf{A}_M$ will be denoted by λ_k^u for $k = 1, \ldots, 4$. Similarly to the ISS test case (Section 4.1), two options for the eigenvalue solutions to the optimization problem (14) will be considered:

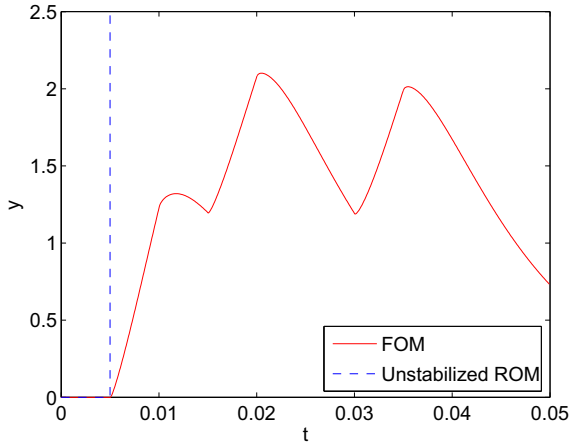


Fig. 7. Outputs for $M = 17$ unstabilized POD/Galerkin ROM vs. FOM output for electrostatically actuated beam problem.

Table 5
Time-integration CPU times for ISS problem: FOM vs. $M = 20$ POD/Galerkin ROM stabilized via Algorithm 2.

Model	Operations	CPU time (sec)
FOM	Time-integration	1.71×10^2
ROM – offline stage	Snapshot collection (FOM time-integration)	1.71×10^2
	Loading of matrices/snapshots	6.99×10^{-2}
	POD	6.20
	Projection	8.18×10^{-3}
	Optimization*	2.28×10^1
ROM – online stage	Time-Integration	3.77

*Optimization times reported are means of the time required to solve (14) with real eigenvalues and the time required to solve (14) with complex-conjugate eigenvalues.

- *Option 1:* Solve for $\lambda_i^u \in \mathbb{R}$ subject to the constraint that $\lambda_i^u < 0$ for $i = 1, \ldots, 4$.
- *Option 2:* Solve for $\lambda_1^{ur}, \lambda_1^{uc}, \lambda_2^{ur}, \lambda_2^{uc} \in \mathbb{R}$ subject to the constraint that $\lambda_1^{ur}, \lambda_2^{ur}, < 0$ and set $\lambda_1^u = \lambda_1^{ur} + i\lambda_1^{uc}$, $\lambda_2^u = \lambda_1^{ur} - i\lambda_1^{uc}$, $\lambda_3^u = \lambda_2^{ur} + i\lambda_2^{uc}$, $\lambda_4^u = \lambda_3^{ur} - i\lambda_3^{uc}$ (that is, λ_3^u is taken to be the complex-conjugate of $\lambda_2^u : \lambda_3^u = \bar{\lambda_2^u}$).

Option 2 is more general than Option 1 and has fewer inequality constraints; however, Option 1 may be more consistent with the system dynamics, as the unstable eigenvalues of **A** are all real. As before, the `fmincon` function in the MATLAB optimization toolbox will be used to solve the optimization problem (14), with the `Algorithm` option set to `interior-point` and an initial guess of -1 for all four variables optimized over in (14). For the functional form of the eigenvalues assumed in Option 1, the algorithm converges in 60 optimization iterations, and requires 64 function evaluations. For the functional form of the eigenvalues assumed in Option 2, which has less constraints than Option 1, fewer optimization iterations and function evaluations are required to achieve convergence: 31 optimization iterations, and 32 function evaluations. Some key information about the convergence of the optimization algorithm for both of these options is summarized in Table 6,

Table 6
Performance of `fmincon` interior point method for Algorithm 2 applied to electrostatically actuated beam problem.

	Algorithm 2 with Option 1 (real eigenvalues)	Algorithm 2 with Option 1 (real eigenvalues)
# Upper bound constraints	4	2
# Optimization iterations	60	31
# Function evaluations	64	32
First-order optimality at convergence ($\|\nabla L\|$)	2.27×10^{-7}	8.43×10^{-7}

Table 7
Original (unstable) eigenvalues of $\mathbf{A}_M$ for $M = 17$ mode POD/Galerkin ROM and new stable eigenvalues computed using Algorithm 2 (electrostatically actuated beam problem).

	Original unstable $\mathbf{A}_M$	Algorithm 2 with Option 1	Algorithm 2 with Option 2
λ_1^u	1.61×10^4	-6.88×10^5	$-1.16 \times 10^5 - 2.25 \times 10^4 i$
λ_2^u	4.90×10^1	-3.54×10^2	$-1.16 \times 10^5 + 2.25 \times 10^4 i$
λ_3^u	1.27×10^1	-1.97×10^4	$-3.32 \times 10^3 - 1.81 \times 10^2 i$
λ_4^u	5.20×10^{-2}	-1.40×10^4	$-3.32 \times 10^2 + 1.81 \times 10^2 i$

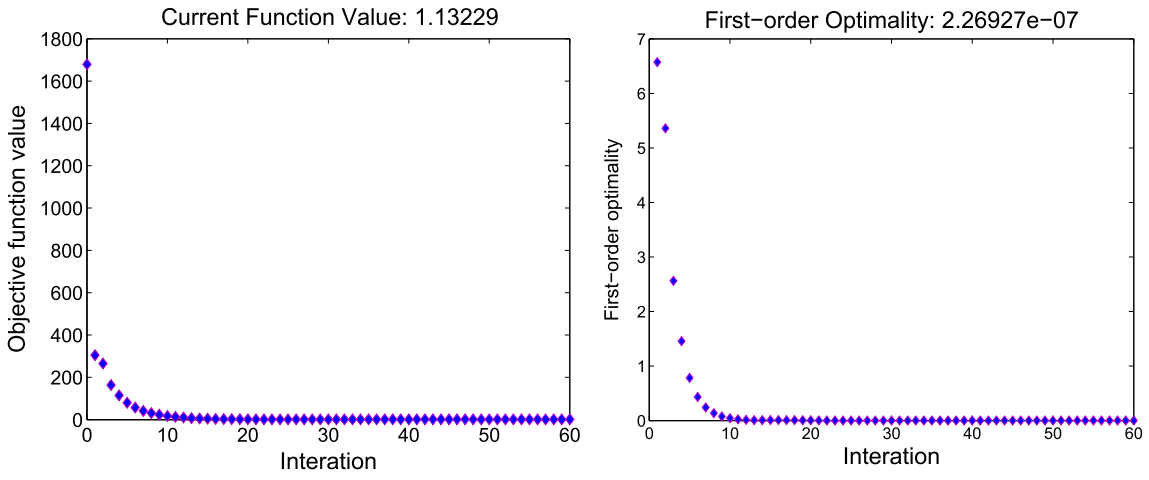
Table 8
Relative errors in $M = 17$ POD/Galerkin ROM for electrostatically actuated beam problem stabilized via Algorithm 2.

ROM	$\mathcal{E}_{rel}$
Unstabilized	`NaN`
ROM stabilized via Algorithm 2 with Option 1 (real eigenvalues)	1.94×10^{-2}
ROM stabilized via Algorithm 2 with Option 2 (complex-conjugate eigenvalues)	2.02×10^{-2}

Table 9
Time-integration CPU times for electrostatically actuated beam problem: FOM vs. $M = 17$ POD/Galerkin ROM stabilized via Algorithm 2.

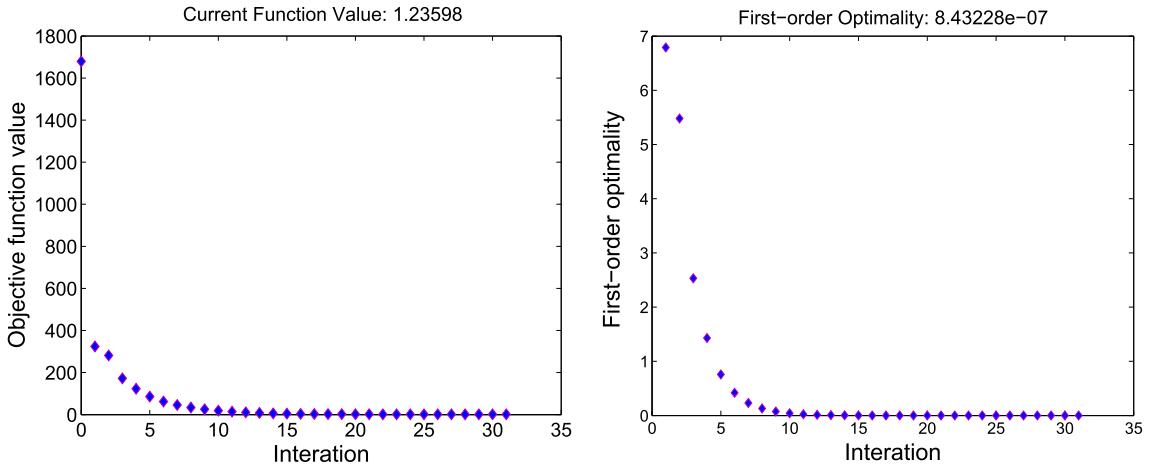
Model	Operations	CPU time (sec)
FOM	Time-integration	7.10×10^4
ROM – offline stage	Snapshot collection (FOM time-integration)	7.10×10^4
	Loading of matrices/snapshots	5.17
	POD	1.09×10^1
	Projection	2.55×10^1
	Optimization*	8.79×10^1
ROM – online stage	Time-Integration	6.78

*Optimization times reported are means of the time required to solve (14) with real eigenvalues and the time required to solve (14) with complex-conjugate eigenvalues.



(a) Function value (b) First order optimality ($|\nabla L|$)

Fig. 8. Performance of interior point algorithm for Algorithm 2 with Option 1 (real eigenvalues) as a function of iteration number (electrostatically actuated beam problem).



(a) Function value (b) First order optimality ($|\nabla L|$)

Fig. 9. Performance of interior point algorithm for Algorithm 2 with Option 2 (complex-conjugate eigenvalues) as a function of iteration number (electrostatically actuated beam problem).

and Figs. 8 and 9. For both options, the optimality conditions are satisfied to the specified tolerance at the value of the optimal solution.

Similarly to the ISS problem, Appendix A.2 (Tables 12,13) gives some additional performance results of the `fmincon` interior point method for Algorithm 2 for different (larger) values of M. ROMs with larger basis sizes possess in general more unstable eigenvalues, and more optimization iterations are required to obtain the solution of the optimization problem (14) using the interior point method.

The solutions obtained by Algorithm 2 with both Option 1 and Option 2 are given in Table 7, compared with the values of the original unstable eigenvalues of $\mathbf{A}_M$. As for the ISS benchmark (Section 4.1), the eigenvalues computed by the optimization algorithm with Option 1 are different in their numerical values from those computed by the optimization algorithm with Option 2. This suggests that the optimization function (14) for this problem has multiple local minimizers/minima.

Table 8 gives the error in the ROM algorithm relative to the FOM output for an $M = 20$ POD/Galerkin ROM stabilized via Algorithm 2 with Option 1 and Option 2. For both options, the relative error in the stabilized ROM output is approximately 2%.

Finally, Fig. 10 shows the output computed from ROMs stabilized using Algorithm 2. There is good agreement between the FOM output and stabilized ROM outputs.

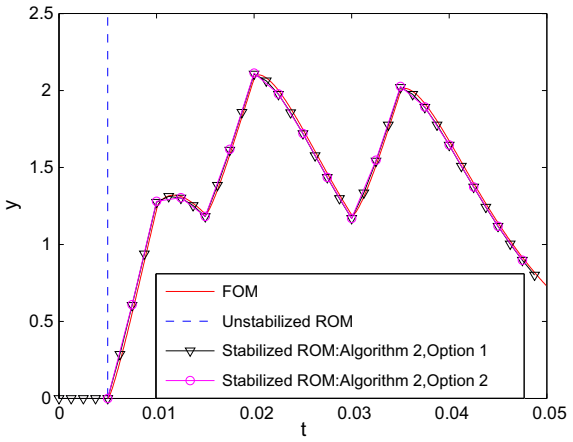


Fig. 10. Outputs for $M = 17$ POD/Galerkin ROMs stabilized via Algorithm 2 vs. FOM output for electrostatically actuated beam problem.

Table 9 summarizes some CPU times for the electrostatically-actuated beam problem: the CPU times for the FOM, as well as the CPU times for the offline and online stages of the $M = 17$ POD/Galerkin electrostatically-actuated beam ROM. The results in this table reveal that the online stage of the model reduction, the stage relevant to real-time calculations involving the ROM, took only 6.78 seconds, compared to 7.10×10^4 seconds for the time-integration stage of the FOM. To offset the total preprocess time of the ROM (the time required to run the FOM to collect snapshots, calculate the POD basis, perform the Galerkin projection, and solve the optimization problem (14)), the ROM would need to be run approximately 1×10^4 times. This large number of online ROM runs required to offset the offline ROM cost is due to the large CPU time associated with the FOM run for this large dense problem. As for the ISS problem, the optimization step of the model reduction does not contribute significantly to the CPU time of the offline stage of the ROM, taking just 1.5 minutes.

5. Conclusions

This paper presents a new approach for stabilizing unstable reduced order models for LTI systems through an *a posteriori* post-processing step applied to the algebraic ROM system. This stabilization step consists of a reassignment of the eigenvalues of the ROM system matrix. First, it is shown how the system's eigenvalues can be modified by adding to the system a linear control term, and solving for the control matrix using full state feedback (a.k.a. pole placement) algorithms from control theory. This approach will yield a stable ROM provided the ROM system's unstable eigenvalues are controllable and observable; however, although the stabilized ROM will be stable, it may not be accurate. To ensure accuracy in the stabilized ROM, a second algorithm is developed, in which the eigenvalues of the stabilized ROM system are computed by solving a constrained nonlinear least-squares optimization problem in which the error in the ROM output is minimized. This problem is small ($< \mathcal{O}(M)$, where M is the number of dofs in the ROM), and therefore computationally inexpensive to solve using standard optimization algorithms. The second stabilization algorithm is the primary contribution of this paper, but both algorithms are presented and evaluated, as the first algorithm led to the formulation of the second. The ROM stabilization approaches developed herein are applicable to ROMs constructed using *any* choice of reduced basis for *any* application. The proposed algorithms are evaluated on two benchmarks: the international space station (ISS) problem and the electrostatically actuated beam problem. Numerical tests reveal that the second algorithm effectively stabilizes an unstable ROM, delivering a modified ROM that is both stable as well as accurate. Extensions of the new method to nonlinear problems and predictive applications, including a study of the robustness of the ROM with respect to parameter changes, will be the subject of future work. For nonlinear problems with stable fixed points and/or limit cycle solutions (e.g., the classical fluid mechanics problem involving flow around a cylinder), a natural extension of the algorithm would involve: (1) determining the stable fixed points of the system, (2) linearizing the system around these points, and (3) using the algorithms developed in this paper to stabilize the linearized system.

Acknowledgments

This research was funded by Sandia National Laboratories Laboratory Directed Research and Development (LDRD) program. Sandia 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-AC04-94AL85000.

The authors would like to thank Prof. Lou Cattafesta at Florida State University and Prof. Karen Willcox at the Massachusetts Institute of Technology for useful discussions that lead to some of the ideas presented in this article. The authors would also like to thank four anonymous reviewers whose insightful comments helped to improve the said work.

Appendix A

A.1. Jacobian of objective function in (14)

In this section, the analytic expression for the Jacobian of the objective function in the optimization problem (14) for the specific case when $\mathbf{u}(t) = \mathbf{0}$, $\mathbf{y} \in \mathbb{R}$ (there is a single output of interest), and $\lambda_i^u \in \mathbb{R}$ is derived. In many cases, it is possible to derive analytically the Jacobian of the objective function in (14) without these simplified assumptions, but this derivation will be problem-dependent (i.e., it will depend on the specific forcing $\mathbf{u}(t)$). Let $y^k \equiv \mathbf{y}^k \in \mathbb{R}$ and $y_M^k \equiv \mathbf{y}_M^k \in \mathbb{R}$. If $\mathbf{u}(t) = \mathbf{0}$, the objective function in (14) evaluates to:

$$f = ||\mathbf{F}||_2^2, \tag{32}$$

where

$$\mathbf{F} \equiv \begin{pmatrix} \mathbf{CS}\exp(\mathbf{D}t_1)\mathbf{S}^{-1}\mathbf{x}(0) - y^1 \\ \mathbf{CS}\exp(\mathbf{D}t_2)\mathbf{S}^{-1}\mathbf{x}(0) - y^2 \\ \vdots \\ \mathbf{CS}\exp(\mathbf{D}t_K)\mathbf{S}^{-1}\mathbf{x}(0) - y^K \end{pmatrix} \in \mathbb{R}^K. \tag{33}$$

Let $\mathbf{J}$ denote the Jacobian of f (32). The reader can verify that

$$\mathbf{J} = 2\mathbf{J}_F^T\mathbf{F} \in \mathbb{R}^L \tag{34}$$

where the (k,l)th entry of $\mathbf{J}_F$ is given by

$$J_F(k,l) = t_k\mathbf{CS}\exp(\hat{\mathbf{D}}_l t_k)\mathbf{S}^{-1}\mathbf{x}(0), \tag{35}$$

for $k = 1,..,K$ and $l = 1,\ldots,L$. In Eq. (35),

$$\hat{\mathbf{D}}_l \equiv \begin{pmatrix} 0 & & & & & & \\ & \ddots & & & & & \\ & & 0 & & & & \\ & & & \lambda_l^u & & & \\ & & & & 0 & & \\ & & & & & \ddots & \\ & & & & & & 0 \end{pmatrix} \in \mathbb{R}^{M\times M}, \tag{36}$$

that is, $\hat{\mathbf{D}}_l$ is a matrix with a single entry of λ_l^u in the position $(\hat{l},\hat{l})$, where $\hat{l}$ is the position of the l^{th} reassigned eigenvalue in the original matrix $\mathbf{D}$.

Table 10
Performance of `fmincon` interior point method for Algorithm 2 applied to ISS problem as a function of M (real eigenvalues).

M	20	40	60
# Unstable eigenvalues	4	5	6
# Upper bound constraints	4	5	6
# Optimization iterations	29	58	45
# Function evaluations	30	59	46
First-order optimality at convergence ($\|\nabla L\|$)	4.00×10^{-7}	9.88×10^{-7}	2.46×10^{-7}

Table 11
Performance of `fmincon` interior point method for Algorithm 2 applied to ISS problem as a function of M (complex-conjugate eigenvalues).

M	20	40	60
# Unstable eigenvalues	4	5	6
# Upper bound constraints	3	3	3
# Optimization iterations	27	50	62
# Function evaluations	30	52	64
First-order optimality at convergence ($\|\nabla L\|$)	5.51×10^{-7}	2.46×10^{-7}	3.94×10^{-7}

Table 12
Performance of `fmincon` interior point method for Algorithm 2 applied to electrostatically actuated beam problem as a function of M (real eigenvalues).

M	17	34	51
# Unstable eigenvalues	4	10	14
# Upper bound constraints	4	10	14
# Optimization iterations	60	78	96
# Function evaluations	64	82	100
First-order optimality at convergence ($\|\nabla L\|$)	2.27×10^{-7}	4.61×10^{-7}	2.13×10^{-7}

Table 13
Performance of `fmincon` interior point method for Algorithm 2 applied to electrostatically actuated beam problem as a function of M (complex-conjugate eigenvalues).

M	17	34	51
# Unstable eigenvalues	4	10	14
# Upper bound constraints	2	5	7
# Optimization iterations	31	35	78
# Function evaluations	32	36	79
First-order optimality at convergence ($\|\nabla L\|$)	8.43×10^{-7}	6.20×10^{-6}	1.08×10^{-7}

A.2. Additional performance results for Algorithm 2

The following tables give some additional performance results (the number of unstable eigenvalues, the number of upper bound constraints, the number of optimization iterations, the number of function evaluations, and the first order optimality at convergence) for Algorithm 2 applied to the ISS and electrostatically actuated beam problems considered in Sections 4.1 and 4.2 respectively. These results enable one to study how these quantities change as M, the reduced basis size, is increased. The performance of the interior point method depends more on the number of dofs in the optimization problem (14), rather than the basis size M directly. For the problems considered herein, as M is increased, in general so does the number of unstable eigenvalues of the ROM.

References

[1] G. Serre, P. Lafon, X. Gloerfelt, C. Bailly, Reliable reduced-order models for time-dependent linearized Euler equations, J. Comput. Phys. 231 (15) (2012) 5176–5194.
[2] D. Amsallem, C. Farhat, Stabilization of projection-based reduced order models, Int. J. Numer. Methods Eng. 91 (4) (2012) 358–377.
[3] M.F. Barone, I. Kalashnikova, D.J. Segalman, H. Thornquist, Stable Galerkin reduced order models for linearized compressible flow, J. Comput. Phys. 288 (2009) 1932–1946.
[4] I. Kalashnikova, M.F. Barone, On the stability and convergence of a Galerkin reduced order model (ROM) for compressible flow with solid wall and far-field boundary treatment, Int. J. Numer. Methosa Eng. 83 (2010) 1345–1375.
[5] S. Gugercin, A.C. Antoulas, A survey of model reduction by balanced truncation and some new results, Int. J. Control 77 (8) (2004) 748–766.
[6] K.J. Astrom, R.M. Murray, Feedback Systems: An Introduction for Scientists and Engineers, Princeton University Press, 2008.
[7] K. Zhou, Robust and Optimal Control, Prentice Hall, 1996.
[8] J.L. Lumley, Stochastic Tools in Turbulence, Academic Press, New York, 1971.
[9] P. Holmes, J.L. Lumley, G. Berkooz, Turbulence, Coherent Structures, Dynamical Systems and Symmetry, Cambridge University Press, 1996.
[10] T. Bui-Thanh, K. Willcox, O. Ghattas, B. van Bloemen Waanders, Goal-oriented, model constrained optimization for reduction of large-scale systems, J. Comput. Phys. 224 (2007) 880–896.
[11] C.W. Rowley, Model reduction for fluids using balanced proper orthogonal decomposition, Int. J. Bifurcation Chaos 15 (3) (2005) 997–1013.
[12] C.W. Rowley, T. Colonius, R.M. Murray, Model reduction for compressible flows using POD and Galerkin projection, Physica D 189 (2004) 115–129.
[13] L. Sirovich, Turbulence and the dynamics of coherent structures, part III: dynamics and scaling, Q. Appl. Math. 45 (3) (1987) 583–590.
[14] N. Aubry, P. Holmes, J. Lumley, E. Stone, The dynamics of coherent structures in the wall region of a turbulent boundary layer, J. Fluid Mech. 192 (1988) 115–173.
[15] K. Veroy, A.T. Patera, Certified real-time solution of the parametrized steady incompressible Navier–Stokes equations: rigorous reduced-bases a posteriori error bounds, J. Numer. Methods Fluids 47 (2005) 773–788.
[16] B. Moore, Principal component analysis in linear systems: controllability, observability, and model reduction, IEEE Trans. Autom. Control 26 (1) (1981).
[17] I. Kalashnikova, S. Arunajatesan, A stable Galerkin reduced order model for compressible flow, in: 10th World Congress on Computational Mechanics (WCCM), WCCM-2012-19407, Sao Paulo, Brazil, 2012.
[18] D. Amsallem, C. Farhat, On the stability of projection-based linear reduced-order models: descriptor vs. non-descriptor forms, in: Reduced order methods for modeling and computational reduction, vol. 8, Springer MS&A series, 2013, accepted for publication.
[19] K. Willcox, J. Peraire, Balanced model reduction via the proper orthogonal decomposition, AIAA J. 40 (11) (2002) 2323–2330.
[20] Z. Wang, I. Akhtar, J. Borggaard, Traian Iliescu, Proper orthogonal decomposition closure models for turbulent flows: A numerical comparison, Comput. Methods Appl. Mech. Eng. (2012) 237–240.
[21] A.C. Antoulas, D.C. Sorensen, S. Gugercin, A survey of model reduction methods for large-scale systems, Contemp. Math. 280 (2001) 193–219.
[22] B.N. Bond, L. Daniel, Guaranteed stable projection-based model reduction for indefinite and unstable linear systems, in: Proceedings of the 2008 IEEE/ACM International Conference on Computer-Aided Design, 2008.
[23] M.J. Balajewicz, E.H. Dowell, B.R. Noack, Low-dimensional modeling of high-Reynolds-number shear flows incorporating constraints from the Navier-Stokes equation, J. Fluid Mech. 729 (2013) 285–308.
[24] Y. Chahlaoui, P. Van Dooren, Benchmark examples for model reduction of linear time invariant systems, Mar. 2013.<http://www.icm.tu-bs.de/NICONET/benchmodred.html>.

[25] The MathWorks Inc., Control Systems Toolbox User's Guide, 1992–1998.
[26] J. Lienemann, E.B. Rudnyi, J.G. Korvink, MST MEMS model order reduction: requirements and benchmarks, Linear Algebra Appl. 415 (2–3) (2006) 469–498.
[27] W. Weaver Jr., S.P. Timoshenko, D.H. Young, Vibration Problems in Engineering, fifth ed., Wiley, 1990.
[28] Oberwolfach benchmark collection, 2005. <http://portal.uni-freiburg.de/imteksimulation/downloads/benchmark/>.
[29] The MathWorks Inc., Optimization Toolbox User's Guide, 1990–2008.
[30] J. Nocedal, S.J. Wright, Numerical Optimization, Springer, 1999.
[31] C.A. Beattie, S. Gugercin. Krylov-based model reduction of second-order systems with proportional damping. in: Proceedings of the 44th IEEE Conference on Decision and Control, and the European Control Conference 2005, Seville, Spain, 2005, pp. 2278–2283.
[32] G. Rozza, Reduced basis approximation and error bounds for potential flows in parametrized geometries, Commun. Comput. Phys. 9 (1) (2011) 1–48.
[33] G. Chen, Stability of nonlinear systems, Encyclopedia of RF and Microwave Engineering, Wiley, NY, 2004 (pp. 4881–4896).
[34] J. Kautsky, N.K. Nichols, D. Van Dooren, Robust pole assignment in linear state feedback, Int. J. Control 41 (1985) 1129–1155.
[35] S. Sirisup, G.E. Karniadakis, A spectral viscosity method for correcting the long-term behavior of POD models, J. Comput. Phys. 194 (2004) 92–116.
[36] I. Akhtar, A.H. Nayfeh, C.J. Ribbens, On the stability and extension of reduced-order Galerkin models in incompressible flows: a numerical study of vortex shedding, Theor. Comput. Fluid Dyn. 23 (3) (2009) 213–237.
[37] B.R. Noack, K. Afanasiev, M. Morzynski, G. Tadmor, F. Thiele, A hierarchy of low-dimensional models of the transient and post-transient cylinder wake, J. Fluid Mech. 497 (2003) 335–363.
[38] M. Bergmann, C.-H. Bruneau, A. Iollo, Enablers for robust POD models, J. Comput. Phys. 228 (2) (2009) 516–538.
[39] M. Couplet, C. Basdevant, P. Sagaut, Calibrated reduced-order POD-Galerkin systems for fluid modelling, J. Comput. Phys. 207 (2005) 192–220.
[40] S. Sirisup, D. Xiu, G. Karniadakis, X. Kevrekidis, Equation-free/Galerkin-free POD-assisted computation of incompressible flows, J. Comput. Phys. 207 (2005) 568–587.
[41] F. Terragni, E. Valero, J.M. Vega, Local POD plus Galerkin projection in the unsteady lid-driven cavity problem, SIAM J. Sci. Comput. 33 (6) (2011) 3538–3561.
[42] F. Chinesta, P. Ladaveze, E. Cueto, A short review on model order reduction based on proper generalized decomposition, Arch. Comput. Methods Eng. 18 (4) (2011) 395–404.