SIAM J. Applied Dynamical Systems
Vol. 16, No. 3, pp. 1563–1586

Feedback Control for Systems with Uncertain Parameters Using Online-Adaptive Reduced Models*

Boris Kramer†, Benjamin Peherstorfer‡, and Karen Willcox†

Abstract. We consider control and stabilization for large-scale dynamical systems with uncertain, time-varying parameters. The time-critical task of controlling a dynamical system poses major challenges: using large-scale models is prohibitive, and accurately inferring parameters can be expensive, too. We address both problems by proposing an offline-online strategy for controlling systems with time-varying parameters. During the offline phase, we use a high-fidelity model to compute a library of optimal feedback controller gains over a sampled set of parameter values. Then, during the online phase, in which the uncertain parameter changes over time, we learn a reduced-order model from system data. The learned reduced-order model is employed within an optimization routine to update the feedback control throughout the online phase. Since the system data naturally reflects the uncertain parameter, the data-driven updating of the controller gains is achieved without an explicit parameter estimation step. We consider two numerical test problems in the form of partial differential equations: a convection-diffusion system, and a model for flow through a porous medium. We demonstrate on those models that the proposed method successfully stabilizes the system model in the presence of process noise.

Key words. feedback control, time-varying parameters, dynamical systems, data-driven reduced models, model reduction, online adaptive model reduction, low-rank approximations

AMS subject classifications. 34H02, 37E02, 93C15

DOI. 10.1137/16M1088958

1. Introduction.

We consider stabilization and control of large-scale dynamical systems with uncertain, time-varying parameters, which bear significant challenges for control engineers. Mathematical models for industrial systems are often parameter-dependent, and the parameters in turn time-varying. Changes in parameters, such as boundary conditions, the viscosity of a fluid, material coefficients in solids, etc., alter the system responses to otherwise similar inputs and external disturbances. More specifically, such systems show various degrees of sensitivity with respect to their governing parameters. A parametrized change in a system can occur suddenly in a discontinuous fashion, e.g., in the damage of an aircraft

*Received by the editors August 10, 2016; accepted for publication (in revised form) by E. Kostelich April 3, 2017; published electronically August 17, 2017.
http://www.siam.org/journals/siads/16-3/M108895.html
Funding: This work was supported by the DARPA EQUiPS program award number UTA15-001067 (program manager F. Fahroo) and by the United States Department of Energy Office of Advanced Scientific Computing Research (ASCR) grants DE-FG02-08ER2585 and DE-SC0009297, as part of the DiaMonD Multifaceted Mathematics Integrated Capability Center (program manager S. Lee).
†Massachusetts Institute of Technology, Cambridge, MA (bokramer@mit.edu, http://web.mit.edu/bokramer/www/, kwillcox@mit.edu, http://kiwi.mit.edu/).
‡University of Wisconsin-Madison (peherstorfer@wisc.edu, https://pehersto.engr.wisc.edu/).

wing. In contrast, parameters can also change slowly and gradually, e.g., in fatigue scenarios of mechanical structures.

When the parameters are critical for stability or performance of the plant, appropriate control action has to be taken to ensure that effects on the dynamics due to changes of the underlying parameters are properly controlled. Model-based feedback control provides an elegant and mathematically sound way to design a controller. However, there are several challenges that have to be addressed in order to design a model-based feedback controller for large-scale, parameter-dependent dynamical systems. First, in industrial practice, assembling and extracting parameter-dependent system matrices is a delicate task, due to the complex structure of legacy codes; doing so for a real-time control application that needs repeated access to those matrices is even more challenging. Second, if partial differential equation (PDE) models are available, their discretizations are large-scale, rendering them infeasible for optimization and control in time-critical applications. Third, it is often difficult and expensive to estimate the underlying parameters accurately during operation of the plant.

We address these challenges by proposing an offline-online strategy that can handle uncertain parameters that change over time. In particular, we build on recent methods in *data-driven* reduced-order modeling [34], to enable reduced-order feedback control of large-scale dynamical systems with uncertain parameters. We use the expensive high-fidelity model only during the offline phase; we learn a reduced-order model (ROM) from system data in the online phase. The learned ROM is employed within a computationally efficient optimization routine to update the feedback control as data is gathered throughout the online phase. Since the system data naturally reflects the uncertain parameter, this data-driven updating of the controller gains is achieved without an explicit parameter estimation step.

This work is related to diverse work in control, numerical linear algebra, and model learning. When a system model itself is not readily available, one can either deviate from model-based control altogether, or estimate system models from data. In this light, the combination of statistical learning theory and control methods opens new pathways for designing efficient controllers. In the early work of [28], a neural network was used during real-time operation, in order to train the control law from data. In the recent work [17], statistical learning ideas are used in a model-free, data-driven framework to compute the controller online. Moreover, the authors in [20] design a dynamic observer (the dual problem to control) in a data-driven setup. Another alternative when models are unavailable is to employ system identification techniques, as used by [29, 19] to estimate the linear time-invariant (LTI) operators of the underlying system. However, rapid changes to the underlying plant might require fast adaptation of the control, whereas learning techniques may need more data to adapt and confidently infer the model.

There is also a significant amount of work on learning (projection-based) reduced models directly from snapshot data, rather than explicitly performing projection with the system matrices. The Loewner framework provides a nonintrusive approach for constructing reduced models of LTI systems [32, 21]. The reduced model is extracted directly from transfer function values, without requiring the system matrices of the full LTI system. Vector fitting [18, 15] fits rational interpolants to frequency response data of LTI systems. The eigensystem realization algorithm is another example of a system identification approach for LTI systems [25, 22, 13, 24]. Dynamic mode decomposition (DMD) learns a linear reduced dynamical system that best fits a snapshot trajectory in the L^2 norm [41, 37, 40] and has been extended

to incorporate control actions in [36]. Originally introduced for analyzing the behavior of dynamical systems, DMD models have been shown to have a predictive capability as well [46]. The work [11] uses sparsity-promoting learning techniques to select and fit basis functions of a library to data.

For systems with time-varying but known parameters, gain scheduling approaches were proposed in [5, 48], where controllers are designed for specific solutions of interest, such as desired operating conditions of the plant. To circumvent the problem of large-scale models, the authors in [35] suggest using parametric ROMs as surrogates for the high-fidelity model. Therein, ROMs are generated for the linearized equations at fixed parameter values, and interpolated online for new parameters. In both cases, it is assumed that the governing parameters (i.e., Reynolds number) are accessible in real-time, an assumption we shall *not* make in this work.

The work of Mathelin and co-workers [30, 31] divides the control design problem into an offline and online phase. In the offline stage, high-fidelity feedback laws are computed with varying initial conditions. The authors parametrize the control through the initial conditions. During an online stage, a compressed sensing approach determines the current state of the plant, and the control is obtained by an interpolation of the expensively computed feedback laws. Moreover, the authors in [12, 39] design offline libraries of dynamic regimes for classification of data in an online routine.

This paper differs from these large bodies of work by considering the case of unknown time-varying parameters. Moreover, we do not assume access to a system matrix during operation of the controller, and therefore propose to learn and update a reduced LTI system representation of the system for feedback control.

This paper is organized as follows. In section 2, we state the problem formulation and briefly review the optimal control problem for dynamical systems. We discuss solution approaches for *fixed* parameters based on low-rank methods. In section 3 we detail the proposed method, including the steps necessary for library generation and online detection. Section 4 then shows numerical results for two PDE test problems. Section 5 offers a brief summary and conclusions.

2. Problem formulation and background.

We start by defining the motivating problem for this research in section 2.1. The subsequent sections then introduce the necessary background material: section 2.2 discusses the optimal control problem for a fixed set of parameters, and section 2.3 introduces low-rank solution strategies for the optimal control problem. We complete this section by stating our contributions in section 2.4.

2.1. Problem formulation.

Consider the large-scale dynamical system with time-varying parameters

$$\dot{z}(t;q(t)) = A(q(t))z(t;q(t)) + Bu(t;q(t)), \quad q(0) = q_0 \in \mathbb{R}^d,\ z(0;q_0) = z_0 \in \mathbb{R}^n, \tag{1}$$

$$y(t;q(t)) = Cz(t;q(t)), \tag{2}$$

for all $t > 0$. The system matrix $A(q(t)) \in \mathbb{R}^{n\times n}$ depends on the time-varying parameter $q(t) \in \mathbb{R}^d$, resulting in a time-varying system matrix. The input matrix $B \in \mathbb{R}^{n\times m}$ and the output matrix $C \in \mathbb{R}^{p\times n}$ are considered to be fixed. The controls $u(t;q(t)) \in \mathbb{R}^m$, the

controlled outputs $y(t;q(t)) \in \mathbb{R}^p$, and the states $z(t;q(t)) \in \mathbb{R}^n$ depend on the parameters, as indicated by the $(\cdot;q(t))$ notation. We also refer to system (1)–(2) as the *high-fidelity model*. When system (1)–(2) stems from the spatial discretization of a PDE, the state vector contains the unknowns corresponding to the spatially discretized PDE state variable. Our objective is to find a control $u(t;q(t))$ that minimizes the convex cost

$$J(z,u;q) = \int_0^\infty ||Cz(t;q(t))||_2^2 + ||Ru(t;q(t))||_2^2 \, dt, \tag{3}$$

subject to the dynamic constraints (1). The matrix $0 < R \in \mathbb{R}^{m\times m}$ contains the control weights. The time horizon is chosen infinite, since we assume no information as to when our controller process terminates.

We model the time-dependency in the parameters by a piecewise constant function in time, that is

$$q(t) = q_{\mathcal{T}_i} \quad \text{for} \quad t \in \mathcal{T}_i = [t_{i-1}, t_i], \ i = 1, 2, \ldots.$$

However, it is not known a priori when the parameter changes, so the *switching times* t_i are unknown, and hence have to be detected during online operation of the plant. Therefore, the time interval $\mathcal{T}_i$ has unknown starting and end points. We discuss in section 3.3.2 a technical assumption on the length of the time intervals $\mathcal{T}_i$. Owing to the piecewise continuity of the parameters, the control takes piecewise form

$$u(t;q(t)) = u(t;q_{\mathcal{T}_i}), \quad t \in \mathcal{T}_i.$$

Throughout this paper, we shall use the term *offline* to denote non-time-critical situations, such as the process of control design. In the offline stage, we assume that computational time is not of major concern, so that large-scale simulations/optimization and computationally expensive tasks can be carried out. By *online* stage, we refer to time-critical scenarios, when the plant (modeled by the dynamical system) is under operation and data streamed. These data need to be processed, used, and computed with in a time-critical manner.

Problem 2.1. *Let the system matrix $A(q(t))$ be accessible offline, but not online. Moreover, assume that B, C are stored and available online. For time-varying, piecewise constant parameters $q(t) \equiv q_{\mathcal{T}_i}$ for $t \in \mathcal{T}_i = [t_{i-1}, t_i]$ for $i \in \mathbb{N}$ with the switching times t_i unknown a priori, solve the minimization problem*

$$\begin{aligned} \forall i \in \mathbb{N}: \quad & \min_{z_{\mathcal{T}_i}, u_{\mathcal{T}_i}} \quad J(z_{\mathcal{T}_i}, u_{\mathcal{T}_i}; q_{\mathcal{T}_i}) \\ & s.t. \quad \dot{z}_{\mathcal{T}_i}(t) = A(q_{\mathcal{T}_i}) z_{\mathcal{T}_i}(t) + B u_{\mathcal{T}_i}(t), \end{aligned}$$

where the cost function is given by (3) and the subscripts indicate the state and control in the interval $\mathcal{T}_i = [t_{i-1}, t_i]$.

2.2. Optimal control for dynamical systems with time-invariant parameters. We briefly review the optimal control problem and its solution for time-invariant parameters, which then illustrates the additional challenges imposed by time-varying parameters. For a *time-invariant parameter* q, the problem of controlling (and stabilizing) the state $z(t) = z(t;q)$ of (1) to

a desired target state as $t \to \infty$ *independent of the initial condition* z_0 has been studied extensively. Indeed, a sound mathematical theory for the performance of a feedback control exists [27] for the case of state (or measured) feedback $u(t) = u(z(t))$.

Definition 2.2. *The* linear quadratic regulator (LQR) *control problem for fixed parameter* q *is as follows:*

$$\begin{aligned} \min_{z,u} \quad & J(z,u) \\ s.t. \quad & \dot{z}(t) = A(q)z(t) + Bu(t), \end{aligned}$$

where

$$J(z,u) = \int_0^\infty ||Cz(t)||_2^2 + ||Ru(t)||_2^2 \, dt.$$

The first term under the integral in (3) penalizes the deviation of the controlled output $y(t)$ from zero. The second term penalizes a weighted control action, such that a balance between achieving the goal of driving the controlled output back to zero and the control effort used is found. Therefore, the LQR problem defines a family of controllers, parametrized by the control weights R.

The LQR problem has a well known solution [27, sect. 3.4] in the form of *linear* state feedback

$$u(t) = -K(q)z(t). \tag{4}$$

Here, $K(q)$ denotes the *gain matrix*, containing the *feedback gains* as rows. The feedback gains contain relevant information about the impact of the state on the control action; for instance, feedback gains can be used to optimize sensor and actuator locations [1]. For a fixed parameter q, and under mild assumptions (the pair $A(q), B$ must be stabilizable, see [3, 27]), the gain matrix and control can be computed by solving the algebraic Riccati equation (ARE) for the unique symmetric and positive definite solution Π:

$$A(q)^T\Pi(q) + \Pi(q)A(q) - \Pi(q)BB^T\Pi(q) + C^TC = 0, \tag{5}$$

so that

$$K(q) = R^{-1}B^T\Pi(q). \tag{6}$$

Throughout this paper we assume that the Riccati equation (5) has a unique positive definite solution for all parameter values q.

2.3. Low-rank methods to solve control problem. For a *fixed parameter* q, significant advances have been made to solve the ARE (5) in a large-scale setting; see for instance the survey in [8]. When n is large, storing an $n \times n$ matrix is computationally infeasible—even more so when the solution is needed in an online fashion—and exploiting additional structure is inevitable. As it turns out, methods that devise a low-rank factorization

$$\Pi(q) = W(q)W(q)^T, \quad W(q) \in \mathbb{R}^{n\times r}, \tag{7}$$

have been successful [42, 7, 16, 6, 43]. The rank of a matrix is defined by the maximum number of linearly independent rows or columns. Notably, one only has to store the matrix $W(q)$ of size $n \times r$, where $r \ll n$, and computing $K(q) = R^{-1}B^T W(q)[W(q)^T]$ does not require processing a square matrix anymore. One way to arrive at a low-rank approximation of $\Pi(q)$ is by considering a Galerkin projection framework. In [42], the authors showed that projection-based methods provide a viable path to solving (5) efficiently. More generally, physics-based ROMs derived via Galerkin projection have provided viable surrogates used in real-time estimation and control [26, 4, 9, 38, 2, 33, 45].

For a fixed parameter q, we thus compute low-dimensional solutions to AREs

$$\hat{A}(q)^T\hat{\Pi}(q) + \hat{\Pi}(q)\hat{A}(q) - \hat{\Pi}(q)\hat{B}\hat{B}^T\hat{\Pi}(q) + \hat{C}^T\hat{C} = 0 \tag{8}$$

with a projection matrix $V \in \mathbb{R}^{n\times r}$ and $\hat{A}(q) \in \mathbb{R}^{r\times r}$, $\hat{B} \in \mathbb{R}^{r\times m}$, and $\hat{C} \in \mathbb{R}^{p\times r}$. An approximation of the feedback matrix is then obtained via

$$\hat{K}(q) = R^{-1}\hat{B}^T\hat{\Pi}(q),$$

which then yields the suboptimal feedback controller for the high-fidelity model

$$u_r(t;q) = -\hat{K}(q)Vz(t;q). \tag{9}$$

2.4. Challenges and contribution. The challenge of Problem 2.1 lies in the time-varying parametric dependence of the control, requiring the computationally expensive gain computation (5)–(6) online. Since an exact solution is computationally infeasible, one has to approximate (or update) $K(q(t))$ online by only having system data available, but without knowing $q(t)$ explicitly. This is further complicated by the fact that $A(q(t))$ is also unavailable online, since it is too expensive to evaluate.

To solve this problem, we propose a new two-stage approach based on the LQR theory introduced in section 2.2. In the offline stage, the ARE (5) is solved with high accuracy for some pre-selected parameter values, and the resulting feedback gains are stored in a library. In the online stage (i.e., during operation of the plant), we design a mapping from the state (or partial state for computational efficiency) to an index of the library elements. This mapping provides a rapid way to classify a change in the underlying parameters. Based on the outcome of this classification step, we select a feedback gain from the library, and initiate an operator inference using real-time state information from the plant. Once a ROM is learned, we can recompute the feedback gain, leading to suboptimal controllers for the high-dimensional system. Based on LQR theory, the controllers are of full-state feedback type.

The proposed method exploits the low-rank structure of the Riccati solution $\Pi(q(t))$ in a projection-based framework, but does not require knowing the parameter $q(t)$; instead, the method uses data to update/infer a reduced-order system representation during online operation of the plant.

Remark 2.3. In an ideal situation where 1) the parameter function $q(t)$ is known for all times, and 2) the system matrices are available online and of moderate dimension (where

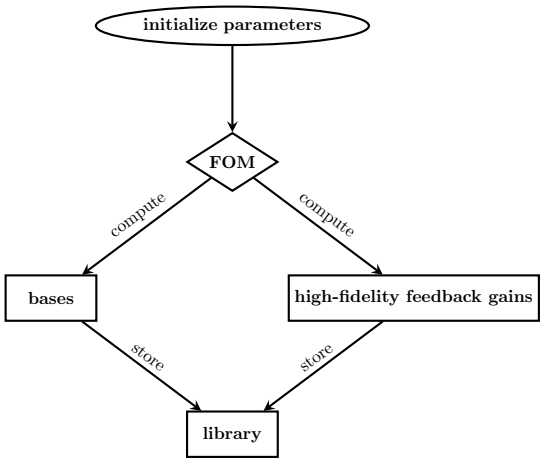
ARE can be solved rather cheaply online), one can solve the optimal control, for instance with projection-based intrusive ROMs. We address the situation where both of these assumptions fail, and where the incorporation of real system data enables us to compute model-based feedback controllers.

3. Closed-loop control: Combining model libraries and model learning. Our proposed method learns a state-space representation for control and combines this with a library of precomputed feedback gains, to arrive at a control strategy for systems with time-varying and unknown parameters. We outline the methodology in Figure 1, and give details of the method below.

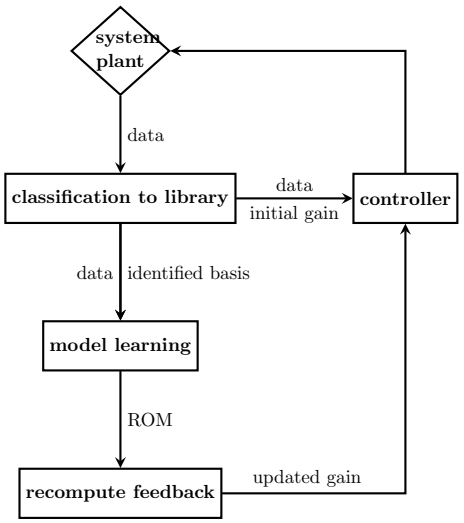
Section 3.1 briefly introduces the reduced-order modeling framework, on which we build in the following subsections. The offline stage is formulated in section 3.2, including the computation of the library of feedback gains and low-dimensional bases. In this stage, we work with the expensive high-fidelity model.

The online part of our proposed method avoids expensive computations and is described in section 3.3. The method detects changes in the parameters, and acts on the information by immediately switching to a feedback gain from the library, and by learning a new model to update the gains. The online section ends with a statement of the complete algorithm.

We subsequently ease our notation by using $q = q(t)$ except in places where it is necessary to emphasize the time-dependence. However, the reader should note that the time-dependency



(a) Offline stage: For an initial set of parameters, the high-fidelity model is used for the expensive computation of bases needed for model learning and parameter detection, as well as for the computation of feedback gains. The high-fidelity information is then stored in the library.



(b) Online stage: System data is classified by mapping the data to an index in the library of bases; see section 3.3.1. Then, model learning is initiated, and a feedback gain initialized. Once the learned model is available, feedback is recomputed, updated, and applied to the system.

Figure 1. *Outline of the proposed offline-online method.*

of the parameters makes the control problem significantly more challenging than considering only fixed parameters.

3.1. Reduced-order models to represent dynamics. In the offline and online stages, we build on results from reduced-order modeling of large-scale systems; see Antoulas' book [3]. The key observation in reduced-order modeling is that the state of the dynamical system (1)–(2) can often be represented with a basis of drastically reduced dimensions, i.e.,

$$z(t;q) \approx V(q)\hat{z}(t;q), \qquad V(q) \in \mathbb{R}^{n\times r}, \quad r \ll n,$$

where $V(q)$ contains basis vectors for a low-dimensional, accurate representation of the dynamics of (1)–(2). Inserting this approximation into system (1)–(2), and multiplying by $V(q)^T$ from the left leads to a ROM of similar structure

$$\dot{\hat{z}}(t;q) = \hat{A}(q)\hat{z}(t;q) + \hat{B}u(t), \quad q(0) = q_0 \in \mathbb{R}^d, \ \hat{z}(0;q_0) = \hat{z}_0 \in \mathbb{R}^r, \tag{10}$$

$$y(t;q) = Cz(t;q), \tag{11}$$

where $\hat{A}(q) = V(q)^T A(q) V(q)$ is the reduced-order system matrix and $\hat{B}(q) = V(q)^T B$. The states of the ROM evolve in the low-dimensional subspace $V(q)$. Subsequently, we compute a low-dimensional basis both for learning a ROM, as well as for detection of changes in the time-varying parameter.

3.2. Offline: High fidelity library generation. In the offline stage, a set of M parameters $\{q_1, \ldots, q_M\}$ is chosen. The aim is to sample over parameters that are representative of the conditions one might expect during operation of the system. This can be done by expert opinion, a greedy-type sampling approach, or heuristic considerations. The offline stage requires three steps.

First, for each of the selected parameters, we use the high-fidelity matrices $A(q_i), B, C$ to compute a high-fidelity LQR feedback matrix $K(q_i)$ from (6). This requires solving the ARE (5). We then store the resulting feedback matrices in a *library*—a memory location (e.g., array) that is easily and quickly accessible during operation of the system. Having the feedback matrices in the library allows us to quickly react to changes in the parameters in the online phase.

In the second offline step, we compute *detection bases* $V_D(q_i), i = 1, \ldots, M$, which provide a low-dimensional approximation of the system state and which are later needed for detection of parametric changes. Through a projection of the system measurements onto the detection basis in the online phase, we are able to infer the parameter q_i that generated the data. In this work, we use the method of proper orthogonal decomposition (POD) (see, e.g., [47] for a detailed description of POD). The POD method requires the generation of $S \in \mathbb{N}$ snapshots of the dynamical system (1) through initial excitation, or by using a time-dependent input function. The snapshots are stored in the matrix

$$Z(q_i) = [z(t_1;q_i), \ z(t_2;q_i), \ldots, z(t_S;q_i)] \in \mathbb{R}^{n\times S}, \tag{12}$$

which is the starting point to extract a low-dimensional basis to best represent the data in an L_2 sense. For large-scale systems, one likely has $n \gg S$, so that the method of snapshots

[44] provides an efficient way of computing the basis. Thus, compute the singular value decomposition

$$Z(q_i)^T Z(q_i) = \Psi \Sigma \Psi^T \quad \in \mathbb{R}^{S\times S}, \tag{13}$$

let $\Sigma_{\tilde{r}_i}$ be the leading $\tilde{r}_i \times \tilde{r}_i$ submatrix of Σ, and let $\Psi_{\tilde{r}_i}$ denote the matrix containing the leading $\tilde{r}_i$ columns of Ψ. We then compute the *proper orthogonal basis*, and store it in the library for later use as the detection basis

$$V_D(q_i) = Z(q_i)\Psi_{\tilde{r}_i}\Sigma_{\tilde{r}_i}^{1/2}. \tag{14}$$

A decision on how many POD basis vectors to keep is often based on the decay of the singular values $\sigma_j = \Sigma_{jj}$ of the snapshot matrix, that is, by setting an energy threshold $\epsilon \leq \sum_{j=1}^{\tilde{r}_i} \sigma_j^2 \backslash \sum_{j=1}^{S} \sigma_j^2$.

In the third offline step, we compute an r_i-dimensional *learning basis* $V_L(q_i), i = 1, \ldots, M$, which provides a low-dimensional basis for online learning of the reduced-order system matrix. The details of the learning are introduced in section 3.3.2 below. One can use $V_L(q) = V_D(q)$, but this is not a requirement. Indeed, we show in the numerical examples that the learning and detection bases can be different. For instance, the learning basis could be computed through the eigenvalue decomposition of the system matrix $A(q_i)$. Our proposed method is agnostic to the selection of method to compute the reduced-order basis. Nevertheless, the basis needs to be well chosen to reflect the dynamics of the system matrix.

The $\tilde{r}_i$-dimensional detection basis $V_D(q_i)$ is used to detect a change in the underlying parameter by a suitable projection, as described below. Typically, fewer basis vectors are needed to classify dynamic regimes, compared to accurately representing the dynamics. In our numerical experiments, we thus use $\tilde{r}_i < r_i$, as specified in section 4. Moreover, the number of basis functions can be different for each dynamic regime.

In sum, the library $\mathcal{L}$ contains the feedback gains $K(q_i)$, the learning bases $V_L(q_i)$, and the detection bases $V_D(q_i)$, for $i = 1, \ldots, M$; that is,

$$\mathcal{L} := \left\{ \begin{Bmatrix} V_L(q_1) \\ V_D(q_1) \\ K(q_1) \end{Bmatrix}, \ldots, \begin{Bmatrix} V_L(q_M) \\ V_D(q_M) \\ K(q_M) \end{Bmatrix} \right\}. \tag{15}$$

The offline construction of the library allows us to use the precomputed expensive high-fidelity information in the online stage, when the system is under operation. The details of the online stage are given in the next section.

3.3. Online: Detecting parameter changes and updating feedback matrix. The online stage of our algorithm is comprised of two parts. First, we build an classifier that decides if the underlying system parameter has changed, and subsequently switches the feedback controller to our best-fit feedback gain from the library. If a sudden parametric change in $A(q(t))$ occurs, switching the feedback law can quickly stabilize the dynamics until more information about the parametric change becomes available (through the observed data). Second, the algorithm initiates a learning mechanism to subsequently update the feedback gain as more data are processed.

We apply the online algorithm to a system with external disturbances $g(t)$ that enter through $B_d \in \mathbb{R}^{n\times m_g}$. Adding the disturbance to system (1) yields a disturbed system model

$$\dot{z}(t;q(t)) = A(q)z(t;q(t)) + Bu(t;q(t)) + B_d g(t), \quad t > 0, \tag{16}$$

with otherwise similar controlled outputs $y(t;q(t)) = Cz(t;q(t))$, and initial conditions $q(0) = q_0$ and $z(0;q_0) = z_0$. Note that this does not alter the solution to the LQR problem, yet it provides a more realistic model of a system plant, and it allows us to test the robustness of our derived controller with respect to external disturbances.

3.3.1. Detection of parametric changes. As the time-varying parameters $q(t)$ undergo piecewise constant transitions that alter the system dynamics, it is key for our control method to quickly detect such changes. After identification of a change in the parameter, we can quickly access the feedback gains from the library $\mathcal{L}$ defined above to change the control, $u(t;q(t)) = -K(q(t))z(t;q(t))$, and provide the proper basis for learning the model, V_L. For classification, we use $p' \in \{1,\dots,n\}$ entries of the full state $z(t;q(t))$, where $p' = n$ would imply using all states, but for computational efficiency reasons—classification happens online—we use only $p' \ll n$. Let $\mathcal{S}$ be a selection operator that selects p' entries from the states $z(t;q(t))$. Then, we define a classifier, i.e., a mapping from a subset of the state vector z to the index of the parameters in the library, $h : \mathbb{R}^{p'} \to \{1,\dots,M\}$ via

$$h(\mathcal{S}z(t;q_k)) = k. \tag{17}$$

The choice of the classification function $h(\cdot)$ and selection operator $\mathcal{S}$ is important for the success of the proposed method and at the same time a challenging task. In machine learning, various classifiers have been introduced that optimize different performance and selection metrics; see, e.g., [49]. However, in the context of modeling physical systems, classification relies on additional design choices, such as sensor placement. Classification methods that explicitly take into account the placement of sensors have been recently shown to successfully classify states of dynamical systems into regimes of characteristic behaviors [12, 39, 11, 23, 10], and further improvements are ongoing. The suitability of a given classification approach also depends on the problem character. For PDE-type systems, especially fluid flows, we follow the classification approach in [23] based on random selection of $p' \in \{1,\dots,n\}$ elements, which relies on compressed sensing [14]. It was observed in [23] that the approach is robust to measurement noise and external disturbances $g(t)$.

Let $\mathcal{S}$ select p' entries from the state *uniformly at random.* Then, consider the projection $P_i : \mathbb{R}^{p'} \mapsto \mathbb{R}^{p'}$ defined as

$$P_i = \mathcal{S}V_D(q_i)[(\mathcal{S}V_D(q_i))^T(\mathcal{S}V_D(q_i))]^{-1}(\mathcal{S}V_D(q_i))^T. \tag{18}$$

Thus, for randomly selected partial states $\mathcal{S}z(t;q)$, we define the online classifier in the sense of (17) as

$$k = \arg\max_{i=1,\dots,M} ||P_i(\mathcal{S}z(t;q))||_2. \tag{19}$$

The classifier requires a computationally cheap product of the selected state entries $\mathcal{S}z(t;q)$ with the $p' \times p'$ matrices P_i for each basis $V_D(q_i)$ in the library $\mathcal{L}$.

We note that the classification problem (i.e., the mapping from the selected states of a system with time-varying parameters to bases from a library) is different from the task of optimal state reconstruction. In practice, having a large number of elements in the library can cause misidentification, thus we must ensure that the library contains a suitable amount of information for classification. This can be achieved via heuristics, using prior knowledge of the system and its parametric dependence, or by more rigorous approaches such as measuring the alignment of the bases and corresponding subspaces, as well as the energy in the subspaces as presented in [23].

3.3.2. Model learning. The data from the operating plant will in general not be exactly represented in the precomputed bases of the library. In other words, the current dynamics may be generated by a parameter that was not in the training set, $\tilde{q} \notin \{q_1, q_2, \ldots, q_M\}$. Thus, we opt to learn and update low-dimensional models from data of the underlying system/plant with the operator inference procedure as developed in [34]. For ease of notation, let $V_L = V_L(q_i) \in \mathbb{R}^{n\times r}$ be the low-dimensional learning basis, computed for a particular parameter q_i. Moreover, let $\hat{B} = V_L^T B$ and $\hat{B}_d = V_L^T B_d$, assume that a record of the past control inputs $u_k = u(t_k; q(t_k))$, and assume that the disturbance model $g_k = g(t_k)$ for some sampling times $t_1 < t_2 < \cdots < t_s$ is available and stored in

$$U = [u_1, u_2, \ldots, u_s]^T \in \mathbb{R}^{s\times m}, \qquad G = [g_1, g_2, \ldots, g_s]^T \in \mathbb{R}^{s\times m_g}.$$

Our aim is to estimate a reduced system matrix $\hat{A}(\tilde{q}) \in \mathbb{R}^{r\times r}$ from data $z(t; \tilde{q})$ of the system (16). The reduced states at discrete time instances t_i for $i = 1, \ldots, s$ are denoted by $\hat{z}_i := V_L^T z(t_i)$ and stored in

$$\hat{Z} = [\hat{z}_1, \hat{z}_2, \ldots, \hat{z}_s]^T \in \mathbb{R}^{s\times r}. \tag{20}$$

The derivative of the reduced state is approximated by a finite differencing scheme with time step size Δt, so that $\dot{\hat{z}}_i := \frac{\hat{z}_{i+1} - \hat{z}_i}{\Delta t}$. The finite differences are recorded in the right-hand-side matrix

$$\hat{O} = [\dot{\hat{z}}_1, \dot{\hat{z}}_2, \ldots, \dot{\hat{z}}_s]^T \in \mathbb{R}^{s\times r}. \tag{21}$$

The operator inference problem for $\hat{A} = \hat{A}(\tilde{q})$ then becomes

$$\min_{\hat{A}\in\mathbb{R}^{r\times r}} \sum_{i=1}^{s} \left\| \dot{\hat{z}}_i - \hat{A}\hat{z}_i - \hat{B}u_i - \hat{B}_d g_k \right\|_2^2, \tag{22}$$

which can be rewritten [34, sect. 3.2] as

$$\min_{\hat{A}\in\mathbb{R}^{r\times r}} \|\hat{O} - U\hat{B}^T - G\hat{B}_d^T - \hat{Z}\hat{A}^T\|_F^2. \tag{23}$$

The operator inference problem requires $s = r$ linearly independent snapshots of the reduced state z_i and the finite difference approximations $\dot{\hat{z}}_i$ to solve the least-squares problem in (22). However, s subsequent snapshots of a dynamical system are not necessarily numerically linearly independent. In practice, we often need to collect $s \gg r$ snapshots to get a numerically

well-conditioned least-squares problem. The number of snapshots to solve the least-squares problem depends on the system dynamics and selected sampling time, as smaller sampling times are more likely to lead to numerically linearly dependent snapshots. This observation then translates into a requirement for the parameter $q(t)$, namely that it is constant at least over a time interval of $|\mathcal{T}_i| > s\Delta t$ during which we can collect numerically linearly independent snapshots to infer $\hat{A}$.

The operator inference problem can be solved online with a least-squares approximation in $\mathcal{O}(sr^3)$ operations. It was shown in [34, Thm. 1] that the inferred matrix $\hat{A}$ recovers the matrix $\hat{A}(\tilde{q}) = V_L^T A(\tilde{q}) V_L$ that would be obtained from (intrusive) projection of the system matrix onto the reduced basis V_L. The recovery takes place under the condition that a sufficient amount of data $z(t)$ is available and that the time discretization is convergent. Once the system matrix is inferred, the ROM reads as

$$\dot{\hat{z}}(t;q) = \hat{A}(\tilde{q})\hat{z}(t;q) + \hat{B}\tilde{u}(t) + \hat{B}_d g(t), \tag{24}$$

and can serve as a computationally cheap surrogate for the high-fidelity model. The updated ROM is available after s time steps of data are collected. The inferred matrix representations are subsequently used to compute the solution to a low-dimensional ARE as

$$\hat{A}^T\hat{\Pi} + \hat{\Pi}\hat{A} - \hat{\Pi}\hat{B}\hat{B}^T\hat{\Pi} + \hat{C}^T\hat{C} = 0. \tag{25}$$

An approximation of the feedback gain is then obtained via

$$K(\tilde{q}) \approx \hat{K} V_L(q_k)^T = R^{-1} B^T \hat{\Pi} V_L^T(q_k),$$

which is used to update the current feedback gain, so that the control becomes $u(t;\tilde{q}) = -K(\tilde{q})z(t;\tilde{q})$. Note that we never used the actual value of $\tilde{q}$; we only used *data* $z(t;\tilde{q})$ of the system available online.

Remark 3.1. In (23), we learn the parameter-dependent system matrix $\hat{A}$ from data. This approach could be extended to learning the control input matrix $\hat{B}$ by using a learning step on $[\hat{A} \;\; \hat{B}]$ similar to [36, sect. 3.3].

3.3.3. Complete algorithm. In the online stage of the algorithm, we initialize the controller with the feedback gain $K(q_0)$. Then, at every time step, we evaluate the classifier $h(y(t;q(t)))$ from (17). If the result indicates a change to a new regime, say k, the algorithm then uses the high-fidelity feedback gain $K(q_k)$ until a new model is learned. In Algorithm 1 below, we summarize the steps of our hybrid method, which were previously shown in Figure 1 above.

3.3.4. Online costs of the method. This section discusses the costs of adapting the gain with Algorithm 1. Three steps in Algorithm 1 dominate the costs: (1) detecting the best fit basis in $\mathcal{L}$ (line 2), (2) solving for the reduced operator $\hat{A}$ (line 12), and (3) solving the ARE for $\hat{\Pi}$ (line 13). Detecting the best fit basis in line 2 requires projecting the current selected states $\mathcal{S}z(t) \in \mathbb{R}^{p'}$ onto M bases in the library, with total costs bounded in $\mathcal{O}(mM^2)$. The system of linear equations in line 12 is of size $s \times r$, with s being the number of collected data

Algorithm 1. Online detection and model updating with control.

Input: Model and gain library $\mathcal{L}$, initial $K_0 \in \mathcal{L}$, data window s

1: Initialize control: $u(t; q_0) \leftarrow -K_0 z(t)$
2: Detect basis in $\mathcal{L}$: $k = \arg\max_{i=1,\dots,M} ||P_i y(t; q(t))||_2$ as in (18)
3: Use $K \leftarrow K(q_k) \in \mathcal{L}$
4: $\hat{B} \leftarrow V_L(q_k)^T B$, $\hat{C} \leftarrow C V_L(q_k)$ from $\mathcal{L}$
5: **for** $i = 1, \dots, s$ **do**
6: $\quad \hat{z}_i = V_L(q_k)^T z(t_i)$
7: $\quad \hat{Z} = [\hat{Z}^T \ \hat{z}_i]^T$
8: $\quad U = [U \ u_i^T]^T$
9: $\quad G = [g_1, g_2, \dots, g_s]^T$
10: $\quad O = [O \ \dot{\hat{z}}_i^T]^T$, where $\dot{\hat{z}}_i := (\hat{z}_i - \hat{z}_{i-1}) \backslash (t_i - t_{i-1})$
11: **end for**
12: Solve

$$\min_{\hat{A} \in \mathbb{R}^{r \times r}} \|R - U\hat{B}_k^T - \hat{Z}\hat{A}^T\|_F^2 \tag{26}$$

For instance, in Matlab $\hat{A} = (\hat{Z} \backslash [R - U\hat{B}_k^T])^T$
13: Solve ARE for $\hat{\Pi}$:

$$\hat{A}^T\hat{\Pi} + \hat{\Pi}\hat{A} - \hat{\Pi}\hat{B}R^{-1}\hat{B}^T\hat{\Pi} + \hat{C}\hat{C}^T = 0 \quad \in \mathbb{R}^{r \times r}$$

14: Update gain: $K \leftarrow R^{-1}B^T\hat{\Pi}V_L(q_k)^T$
15: Apply control to system $u(t) \leftarrow -Kz(t)$

for learning the model online, and r being the ROM dimension of the learned model, where typically $s \gg r$. A crude upper bound on solving an $s \times r$ least-squares problem is $\mathcal{O}(sr^3)$. Solving the ARE in line 13 is bounded in $\mathcal{O}(r^3)$. In total, the costs of adapting the gain with Algorithm 1 are in $\mathcal{O}(p'M^2 + sr^3)$.

4. Numerical results.

We present numerical results for two PDE models of fluids. Section 4.1 considers a two-dimensional model of a flow through a porous medium, where the permeability field is uncertain. The model in section 4.2 is a convection-diffusion equation in two dimensions, where the uncertain parameter is the viscosity of the fluid.

4.1. Permeability of porous media.

4.1.1. Problem setup.

We consider a two-dimensional PDE that models flow through a porous medium. The material's permeability, a spatially varying parameter field, describes the ability of a porous medium to allow fluids to pass through it. The resulting model is given by a Laplace equation of the form

$$\frac{\partial}{\partial t}\theta(t, \boldsymbol{x}) = \nu(t, \boldsymbol{x}) \cdot \left(\frac{\partial^2}{\partial x_1^2} + \frac{\partial^2}{\partial x_2^2}\right)\theta(t, \boldsymbol{x}) + b(\boldsymbol{x})u(t) + b_d^1(\boldsymbol{x})g_1(t) + b_d^2(\boldsymbol{x})g_2(t),$$

where the state $\boldsymbol{x} = [x_1, x_2]^T \in \Omega = [0,1]^2$, and the time $t \in (0, \infty)$. Here, $\nu(t, \boldsymbol{x})$ is the uncertain permeability field, assumed to be zero on the boundary of the domain. The function $b(\cdot)$ is a bivariate normal distribution with mean at $\boldsymbol{x} = [.6, .7]^T$ and standard-deviation 3×10^{-2}. The external disturbances enter through $b_d^1(\cdot)$ (bivariate normal with mean $\boldsymbol{x} = [.3,\ .5]^T$ and standard deviation 3×10^{-2}) and $b_d^2(\cdot)$, which is also bivariate normal with the same standard deviation, but mean at $\boldsymbol{x} = [.3, .7]^T$. Moreover, $\theta(t, \boldsymbol{x})$ is interpreted as the velocity of the flow at time t and space coordinate $\boldsymbol{x}$. As boundary conditions for the velocity of the fluid, we impose the Dirichlet conditions

$$\theta(t, x_1, 0) = 0, \quad \theta(t, 1, x_2) = 0, \quad \theta(t, x_1, 1) = 0, \quad \theta(t, 0, x_2) = 0.5.$$

The controlled output of the model is given by

$$\eta(t) = \int_\Omega c(\boldsymbol{x})\theta(t, \boldsymbol{x}) d\boldsymbol{x}, \tag{27}$$

where the function $c(\cdot)$ is modeled as a bivariate normal distribution with mean at $\boldsymbol{x} = [.5, .6]^T$ and the same standard deviation 3×10^{-2}. A spatial discretization with finite differences leads to the system of ordinary differential equations

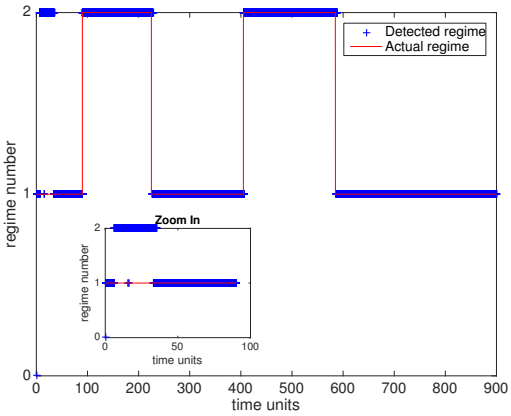
$$\dot{z}(t) = A(q(t))z(t) + Bu(t) + B_d g(t), \tag{28}$$

$$y(t) = Cz(t), \tag{29}$$

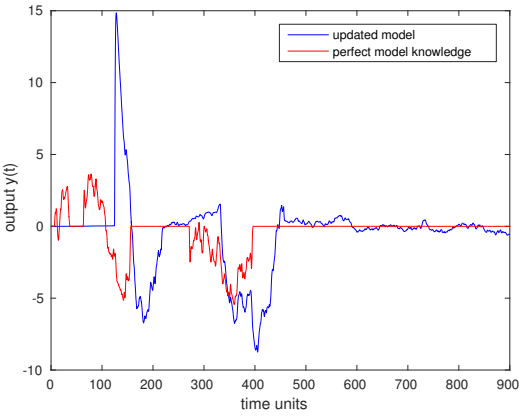
where $z(t)$ is the finite-dimensional state variable and $y(t)$ the controlled output of the model. Similarly, the parameters $q(t)$ are the spatially discretized version of $\nu(t, \boldsymbol{x})$. The matrix B_d consequently has two columns, and B only a single column. The measurement matrix C has one row.

4.1.2. Offline stage. To compute the library of high-fidelity gains, as well as detection and learning bases, we generated three different permeability fields $\nu_1(\boldsymbol{x}), \nu_2(\boldsymbol{x}), \nu_3(\boldsymbol{x})$ leading to spatially discretized parameters q_1, q_2, q_3. The two-dimensional permeability fields $\nu_i(\boldsymbol{x})$ are generated as follows: All three fields are initialized identically. Then, for each $\nu_i(\boldsymbol{x})$ we select a different area where its permeability is $0.5, 0.7, 0.6$ times its baseline value, respectively. This results in the permeability fields being parametrized by the multiples above, and the location of the deviation from the baseline value. The permeability q_1 leads to a stable dynamical system, whereas the parameters q_2 and q_3 lead to unstable dynamical systems. For the unstable cases, stabilization through the controller is most important, so that unbounded growth can be prevented. The penalty on the control action is set to $R = 0.1 \cdot I_n$.

For each permeability field, we compute the eigenbasis of $A(q_i)$ and keep the leading 20 basis vectors, which we then store as the learning basis $V_L(q_i)$, for $i = 1, 2, 3$. For detection, we compute a POD basis from simulation of the closed-loop system (28) with two external disturbances $g(t) = [g_1(t), g_2(t)]^T$, modeled as a Gaussian noise process with mean six and variance $\sigma = 3$. The initial condition is set to zero. The system is simulated for 500 time units with a backward Euler time integration scheme with time step 5×10^{-2}. Then, $S = 10,000$ snapshots are used to compute the POD basis. We store the POD basis of order 20 as a detection basis in $V_D(q_i)$, for $i = 1, 2, 3$ in the library $\mathcal{L}$.



(a) Selected permeabilities as indicated by the detection function $h(\cdot)$.



(b) The output $y(t)$ of the system with control derived from the updated model (+), compared to an "ideal" model where the matrices are known and optimal controllers are available (−).

Figure 2. *Switches in the permeabilities, as indicated by the index, together with the results of the detection function* (a). *The controlled output of the system with a learned model, and hence learned controller, is compared to a hypothetical situation where we have perfect knowledge of the parameters and their transitions. In the latter case, we used a controller computed from an intrusive projection-based model* (b).

4.1.3. Online stage. The online stage of the algorithm considers a time horizon of $t_f = 900$ time units, where the backward Euler time integration methods takes time steps of size 3×10^{-2}. We choose the time-varying parameter $q(t)$ as in Figure 2(a). The time-varying parameter changes between q_1 and q_2 at the transition times $t_i = \{60, 150, 270, 390\}$, resulting in five time intervals $\mathcal{T}_i$. The time evolution of $q(t)$ is not known to our online adaptive control algorithm, and has to be detected. Thus, Figure 2(a) also shows the results of the detection method from section 3.3, which our proposed algorithm uses to detect parametric changes during online operation. We switched the dynamic regime to the new indicated index k by using (19) if the past ten classification steps yielded identical results. The projection method provided the correct regime in more than 94% of the cases. We purposely neglected the third dynamic regime with q_3, but included it in the library. The detection results from Figure 2(a) also illustrate that there was not a single instance where parameter q_3 was detected.

The corresponding output $y(t)$ of the controlled system (28) for two different controllers is shown in Figure 2(b). Our proposed control strategy follows Algorithm 1, which first detects the parameter switches as indicated in plot (a), learns the system matrices from data online, and then recomputes the feedback gains. We compare this approach to an ideal scenario, where it is assumed that the system matrix $A(q(t))$ and the correct low-dimensional basis for projection are known at all times. In that case, we can compute projection-based (intrusive) ROMs, and obtain the feedback matrix by solving the low-dimensional Riccati equation. From Figure 2(b), we see that our offline-online strategy successfully stabilizes the large-scale system, and rejects the external disturbances. In the online phase, the algorithm solely relies on the precomputed library $\mathcal{L}$ and data of the system.

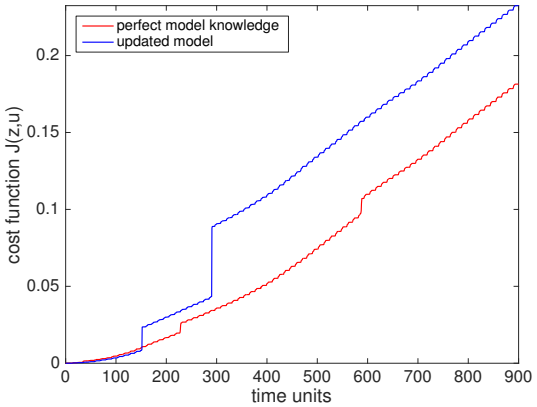


Figure 3. *Control cost function $J(z, u)$ for both controllers on the full-order model.*

Figure 2 shows that the controller successfully drives the controlled output to zero, and holds it there in the presence of external disturbances $g(t)$. A quantitative metric for the performance of a controller is given by the control cost (objective function in (3)). In Figure 3 we plot the control cost function for both controllers, i.e., the "ideal" ROM controller obtained from the perfect model and parameter knowledge, and the one computed by learning the reduced-order model from data.

Let us now compare the feedback gains computed from both approaches. In Figure 4, the left two plots show the feedback gains computed from the intrusive projection-based reduced-order model. The right two plots then show the feedback gains computed from the learned ROMs. The results are shown at $t = 122$, which corresponds to the permeability q_2, and at time $t = 574$, which corresponds to the permeability q_1. Figure 4 then shows that the learned feedback gains indeed look qualitatively similar to their intrusively computed counterparts.

4.2. Convection-diffusion equation.

4.2.1. Problem setup. We consider another model from fluid dynamics, namely the convection-diffusion equation as a model for particle transfer. To that end, let $\theta(t, \boldsymbol{x})$ be a species concentration satisfying the PDE

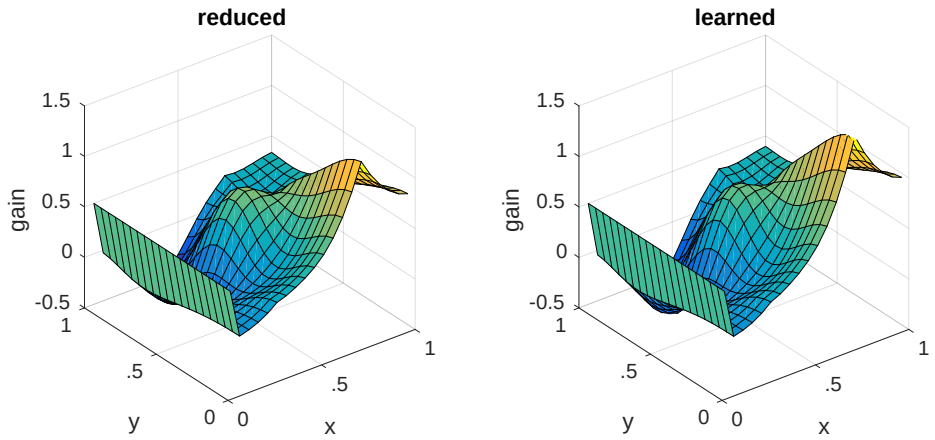
$$\frac{\partial \theta}{\partial t}(t, \boldsymbol{x}) + x_2 \frac{\partial \theta}{\partial x_2}(t, \boldsymbol{x}) = q(t) \left(\frac{\partial^2}{\partial x_1^2} + \frac{\partial^2}{\partial x_2^2} \right) \theta(t, \boldsymbol{x}) + b(\boldsymbol{x}) u(t) + b_d(\boldsymbol{x}) g(t)$$

for $\boldsymbol{x} = [x_1, x_2]^T \in \Omega = [0, 1]^2$ with Dirichlet boundary conditions on the bottom, right, and top walls:

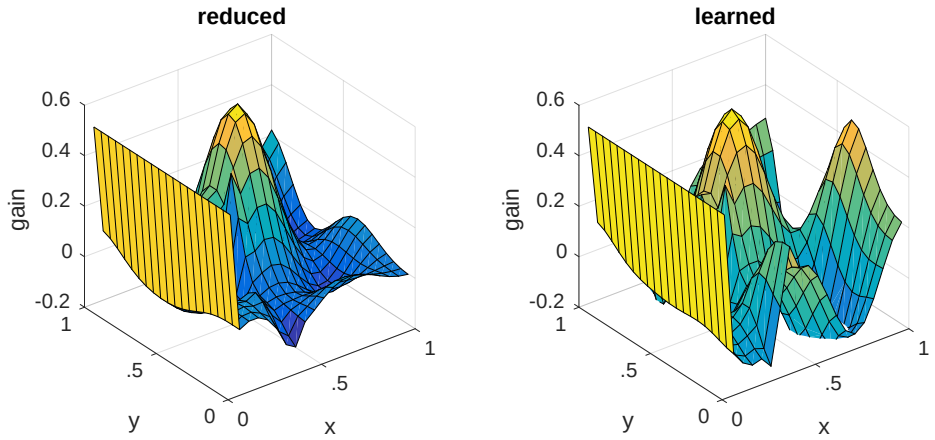
$$\theta(t, x_1, 0) = 0, \quad \theta(t, 1, x_2) = 0, \quad \theta(t, x_1, 1) = 0,$$

and Neumann boundary condition on the left wall:

$$\frac{\partial \theta}{\partial x_1}(t, 0, x_2) = 0.$$



(a) Feedback gains at $t = 122s$.



(b) Feedback gains at $t = 574s$.

Figure 4. *The feedback gains computed from an intrusive ROM, where it is assumed that the permeability is known, so that the system matrix can be projected onto the POD basis (left); computed from the learned model without any knowledge of the parameters (right).*

We choose $b(\boldsymbol{x}) = 5$ if $x_1 \geq 1/2$ and 0 otherwise. The uncertain parameter is $q(t) \in \mathbb{R}_+$ for all $t > 0$, the diffusivity, which undergoes piecewise constant transitions in time. However, the time instances at which the transitions occur are again unknown, and have to be detected by our online routine. The controlled output is a weighted average of the concentration:

$$\eta(t) = \int_\Omega 5 \cdot \theta(t, \boldsymbol{x}) d\boldsymbol{x}.$$

The model is discretized in space with a finite element method with piecewise linear basis functions, leading to a state-space dimension $n = 3540$, so that the high-fidelity model with external disturbances reads as

$$\dot{z}(t; q) = A(q(t))z(t; q) + B\tilde{u}(t) + B_d g(t), \quad z(0) = z_0 \in \mathbb{R}^n, \tag{30}$$

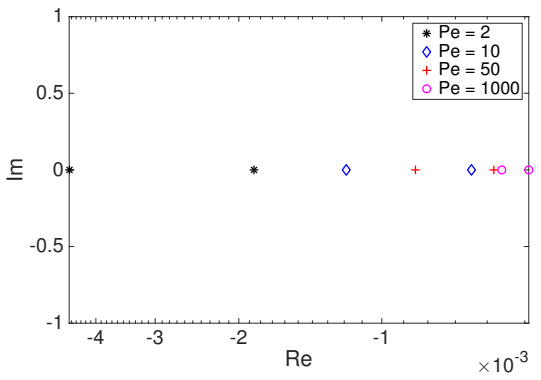


Figure 5. *The two eigenvalues with largest real part of the system matrix $A(q_i)$ from the convection-diffusion equation. The eigenvalues move closer to the imaginary axis as the Péclet number grows, and convection dominates.*

with corresponding output

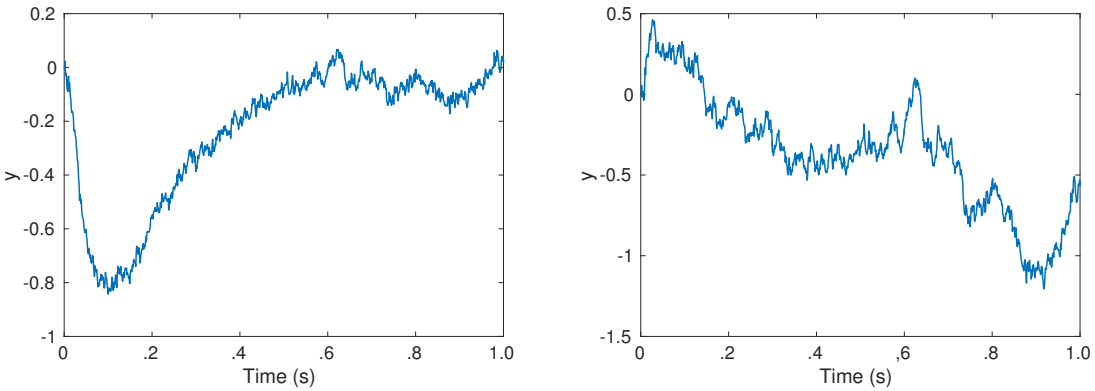
$$y(t) = Cz(t) \in \mathbb{R}.$$

Here, $m = 1$, so there is only one control input, and the disturbance $g(\cdot)$ enters through the left boundary of the domain at $0 \leq x_1 \leq 0.05$. In this setting, a characteristic non-dimensional quantity that qualitatively describes the flow behavior is given by the Péclet number, which quantifies the relative importance of the convection with respect to the diffusion. High Péclet numbers indicate strongly convective flows.

4.2.2. Offline stage. To generate the library $\mathcal{L}$, we pick four different diffusivity values, namely $q_1 = 5 \times 10^{-1}$, $q_2 = 10^{-1}$, $q_3 = 5 \times 10^{-2}$, and $q_4 = 10^{-3}$. This leads to Péclet numbers $Pe \in \{2, 10, 50, 1000\}$. We first illustrate the qualitative and quantitative differences in the dynamics when Pe changes.

Figure 5 shows the two eigenvalues with largest real part of the parameter-dependent system matrix $A(q_i)$ with corresponding Péclet numbers. The larger the Péclet number, the closer the spectrum of the system matrix to the imaginary axis. By design, stabilization and control will move the spectrum of the closed-loop system matrix $[A(q_i) - BK(q_i)]$ further away from the imaginary axis.

Figure 6 shows the open loop outputs $y(t)$ of the convection-diffusion model for two Péclet numbers, $Pe_1 = 2$ and $Pe_4 = 1000$, respectively. These outputs are generated starting from the initial condition $z_0 = 15 \sin(2\pi x_1) \sin(\pi x_2)$, and by imposing an external excitation $g(t)$ in the form of Gaussian noise with variance $\sigma = 0.5$. The backward Euler time discretization was solved until $t_f = 1$ time units, with time step size $\Delta t = 10^{-3}$. In the case where the Péclet number is small, the output quickly returns to zero. However, for the larger Péclet number, the output grows initially, then crosses zero and becomes negative; see Figure 6(b). Such a scenario requires more control action to drive the output back to zero. Recall that the control cost function (3) penalizes the deviation of the controlled output from zero, and balances this with the invested control cost.



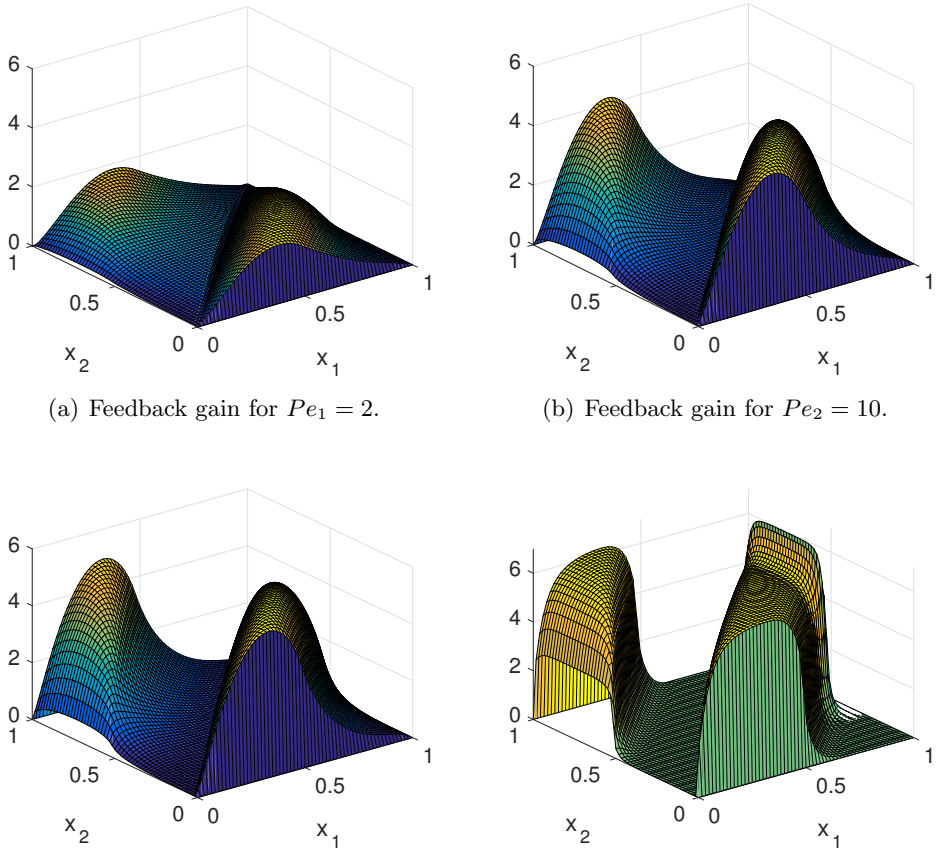
(a) Output for $Pe_1 = 2$. (b) Output for $Pe_2 = 1000$.

Figure 6. *Output $y(t)$ of the open loop convection-diffusion system, excited with nonzero initial condition $z_0 = 15\sin(2\pi x)\sin(\pi y)$ and Gaussian disturbance $g(t)$ with variance $\sigma = 0.5$ applied through a disturbance term at $0 \leq x_1 \leq 0.05$. For low Péclet number, the uncontrolled output returns to the zero state by the end of the simulation* (a)*, whereas for high Péclet number the system remains away from the zero state* (b)*.*

To generate the library $\mathcal{L}$, we compute high-fidelity LQR feedback gains from equations (5)–(6) with control penalty $R = 0.1$ for the four selected Péclet numbers. The resulting optimal feedback gains are plotted in Figure 7 with similar scaling. The feedback gains show significant differences, both qualitatively and quantitatively. Hence, adapting the control when the ratio of convection to diffusion changes becomes important. This can be seen both by looking at the feedback gains in Figure 7 as well as by considering the spectrum of the open loop operators in Figure 5.

We generate the learning basis $V_L(q_i)$ from the eigendecomposition of the four resulting system matrices, keeping the leading 10 basis functions. The detection basis $V_D(q_i)$ is computed with the POD method from $S = 1,000$ snapshots from zero to $t_f = 1$, and the leading 30 left singular vectors (i.e., POD basis functions) of the snapshot matrix are kept. We used the same initial condition z_0 as for the open loop simulation above.

4.2.3. Online stage. To test the method online, a longer time horizon of $t_f = 2.5$ is considered, where the system of ordinary differential equations (30) is solved with the backward Euler scheme with constant step size $\Delta t = 1.3 \times 10^{-3}$. In Figure 8 the performance of the proposed model is further demonstrated. Part (a) shows the prescribed switches for the four viscosities, and compares this with the result of the detection function $h(\mathcal{S}z(t))$. We switched the dynamic regime to the new indicated index k by using (19) if the past ten classification steps yielded identical results. In Figure 8 (a), we see that once the controlled output reaches zero, the detection misidentifies regime 1 for regime 2. Recall that regime 1 is a dynamical system with a Péclet number $Pe = 2$, and regime 2 uses a Péclet number $Pe = 10$. Both systems are diffusion dominated, and so after approximately $t > 1.5$ they are very close to the zero equilibrium solution (same zero solution for both systems), hence the misclassification. Thus, after some time, the two regimes are similar.



(a) Feedback gain for $Pe_1 = 2$.

(b) Feedback gain for $Pe_2 = 10$.

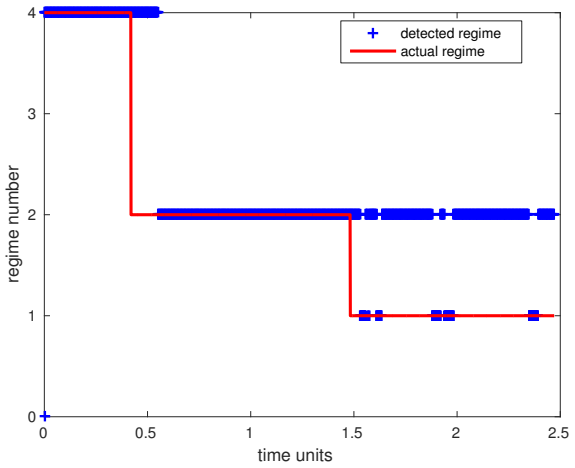
(c) Feedback gain for $Pe_3 = 50$.

(d) Feedback gain for $Pe_4 = 1000$.

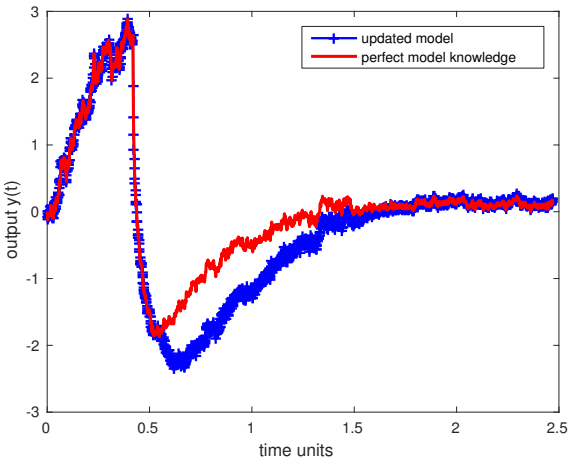
Figure 7. *The feedback gains according to four different viscosities.*

Figure 8(b) then shows the controlled output of the full closed-loop model with two different controllers. The control computed from the learned ROM (blue +) is compared to the controller obtained from a direct projection-based, intrusive model, as in the previous example. The latter assumes that the parameter and its prescribed transitions (red line in plot (a)) are known. While this is unrealistic, it serves as a best-case comparison to the control we computed with the offline-online method. We conclude by observing that our learning-based controller is successful in rejecting the disturbance and driving the controlled output to zero, as targeted by the control cost formulated in (3). We also see that the misclassification after $t > 1.5$ does not affect the performance of the controller.

5. Discussion and conclusions. This work combines methods from data-driven reduced-order modeling, and optimal LQR feedback control to arrive at a computationally feasible suboptimal control and stabilization strategy for dynamical systems with time-varying parameters. The system parameters are considered to be uncertain and unknown in real time. Our method leverages libraries computed offline to avoid the expensive step of estimating the



(a) Selected library elements as indicated by the detection function $h(\cdot)$ for the convection-diffusion problem.



(b) The output $y(t)$ of the system with control derived from the updated model (+), compared to an "ideal" model where the matrices are known, and optimal controllers are avaibalbe (−).

Figure 8. *In this example, the detection algorithm was correct in more than* 85% *of the cases, where it identified the correct basis* (a). *The output of the learned model with recomputed controller is compared to a hypothetical situation where we have perfect knowledge of the parameters and their switching times.*

uncertain system parameter during operation. In doing so, we combine data-based methods with physics-based modeling towards control.

By using learned ROMs, our method is feasible for a large class of applications. In particular, we incorporate data from the system plant into a state-space model learning procedure. We also use the expensive-to-evaluate high-fidelity model (e.g., industrial legacy code) when building the library of feedback controllers and low-dimensional bases. This allows us to extract information from both first-principles modeling and real system data.

Our method circumvents the possible error-prone need to estimate the time-varying uncertain system parameters before assembling the system matrix. Moreover, from a computational perspective, building $\hat{A}(q(t))$ in an intrusive reduced-order modeling framework requires estimating the parameters, building a high-fidelity model with the available legacy code (expensive), and subsequently projecting the model to reduced dimensions—we replace those three steps with a single step. One instance where this is advantageous is in the treatment of boundary conditions in PDE-based modeling—the boundary conditions are typically built in to the approximation spaces, as well as the formulation of the dual problem. If the boundary conditions are unknown and uncertain, building the approximating spaces becomes a formidable task. However, if we build ROMs from data of the actual system—which automatically reflects the boundary conditions—then the inferred model is built from the proper set of boundary conditions.

The numerical results for a fluid flowing through a porous medium and a convection-diffusion flow show that the learned controllers successfully stabilize the plant, and drive the controlled output back to the zero solution. Moreover, the feedback gains show strong

similarities to the gains obtained from an intrusive, projection-based model. By learning the reduced-order state-space model, we are able to learn the control mechanism, as evidenced by the feedback gains.

In the future, we are looking into extending this data-driven method to use partial state information for the control task, in the framework of Kalman filtering and linear quadratic Gaussian (LQG) design. This non-trivial task will include a second learning procedure for low-dimensional representation and learning of the filtering gains.

REFERENCES

[1] I. Akhtar, J. Borggaard, J. A. Burns, H. Imtiaz, and L. Zietsman, *Using functional gains for effective sensor location in flow control: A reduced-order modelling approach*, J. Fluid Mech., 781 (2015), pp. 622–656, https://doi.org/10.1017/jfm.2015.509.

[2] A. Alla and M. Falcone, *An adaptive POD approximation method for the control of advection-diffusion equations*, in Control and Optimization with PDE Constraints, K. Bredies, C. Clason, K. Kunisch, and G. von Winckel, eds., Springer, Basel, 2013, pp. 1–17.

[3] A. C. Antoulas, *Approximation of Large-Scale Dynamical Systems*, Advances in Design and Control, Society for Industrial and Applied Mathematics, Philadelphia, PA, 2005.

[4] J. Atwell, J. Borggaard, and B. King, *Reduced order controllers for Burgers' equation with a nonlinear observer*, Appl. Math. Comput. Sci., 11 (2001), pp. 1311–1330.

[5] G. Becker and A. Packard, *Robust performance of linear parametrically varying systems using parametrically-dependent linear feedback*, Systems Control Lett., 23 (1994), pp. 205–215.

[6] P. Benner, M. Heinkenschloss, J. Saak, and H. K. Weichelt, *An inexact low-rank Newton–ADI method for large-scale algebraic Riccati equations*, Appl. Numer. Math., 108 (2016), pp. 125–142.

[7] P. Benner, H. Mena, and J. Saak, *On the parameter selection problem in the Newton-ADI iteration for large-scale Riccati equations*, Electron. Trans. Numer. Anal., 29 (2008), pp. 136–149.

[8] P. Benner and J. Saak, *Numerical Solution of Large and Sparse Continuous Time Algebraic Matrix Riccati and Lyapunov Equations: A State of the Art Survey*, Preprint MPIMD/13-07, Max Planck Institute Magdeburg, 2013.

[9] J. Borggaard and M. Stoyanov, *An efficient long-time integrator for Chandrasekhar equations*, in Proceedings of the 47th IEEE Conference on Decision and Control, Cancun, Mexico, 2008, pp. 3983–3988, https://doi.org/10.1109/CDC.2008.4738965.

[10] B. Brunton, S. Brunton, J. Proctor, and J. Kutz, *Sparse sensor placement optimization for classification*, SIAM J. Appl. Math., 76 (2016), pp. 2099–2122.

[11] S. L. Brunton, J. L. Proctor, and J. N. Kutz, *Discovering governing equations from data by sparse identification of nonlinear dynamical systems*, Proc. Natl. USA Acad. Sci., 113 (2016), pp. 3932–3937, https://doi.org/10.1073/pnas.1517384113.

[12] S. L. Brunton, J. H. Tu, I. Bright, and J. N. Kutz, *Compressive sensing and low-rank libraries for classification of bifurcation regimes in nonlinear dynamical systems*, SIAM J. Appl. Dyn. Syst., 13 (2014), pp. 1716–1732.

[13] M. Döhler and L. Mevel, *Fast multi-order computation of system matrices in subspace-based system identification*, Control Eng. Practice, 20 (2012), pp. 882–894.

[14] D. L. Donoho, *Compressed sensing*, IEEE Trans. Inform. Theory, 52 (2006), pp. 1289–1306.

[15] Z. Drmac, S. Gugercin, and C. Beattie, *Quadrature-based vector fitting for discretized $\mathcal{H}_2$ approximation*, SIAM J. Sci. Comput., 37 (2015), pp. A625–A652.

[16] F. Feitzinger, T. Hylla, and E. W. Sachs, *Inexact Kleinman–Newton method for Riccati equations*, SIAM J. Matrix Anal. Appl., 31 (2009), pp. 272–288.

[17] F. Guéniat, L. Mathelin, and M. Hussaini, *A statistical learning strategy for closed-loop control of fluid flows*, Theor. Computational Fluid Dyn., 30 (2016), pp. 497–510, https://doi.org/10.1007/s00162-016-0392-y.

[18] B. Gustavsen and A. Semlyen, *Rational approximation of frequency domain responses by vector fitting*, IEEE Trans. Power Delivery, 14 (1999), pp. 1052–1061.

[19] A. Herv, D. Sipp, P. J. Schmid, and M. Samuelides, *A physics-based approach to flow control using system identification*, J. Fluid Mech., 702 (2012), pp. 26–58, https://doi.org/10.1017/jfm.2012.112.
[20] J. G. Iñigo, D. Sipp, and P. J. Schmid, *A dynamic observer to capture and control perturbation energy in noise amplifiers*, J. Fluid Mech., 758 (2014), pp. 728–753.
[21] A. Ionita and A. Antoulas, *Data-driven parametrized model reduction in the Loewner framework*, SIAM J. Sci. Comput., 36 (2014), pp. A984–A1007.
[22] J.-N. Juang and R. Pappa, *An eigensystem realization algorithm for modal parameter identification and model reduction*, J. Guidance, Control, and Dynamics, 8 (1985), pp. 620–627.
[23] B. Kramer, P. Grover, P. Boufounos, S. Nabi, and M. Benosman, *Sparse sensing and DMD based identification of flow regimes and bifurcations in complex flows*, SIAM J. Appl. Dyn. Syst., 16 (2017), pp. 1164–1196, https://doi.org/10.1137/15M104565X.
[24] B. Kramer and S. Gugercin, *Tangential interpolation-based eigensystem realization algorithm for MIMO systems*, Math. Comput. Model. Dynamical Syst., 22 (2016), pp. 282–306, https://doi.org/10.1080/13873954.2016.1198389.
[25] S.-Y. Kung, *A new identification and model reduction algorithm via singular value decomposition*, in Proc. 12th Asilomar Conf. Circuits, Syst. Comput., Pacific Grove, CA, 1978, pp. 705–714.
[26] K. Kunisch and S. Volkwein, *Control of the Burgers equation by a reduced-order approach using proper orthogonal decomposition*, J. Optim. Theory Appl., 102 (1999), pp. 345–371.
[27] H. Kwakernaak and R. Sivan, *Linear Optimal Control Systems*, Wiley-Interscience, New York, 1972.
[28] C. Lee, J. Kim, D. Babcock, and R. Goodman, *Application of neural networks to turbulence control for drag reduction*, Phys. Fluids, 9 (1997), pp. 1740–1747, https://doi.org/10.1063/1.869290.
[29] Z. Ma, S. Ahuja, and C. Rowley, *Reduced-order models for control of fluids using the eigensystem realization algorithm*, Theoret. Comput. Fluid Dyn., 25 (2011), pp. 233–247.
[30] L. Mathelin, M. Abbas-Turki, L. Pastur, and H. Abou-Kandil, *Closed-loop fluid flow control using a low dimensional model*, Math. Comput. Modelling, 52 (2010), pp. 1161–1168, https://doi.org/10.1016/j.mcm.2010.01.007.
[31] L. Mathelin, L. Pastur, and O. Le Maître, *A compressed-sensing approach for closed-loop optimal control of nonlinear systems*, Theoret. Comput. Fluid Dyn., 26 (2012), pp. 319–337.
[32] A. Mayo and A. Antoulas, *A framework for the solution of the generalized realization problem*, Linear Algebra Appl., 425 (2007), pp. 634–662.
[33] S. Nicaise, S. Stingelin, and F. Tröltzsch, *On two optimal control problems for magnetic fields*, Comput. Methods Appl. Math., 14 (2014), pp. 555–573.
[34] B. Peherstorfer and K. Willcox, *Data-driven operator inference for nonintrusive projection-based model reduction*, Comput. Methods Appl. Mech. Engrg., 306 (2016), pp. 196–215.
[35] C. Poussot-Vassal and D. Sipp, *Parametric reduced order dynamical model construction of a fluid flow control problem*, IFAC-PapersOnLine, 48 (2015), pp. 133–138.
[36] J. L. Proctor, S. L. Brunton, and J. N. Kutz, *Dynamic mode decomposition with control*, SIAM J. Appl. Dyn. Syst., 15 (2016), pp. 142–161.
[37] C. Rowley, I. Mezić, S. Bagheri, P. Schlatter, and D. Henningson, *Spectral analysis of nonlinear flows*, J. Fluid Mech., 641 (2009), pp. 115–127.
[38] E. Sachs and S. Volkwein, *Pod-galerkin approximations in pde-constrained optimization*, GAMM-Mitteilungen, 33 (2010), pp. 194–208.
[39] S. Sargsyan, S. L. Brunton, and J. N. Kutz, *Nonlinear model reduction for dynamical systems using sparse sensor locations from learned libraries*, Phys. Rev. E, 92 (2015), 033304.
[40] P. Schmid, *Dynamic mode decomposition of numerical and experimental data*, J. Fluid Mech., 656 (2010), pp. 5–28.
[41] P. Schmid and J. Sesterhenn, *Dynamic mode decomposition of numerical and experimental data*, J. Fluid Mech., 656 (2010), pp. 5–28, https://doi.org/10.1017/S0022112010001217.
[42] V. Simoncini, D. B. Szyld, and M. Monsalve, *On two numerical methods for the solution of large-scale algebraic Riccati equations*, IMA J. Numer. Anal., 34 (2013), pp. 904–920.
[43] J. R. Singler and B. Kramer, *A POD projection method for large-scale algebraic Riccati equations*, Numer. Algebra, Control Optim., 6 (2016), pp. 413–435, https://doi.org/10.3934/naco.2016018.
[44] L. Sirovich, *Turbulence and the dynamics of coherent structures. I. Coherent structures*, Quart. Appl. Math., 45 (1987), pp. 561–571.

[45] G. Tissot, L. Cordier, and B. R. Noack, *Feedback stabilization of an oscillating vertical cylinder by POD reduced-order model*, in J. Physics: Conference Series, 574 (2015), 012137.

[46] J. H. Tu, C. W. Rowley, D. M. Luchtenburg, S. L. Brunton, and J. N. Kutz, *On dynamic mode decomposition: Theory and applications*, J. Comput. Dyn., 1 (2014), pp. 391–421.

[47] S. Volkwein, *Proper Orthogonal Decomposition: Theory and Reduced-Order Modelling*, lecture notes, University of Konstanz, (2013), http://www.math.uni-konstanz.de/numerik/personen/volkwein/teaching/scripts.php.

[48] F. Wang and V. Balakrishnan, *Improved stability analysis and gain-scheduled controller synthesis for parameter-dependent systems*, IEEE Trans. Automat. Control, 47 (2002), pp. 720–734.

[49] X. Wu, V. Kumar, J. R. Quinlan, J. Ghosh, Q. Yang, H. Motoda, G. J. McLachlan, A. Ng, B. Liu, S. Y. Philip, Z. Zhou, M. Steinbach, D. Hand, and D. Steinberg, *Top* 10 *algorithms in data mining*, Knowledge Informat. Syst., 14 (2008), pp. 1–37.