Dynamic Data-Driven Reduced-Order Models

Benjamin Peherstorfer[a,*], Karen Willcox[a]

[a]*Department of Aeronautics & Astronautics, MIT, 77 Massachusetts Avenue, Cambridge, MA 02139, USA*

Abstract

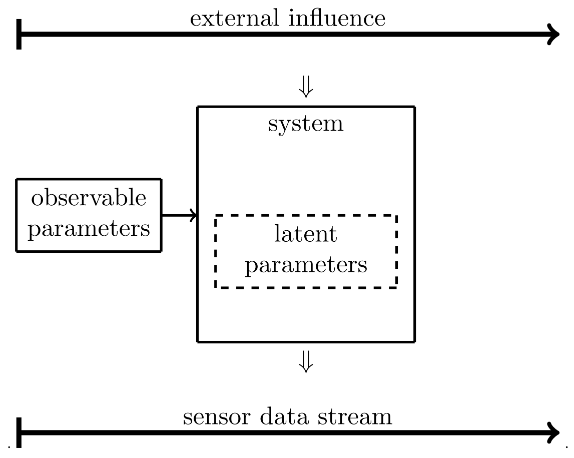
Data-driven model reduction constructs reduced-order models of large-scale systems by learning the system response characteristics from data. Existing methods build the reduced-order models in a computationally expensive offline phase and then use them in an online phase to provide fast predictions of the system. In cases where the underlying system properties are not static but undergo dynamic changes, repeating the offline phase after each system change to rebuild the reduced-order model from scratch forfeits the savings gained in the online phase. This paper proposes dynamic reduced-order models that break with this classical but rigid approach. Dynamic reduced-order models exploit the opportunity presented by dynamic sensor data and adaptively incorporate sensor data during the online phase. This permits online adaptation to system changes while circumventing the expensive rebuilding of the model. A computationally cheap adaptation is achieved by constructing low-rank updates to the reduced operators. With these updates and with sufficient and accurate data, our approach recovers the same model that would be obtained by rebuilding from scratch. We demonstrate dynamic reduced-order models on a structural assessment example in the context of real-time decision making. We consider a plate in bending where the dynamic reduced-order model quickly adapts to changes in structural properties and achieves speedups of four orders of magnitude compared to rebuilding a model from scratch.

Keywords: model reduction, online adaptivity, dynamic data-driven application systems, proper orthogonal decomposition

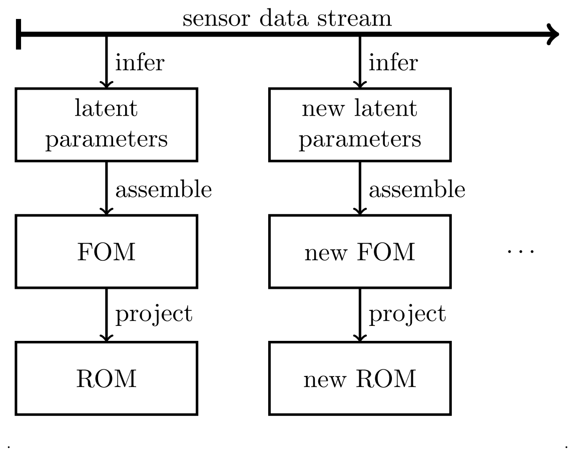
1. Introduction

We consider computational methods for dynamic data-driven decision making with a focus on problems for which the dynamics of the underlying system are modeled by parametrized partial differential equations (PDEs) and dynamic sensor data provides additional information regarding the current state of the system. In such a setting, the involved models and their corresponding computational solution methods must meet two particular requirements. First, the decision has to be made quickly (in real or near real time) and thus estimates and predictions that support this decision must be provided rapidly. Second, the underlying system may undergo changes in its properties, to which the model and solution methods must adapt. Again, this adaptation must be achieved rapidly. We address the real-time constraint by employing projection-based and data-driven model reduction to derive a computationally cheap reduced-order model (ROM) of the more expensive PDE discretization, referred to as the full-order model (FOM), of the system; however, in the case of system changes, classical model reduction techniques do not permit direct adaptation of the ROM but instead require a computationally costly rebuilding from scratch. To address this limitation, we develop dynamic data-driven ROMs that do not need to be rebuilt, but instead directly adapt to changes in the underlying system, using only the information provided by the sensors to drive the adaptation. Since the adaptation is achieved without recourse to the computationally expensive FOM, it can be achieved sufficiently rapidly to support online decision-making.

[*]Corresponding author
Email addresses: peħ (Benjamin Peherstorfer), +1-617-253-7831 (Benjamin Peherstorfer)



(a) system with latent parameters



(b) classical model reduction rebuilds ROMs from scratch

Figure 1: The sketch in (a) shows a system depending on observable and latent parameters. The observable parameters are given as inputs. The latent parameters describe changes in the system itself and cannot be controlled. Adapting ROMs to such system changes with classical model reduction techniques requires that we first infer the latent parameters from sensor data, then assemble the FOM, and finally rebuild the ROM from scratch, see (b).

One class of applications in the context of dynamic data-driven decision making is structural assessment. In these settings, the structure of a system is monitored by sensors. ROMs may be derived to predict the behavior of the system in response to different loading and operating conditions. During operation of the system, changes in the properties of the structure (e.g., due to sudden events, degradation or fatigue) can lead to different response characteristics. For the ROMs to be of continued use, it is therefore necessary to adapt them using the sensor data. One specific example is onboard structural health monitoring and structural assessment of aerospace vehicles, a field in which new sensor technologies offer significant opportunities. For example, future sensing technologies may include a “sensor skin”, providing strain and deflection data over the entire wing. The advent of low-cost high-resolution sensors will make feasible the concept of a self-aware aerospace vehicle—a vehicle that can dynamically adapt the way it performs missions by gathering information about itself and its surroundings and responding intelligently[1, 2]. In this paper we develop the algorithms that would make use of such data to enable online adaptation of structural response ROMs.

We model system changes with latent parameters and inputs to the system with observable parameters, see Figure 1a. The latent parameters describe, e.g., damage, erosion, and fatigue of the system and cannot be controlled. Except for an initial state that represents nominal system parameters, these latent parameters cannot be observed directly but only inferred from the sensor data with a model of the changed underlying system. Therefore, in classical projection-based and data-driven model reduction, adapting the ROM would require us to first infer the latent parameter from the data, then to assemble the FOM, and, finally, to rebuild the ROM from scratch, see Figure 1b. This is usually too expensive in the context of real-time decision making. In contrast, our dynamic ROM approach avoids expensive computations in the online phase by building on the following two key novel ideas. First, we completely avoid the FOM corresponding to the changed latent parameter by directly learning the reduced operators from the data. This is visualized in Figure 2. Second, we successively adapt the reduced operators with additive low-rank updates. The rank of the update depends on how much data are available. This guarantees valid updates if only a few data points are available, and, in the absence of sensor noise, it guarantees eventual recovery of the true ROM that we would obtain if we rebuilt the ROM from scratch. The computational cost of adapting the ROM to one newly received set of sensor measurements scales only linearly with the dimension of the FOM, provided the full-order operators for specific initial parameter configurations are sparse. Recall that we consider FOMs based on PDEs where this is often the case.

Recently, adaptation of ROMs has attracted much attention. A common technique in parametric model reduction is to interpolate between ROMs to adapt the model to the current parameter without assembling the full-order matrices [3, 4, 5]. In localization approaches, multiple ROMs are built offline and one of them is selected online depending on the current state of the system. The localization can be performed with respect to the parameter domain [6, 7, 8] or the state space [9, 10]. Also the spatial domain can be decomposed as shown in [11]. There are also dictionary approaches [12, 13], which pre-compute offline many basis vectors and then adaptively select several of them online. However, all of these approaches have in common that no new information in the form of data is incorporated and that all changes to the ROM are already anticipated in the offline phase through pre-computed quantities. In [14], the accuracy of local ROMs is improved by updating them after they have been selected in the online phase. A reference state is subtracted from the snapshots corresponding to each newly selected local ROM with the reference state depending on the previously selected local ROM. Thus, this update uses information that becomes available in the online phase; however, subtracting the reference state is also only a limited form of adaptation because, for example, each snapshot receives the same change. The approach in [14] has been recently extended in [15] to allow updates from partial data. Another online adaptive model reduction approach is presented in [16]. An unsupervised learning method is used to split the basis vectors depending on residual information. In [17], ROMs are adapted online during an iterative optimization procedure. Updates to the basis vectors of the ROMs are computed from combinations of snapshots, reduced solutions, and adjoint information. Besides these adaptive methods, there has been an interest in using a data assimilation framework to calibrate ROMs to experimental data [18, 19]. In contrast to our problem setting, however, the goal of data assimilation is to account for the model bias rather than to adapt the ROM to a changed system. Another related approach is Kalman filtering [20], which combines measurement data and a state transition model to derive a better estimate of the state vector than obtained by using either the data or the model. It was made computationally feasible for the often high-dimensional state vectors stemming from the discretization of PDEs by the ensemble Kalman filter [21, 22, 23]. Whereas Kalman filtering primarily focuses on correcting and estimating the state vector, and possibly the corresponding quantity of interest, dynamic ROMs adapt to changes in the latent parameters by identifying and applying low-rank updates to the reduced system operators. Thus, even though there are several adaptive model reduction techniques available, our approach is different because we do not anticipate offline how the FOM or the ROM change during the online phase, and we incorporate new information in the form of sensor data for the update.

The following Section 2 introduces discrete systems of equations stemming from PDEs with latent parameters and derives the corresponding ROMs based on proper orthogonal decomposition (POD). We then give a detailed problem formulation and problem setting of adapting ROMs online from sensor data. Section 3 discusses adapting the reduced basis and the reduced operators of our dynamic ROMs and then combines them into an adaptivity procedure. We demonstrate our dynamic ROMs with numerical examples of a structural assessment example based on the Mindlin plate theory in Section 4. Section 5 concludes the paper.

2. Reduced-order models of systems with latent parameters

We consider FOMs based on PDEs with observable parameters, which are given as inputs during the online phase, and latent parameters, which describe changes in the modeled system and cannot be controlled or directly observed. Section 2.1 formalizes these FOMs in the context of real-time decision making and Section 2.2 derives ROMs based on POD. Section 2.3 then discusses the need to adapt the ROM online due to the changing latent parameters and presents our specific problem formulation.

2.1. Parametrized systems with latent parameters

We consider a model based on a parametrized PDE. Our starting point is the system of equations

$$\boldsymbol{A}_{\boldsymbol{\eta}}(\boldsymbol{\mu})\boldsymbol{y}_{\boldsymbol{\eta}}(\boldsymbol{\mu}) = \boldsymbol{f}(\boldsymbol{\mu}) \tag{1}$$

with $\mathcal{N} \in \mathbb{N}$ degrees of freedom stemming from the discretization of the PDE. System (1) depends on the observable parameter $\boldsymbol{\mu} = [\mu_1, \dots, \mu_d]^T \in \mathcal{D}$ with $d \in \mathbb{N}$ components and the latent parameter $\boldsymbol{\eta} = [\eta_1, \dots, \eta_{d'}]^T \in \mathcal{E}$ with $d' \in \mathbb{N}$ components. The parameter domains $\mathcal{D}$ and $\mathcal{E}$ are subsets of $\mathbb{R}^d$ and $\mathbb{R}^{d'}$, respectively. We have the operator $\boldsymbol{A}_{\boldsymbol{\eta}}(\boldsymbol{\mu}) \in \mathbb{R}^{\mathcal{N} \times \mathcal{N}}$, the solution (state vector) $\boldsymbol{y}_{\boldsymbol{\eta}}(\boldsymbol{\mu}) \in \mathbb{R}^{\mathcal{N}}$, and the

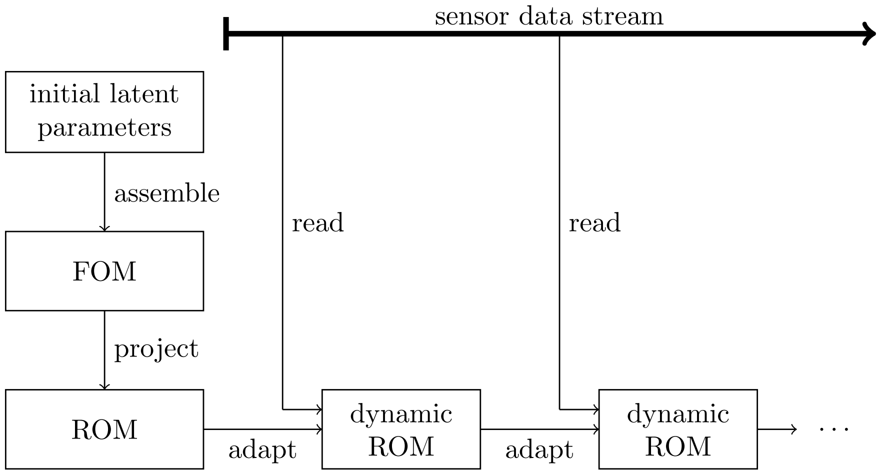


Figure 2: Our dynamic ROMs are informed about changes in the underlying system by sensor data. They then adapt to these changes without recourse to the FOM and without inference of the latent parameter.

right-hand side $\boldsymbol{f}(\boldsymbol{\mu}) \in \mathbb{R}^{\mathcal{N}}$. The operator and the solution vector depend on the observable and the latent parameter. The following dynamic ROM approach is limited to the case where the right-hand side depends on the observable parameter but is independent of the latent parameter, see Section 3.2.3. A dependence on the latent parameter $\boldsymbol{\eta}$ is denoted as subscript to indicate that it cannot be controlled and that its value is in general unknown, except for an initial parameter $\boldsymbol{\eta}_0 \in \mathcal{E}$ describing the initial state of the underlying system. This clearly distinguishes the latent parameter from the observable parameter $\boldsymbol{\mu}$ which is given as input, see Figure 1a. This is highlighted by denoting the observable parameter in parentheses.

We assume the operator $\boldsymbol{A}_{\boldsymbol{\eta}}(\boldsymbol{\mu})$ can be represented with an affine parameter dependence with respect to the observable parameter $\boldsymbol{\mu}$. Thus, it can be represented as a linear combination

$$\boldsymbol{A}_{\boldsymbol{\eta}}(\boldsymbol{\mu}) = \sum_{i=1}^{l_A} \theta_A^{(i)}(\boldsymbol{\mu}) \boldsymbol{A}_{\boldsymbol{\eta}}^{(i)} \tag{2}$$

of $\boldsymbol{\mu}$-independent operators

$$\boldsymbol{A}_{\boldsymbol{\eta}}^{(1)}, \dots, \boldsymbol{A}_{\boldsymbol{\eta}}^{(l_A)} \in \mathbb{R}^{\mathcal{N} \times \mathcal{N}} \tag{3}$$

with $l_A \in \mathbb{N}$ functions $\theta_A^{(1)}, \dots, \theta_A^{(l_A)} : \mathcal{D} \to \mathbb{R}$. The operators (3) might depend nonlinearly on $\boldsymbol{\eta}$. No affine parameter dependence of $\boldsymbol{A}_{\boldsymbol{\eta}}(\boldsymbol{\mu})$ with respect to the latent parameter $\boldsymbol{\eta}$ is required. Similarly to (2), we assume an affine parameter dependence of the right-hand side $\boldsymbol{f}(\boldsymbol{\mu})$ with respect to the observable parameter $\boldsymbol{\mu}$, i.e.,

$$\boldsymbol{f}(\boldsymbol{\mu}) = \sum_{i=1}^{l_f} \theta_f^{(i)}(\boldsymbol{\mu}) \boldsymbol{f}^{(i)} , \tag{4}$$

with $l_f \in \mathbb{N}$ functions $\theta_f^{(1)}, \dots \theta_f^{(l_f)} : \mathcal{D} \to \mathbb{R}$ and $\boldsymbol{\mu}$-independent vectors $\boldsymbol{f}^{(1)}, \dots, \boldsymbol{f}^{(l_f)} \in \mathbb{R}^{\mathcal{N}}$. The right-hand side $\boldsymbol{f}(\boldsymbol{\mu})$ does not depend on $\boldsymbol{\eta}$. We note that if an affine decomposition of $\boldsymbol{A}_{\boldsymbol{\eta}}(\boldsymbol{\mu})$ or $\boldsymbol{f}(\boldsymbol{\mu})$ is not admitted directly by the problem formulation, it can be constructed approximately by, e.g., gappy POD [24, 25] or empirical interpolation [26, 27].

2.2. Reduced-order models of systems with latent parameters

Let

$$\boldsymbol{Y}_{\boldsymbol{\eta}_0} = [\boldsymbol{y}_{\boldsymbol{\eta}_0}(\boldsymbol{\mu}_1), \dots, \boldsymbol{y}_{\boldsymbol{\eta}_0}(\boldsymbol{\mu}_m)] \in \mathbb{R}^{\mathcal{N} \times m} \tag{5}$$

be the snapshot matrix that contains $m \in \mathbb{N}$ linearly independent solution vectors of (1) with observable parameters $\boldsymbol{\mu}_1, \dots, \boldsymbol{\mu}_m \in \mathcal{D}$ and the initial latent parameter $\boldsymbol{\eta}_0$. These solutions are called snapshots. We do not consider here how to best sample the FOM but refer to, e.g., [28, 29, 30, 31]. POD is a method to construct an n-dimensional basis $\boldsymbol{v}_1, \dots, \boldsymbol{v}_n \in \mathbb{R}^{\mathcal{N}}$ such that the snapshots (5) are optimally represented by their orthogonal projections onto the subspace $\operatorname{span}\{\boldsymbol{v}_1, \dots, \boldsymbol{v}_n\} \subset \operatorname{span}\{\boldsymbol{y}_{\boldsymbol{\eta}_0}(\boldsymbol{\mu}_1), \dots, \boldsymbol{y}_{\boldsymbol{\eta}_0}(\boldsymbol{\mu}_m)\} \subset \mathbb{R}^{\mathcal{N}}$.

The POD basis vectors $\boldsymbol{v}_1, \dots, \boldsymbol{v}_n \in \mathbb{R}^{\mathcal{N}}$ are the left-singular vectors corresponding to the n largest singular values of the snapshot matrix (5). Hence, to compute the POD basis for the snapshots in (5), we first compute the singular value decomposition (SVD) of the snapshot matrix $\boldsymbol{Y}_{\boldsymbol{\eta}_0}$. We then order the singular values non-ascending. The first largest n singular values form the diagonal of the diagonal matrix $\boldsymbol{\Sigma}_{\boldsymbol{\eta}_0} \in \mathbb{R}^{n \times n}$, and the n left- and right-singular vectors, corresponding to the n largest singular values, are the columns in the matrices $\boldsymbol{V}_{\boldsymbol{\eta}_0} = [\boldsymbol{v}_1, \dots, \boldsymbol{v}_n] \in \mathbb{R}^{\mathcal{N} \times n}$ and $\boldsymbol{W}_{\boldsymbol{\eta}_0} = [\boldsymbol{w}_1, \dots, \boldsymbol{w}_n] \in \mathbb{R}^{\mathcal{N} \times n}$, respectively. We derive a ROM of the FOM (1) for the initial latent parameter $\boldsymbol{\eta}_0$ by constructing the $\boldsymbol{\mu}$-independent reduced operators

$$\tilde{\boldsymbol{A}}_{\boldsymbol{\eta}_0}^{(i)} = \boldsymbol{V}_{\boldsymbol{\eta}_0}^T \boldsymbol{A}_{\boldsymbol{\eta}_0}^{(i)} \boldsymbol{V}_{\boldsymbol{\eta}_0}\,, \qquad i = 1, \dots, l_A\,, \tag{6}$$

and the $\boldsymbol{\mu}$-independent reduced right-hand sides

$$\tilde{\boldsymbol{f}}_{\boldsymbol{\eta}_0}^{(i)} = \boldsymbol{V}_{\boldsymbol{\eta}_0}^T \boldsymbol{f}^{(i)}\,, \qquad i = 1, \dots, l_f\,, \tag{7}$$

with the POD basis $\boldsymbol{V}_{\boldsymbol{\eta}_0}$ with Galerkin projection. In contrast to the FOM, the reduced right-hand side (7) depends through the POD basis $\boldsymbol{V}_{\boldsymbol{\eta}_0}$ on the latent parameter $\boldsymbol{\eta}_0$. The reduced system for observable parameter $\boldsymbol{\mu} \in \mathcal{D}$ and initial latent parameter $\boldsymbol{\eta}_0$ is then given as

$$\tilde{\boldsymbol{A}}_{\boldsymbol{\eta}_0}(\boldsymbol{\mu}) \tilde{\boldsymbol{y}}_{\boldsymbol{\eta}_0}(\boldsymbol{\mu}) = \sum_{i=1}^{l_A} \theta_A^{(i)}(\boldsymbol{\mu}) \tilde{\boldsymbol{A}}_{\boldsymbol{\eta}_0}^{(i)} \tilde{\boldsymbol{y}}_{\boldsymbol{\eta}_0}(\boldsymbol{\mu}) = \sum_{i=1}^{l_f} \theta_f^{(i)}(\boldsymbol{\mu}) \tilde{\boldsymbol{f}}_{\boldsymbol{\eta}_0}^{(i)} = \tilde{\boldsymbol{f}}_{\boldsymbol{\eta}_0}(\boldsymbol{\mu}) \tag{8}$$

with the reduced operator $\tilde{\boldsymbol{A}}_{\boldsymbol{\eta}_0}(\boldsymbol{\mu}) \in \mathbb{R}^{n \times n}$, the reduced right-hand side $\tilde{\boldsymbol{f}}_{\boldsymbol{\eta}_0}(\boldsymbol{\mu}) \in \mathbb{R}^n$, and the reduced state vector $\tilde{\boldsymbol{y}}_{\boldsymbol{\eta}_0}(\boldsymbol{\mu}) \in \mathbb{R}^n$. Note that we invoked in (8) the affine parameter dependence with respect to the observable parameter $\boldsymbol{\mu}$ as defined in (2) and (4). Evaluating (8) instead of (1) can lead to computational savings because often the number of degrees of freedom n of the ROM can be chosen much smaller than the number of degrees of freedom $\mathcal{N}$ of the FOM while maintaining acceptable accuracy of the solution estimates.

2.3. Problem formulation and problem setting

Let us consider the ROM based on the POD basis $\boldsymbol{V}_{\boldsymbol{\eta}_0}$, and the reduced operator $\tilde{\boldsymbol{A}}_{\boldsymbol{\eta}_0}(\boldsymbol{\mu})$ and right-hand side $\tilde{\boldsymbol{f}}_{\boldsymbol{\eta}_0}(\boldsymbol{\mu})$ as defined in (8). The ROM was built for the initial latent parameter $\boldsymbol{\eta}_0$ and thus captures the behavior of the FOM only for $\boldsymbol{\eta}_0$; however, we consider the case where in the online phase, the latent parameter changes from $\boldsymbol{\eta}_0$ to an unknown value $\boldsymbol{\eta}' \in \mathcal{E}$ due to a system change. Hence, the ROM becomes obsolete and has to be adapted to parameter $\boldsymbol{\eta}'$. Adapting the ROM requires that we adapt the basis $\boldsymbol{V}_{\boldsymbol{\eta}_0}$ as well as the reduced operator $\tilde{\boldsymbol{A}}_{\boldsymbol{\eta}_0}(\boldsymbol{\mu})$ and the reduced right-hand side $\tilde{\boldsymbol{f}}_{\boldsymbol{\eta}_0}(\boldsymbol{\mu})$.

In the following, we successively adapt the ROM in $h = 1, \dots, m'$ adaptivity steps during the online phase where $m' \in \mathbb{N}$. At each step h, we receive data in the form of a so-called sensor sample $\hat{\boldsymbol{y}}_{\boldsymbol{\eta}'}(\boldsymbol{\mu}_{m+h}) \in \mathbb{R}^{\mathcal{N}}$, which is a sensed measurement of the full-order state vector for an observable parameter $\boldsymbol{\mu}_{m+h} \in \mathcal{D}$ and latent parameter $\boldsymbol{\eta}' \in \mathcal{E}$. Note that our setup considers a sensed measurement of the full-order state vector. As discussed in Section 1, new sensor technologies are already making this possible, especially on a component level. We further note that, even though we consider full-order state information to be available through sensor measurements, the ROM is still necessary in order to give us a predictive capability. In particular, since we cannot control the observable parameter $\boldsymbol{\mu}_{m+h}$, a ROM is needed to provide approximations of the full-order state vector for different observable parameters than $\boldsymbol{\mu}_{m+h}$. This predictive capability is particularly relevant for online planning and decision making scenarios, where one is interested in predicting the system behavior for different operating conditions (e.g., different load, velocity). This requires a model that can be evaluated at parameters corresponding to the operating conditions of interest.

The difference $\hat{\boldsymbol{y}}_{\boldsymbol{\eta}'}(\boldsymbol{\mu}_{m+h}) - \boldsymbol{y}_{\boldsymbol{\eta}'}(\boldsymbol{\mu}_{m+h}) \in \mathbb{R}^{\mathcal{N}}$ between the sensor sample, $\hat{\boldsymbol{y}}_{\boldsymbol{\eta}'}(\boldsymbol{\mu}_{m+h})$, and the solution of the FOM corresponding to these parameters, $\boldsymbol{y}_{\boldsymbol{\eta}'}(\boldsymbol{\mu}_{m+h})$, is measurement noise (and potentially also FOM error relative to reality, although here we assume that the FOM is our "truth" model). The sensor sample matrix at step h,

$$\boldsymbol{S}_h = [\hat{\boldsymbol{y}}_{\boldsymbol{\eta}'}(\boldsymbol{\mu}_{m+1}), \dots, \hat{\boldsymbol{y}}_{\boldsymbol{\eta}'}(\boldsymbol{\mu}_{m+h})] \in \mathbb{R}^{\mathcal{N} \times h}\,, \tag{9}$$

is assembled from h linearly independent sensor samples with observable parameters $\boldsymbol{\mu}_{m+1}, \dots, \boldsymbol{\mu}_{m+m'} \in \mathcal{D}$. Note that the linear independence of the sensor samples can be achieved by reading, at step h, sensor data until a sensor sample is received that is linearly independent with respect to all previous $h-1$ sensor samples, and then using this sensor sample to extend the sensor sample matrix (9). We consider the case where we do not have access to the FOM of the changed system—in particular, we cannot assemble the full-order matrices for latent parameter $\boldsymbol{\eta}'$ because it is too costly for the online phase. Our goal then is to adapt the ROM using only the information provided by the sensor data.

3. Dynamic reduced-order models

A dynamic ROM update consists of adapting the POD basis and adapting the reduced operators. Section 3.1 presents an SVD procedure that adapts the POD basis from a snapshot matrix updated with the sensor samples. Section 3.2 derives low-rank additive updates for the reduced operators using new information from the sensor samples. The POD basis and the reduced operator update procedures are combined into the dynamic ROM approach in Section 3.3. Section 3.4 discusses computational costs. For the sake of exposition we only present the theory for the case where the latent parameter changes once, i.e., from $\boldsymbol{\eta}_0$ to $\boldsymbol{\eta}'$. It is straightforward to extend the following approach to multiple changes of the latent parameter, as demonstrated in the results.

3.1. Adapting the POD basis

To initialize the adaptivity, let $\boldsymbol{Y}_0 = \boldsymbol{Y}_{\boldsymbol{\eta}_0}$ be the snapshot matrix containing as columns the snapshots $\boldsymbol{y}_{\boldsymbol{\eta}_0}(\boldsymbol{\mu}_1), \dots, \boldsymbol{y}_{\boldsymbol{\eta}_0}(\boldsymbol{\mu}_m) \in \mathbb{R}^{\mathcal{N}}$ with observable parameters $\boldsymbol{\mu}_1, \dots, \boldsymbol{\mu}_m \in \mathcal{D}$ and initial latent parameter $\boldsymbol{\eta}_0$, and let $\boldsymbol{V}_0 = \boldsymbol{V}_{\boldsymbol{\eta}_0} \in \mathbb{R}^{\mathcal{N} \times n}$ be the POD basis computed from these snapshots in the offline phase. At the first adaptivity step $h = 1$, we receive the sensor sample $\hat{\boldsymbol{y}}_{\boldsymbol{\eta}'}(\boldsymbol{\mu}_{m+1})$. We replace the snapshot $\boldsymbol{y}_{\boldsymbol{\eta}_0}(\boldsymbol{\mu}_1)$ in the snapshot matrix $\boldsymbol{Y}_0$ with this new sensor sample and denote the new snapshot matrix by $\boldsymbol{Y}_1$. We continue this process and so receive at step h the sensor sample $\hat{\boldsymbol{y}}_{\boldsymbol{\eta}'}(\boldsymbol{\mu}_{m+h})$, using it to replace $\boldsymbol{y}_{\boldsymbol{\eta}_0}(\boldsymbol{\mu}_h)$ to obtain the updated snapshot matrix

$$\boldsymbol{Y}_h = [\underbrace{\hat{\boldsymbol{y}}_{\boldsymbol{\eta}'}(\boldsymbol{\mu}_{m+1}), \dots, \hat{\boldsymbol{y}}_{\boldsymbol{\eta}'}(\boldsymbol{\mu}_{m+h})}_{\text{sensor samples with } \boldsymbol{\eta}'}, \underbrace{\boldsymbol{y}_{\boldsymbol{\eta}_0}(\boldsymbol{\mu}_{h+1}), \dots, \boldsymbol{y}_{\boldsymbol{\eta}_0}(\boldsymbol{\mu}_m)}_{\text{snapshots with } \boldsymbol{\eta}_0}] \in \mathbb{R}^{\mathcal{N} \times m}\,. \tag{10}$$

The snapshot matrix (10) contains the sensor samples up to the h-th column and the snapshots from the offline phase with $\boldsymbol{\eta} = \boldsymbol{\eta}_0$ in columns $h+1$ to m. Note that snapshots in the snapshot matrix are replaced and not added, i.e., the number of columns m does not change. Note further that the columns are replaced following the first-in-first-out principle if h becomes larger than m.

At each adaptivity step the POD basis has to be adapted to the updated snapshot matrix. We now derive an algorithm to compute the adapted POD basis $\boldsymbol{V}_h$ for the updated snapshot matrix $\boldsymbol{Y}_h$. We consider $\boldsymbol{Y}_h$ to be the result of a rank-one update to the snapshot matrix from the previous adaptivity step $\boldsymbol{Y}_{h-1}$, i.e.,

$$\boldsymbol{Y}_h = \boldsymbol{Y}_{h-1} + \boldsymbol{a}\boldsymbol{e}_h^T\,, \tag{11}$$

where $\boldsymbol{a} = \hat{\boldsymbol{y}}_{\boldsymbol{\eta}'}(\boldsymbol{\mu}_{m+h}) - \boldsymbol{y}_{\boldsymbol{\eta}_0}(\boldsymbol{\mu}_h) \in \mathbb{R}^{\mathcal{N}}$ and $\boldsymbol{e}_h \in \mathbb{R}^m$ is the h-th canonical unit vector with 1 at the h-th component and 0 elsewhere. The unit vector $\boldsymbol{e}_h$ indicates that we replace the h-th column of $\boldsymbol{Y}_{h-1}$. Because we can represent $\boldsymbol{Y}_h$ as a rank-one update to $\boldsymbol{Y}_{h-1}$, the SVD updating algorithm introduced in [32] is applicable. Note that this is the same algorithm as used in [14]; however, we successively exchange snapshots, whereas the purpose of the update in [14] is to identify a new reference state that is subtracted

Algorithm 1 Adapts the POD basis after rank-one update to snapshot matrix

```
1:  procedure ADAPTBASIS($\boldsymbol{V}_{h-1}, \boldsymbol{\Sigma}_{h-1}, \boldsymbol{W}_{h-1}, \boldsymbol{a}, \boldsymbol{e}_h$)
2:      Extract component of $\boldsymbol{a}$ that is orthogonal to $\boldsymbol{V}_{h-1}$ with $\boldsymbol{\alpha} = \boldsymbol{a} - \boldsymbol{V}_{h-1}\boldsymbol{V}_{h-1}^T\boldsymbol{a}$
3:      Extract component of $\boldsymbol{e}_h$ that is orthogonal to $\boldsymbol{W}_{h-1}$ with $\boldsymbol{\beta} = \boldsymbol{e}_h - \boldsymbol{W}_{h-1}\boldsymbol{W}_{h-1}^T\boldsymbol{e}_h$
4:      Assemble the $(n+1) \times (n+1)$ matrix
            $$\boldsymbol{K} = \begin{bmatrix} \boldsymbol{\Sigma}_{h-1} & \boldsymbol{1} \\ \boldsymbol{0} & 0 \end{bmatrix} + \begin{bmatrix} \boldsymbol{V}_{h-1}^T\boldsymbol{a} \\ \|\boldsymbol{\alpha}\|_2 \end{bmatrix} \begin{bmatrix} \boldsymbol{e}^T\boldsymbol{W}_{h-1} & \|\boldsymbol{\beta}\|_2 \end{bmatrix}$$
5:      Compute SVD $[\overline{\boldsymbol{V}}_h, \overline{\boldsymbol{\Sigma}}_h, \overline{\boldsymbol{W}}_h]$ =SVD($\boldsymbol{K}$)
6:      Normalize $\boldsymbol{\alpha} = \boldsymbol{\alpha}/\|\boldsymbol{\alpha}\|_2$
7:      Normalize $\boldsymbol{\beta} = \boldsymbol{\beta}/\|\boldsymbol{\beta}\|_2$
8:      Extract rotation of $\overline{\boldsymbol{V}}_h$ with $\boldsymbol{V}_h' = \overline{\boldsymbol{V}}_h(1:n, 1:n)$
9:      Extract additive update of $\overline{\boldsymbol{V}}_h$ with $\boldsymbol{q} = \overline{\boldsymbol{V}}_h(n+1, 1:n)^T$
10:     Store $\boldsymbol{p} = \boldsymbol{\alpha}$ for additive update
11:     Extract first $n$ singular values of $\overline{\boldsymbol{\Sigma}}_h$ with $\boldsymbol{\Sigma}_h = \overline{\boldsymbol{\Sigma}}_h(1:n, 1:n)$
12:     Update right-singular vectors to $\boldsymbol{W}_h = \begin{bmatrix} \boldsymbol{W}_{h-1} & \boldsymbol{\beta} \end{bmatrix} \overline{\boldsymbol{W}}_h$
13:     Extract first $n$ right-singular vectors with $\boldsymbol{W}_h = \boldsymbol{W}_h(:, 1:n)$
14: return $[\boldsymbol{V}_h', \boldsymbol{\Sigma}_h, \boldsymbol{W}_h, \boldsymbol{p}, \boldsymbol{q}]$
15: end procedure
```

from all snapshots. The algorithm reuses the adapted SVD of $\boldsymbol{Y}_{h-1}$ to approximately derive the POD basis corresponding to $\boldsymbol{Y}_h$. For that, it is only necessary to compute the SVD of a matrix with size $(n+1)\times(n+1)$ (where $n \ll \mathcal{N}$ is the dimension of the reduced state) instead of the original snapshot matrix $\boldsymbol{Y}_h$ with size $\mathcal{N} \times m$.

The SVD updating method of [32] is summarized in Algorithm 1. The input arguments $\boldsymbol{V}_{h-1}, \boldsymbol{W}_{h-1}$, and $\boldsymbol{\Sigma}_{h-1}$ are the adapted SVD matrices computed on the previous adaptivity step of the snapshot matrix $\boldsymbol{Y}_{h-1}$ and the vectors $\boldsymbol{a} = \hat{\boldsymbol{y}}_{\boldsymbol{\eta}'}(\boldsymbol{\mu}_{m+h}) - \boldsymbol{y}_{\boldsymbol{\eta}_0}(\boldsymbol{\mu}_h) \in \mathbb{R}^{\mathcal{N}}$ and $\boldsymbol{e}_h \in \mathbb{R}^m$ describe the rank-one update (11). The algorithm first extracts the component of $\boldsymbol{a}$ that is orthogonal to the POD basis $\boldsymbol{V}_{h-1}$ and stores it in $\boldsymbol{\alpha} \in \mathbb{R}^{\mathcal{N}}$. The vector $\boldsymbol{\alpha}$ contains the new information that is introduced by the sensor sample $\hat{\boldsymbol{y}}_{\boldsymbol{\eta}'}(\boldsymbol{\mu}_{m+h})$ into the POD basis. Similarly, the component of $\boldsymbol{e}_h$ that is orthogonal to $\boldsymbol{W}_{h-1}$ is stored in $\boldsymbol{\beta} \in \mathbb{R}^m$. Then, the SVD of the matrix $\boldsymbol{K} \in \mathbb{R}^{n+1\times n+1}$ is computed. From this SVD, the rotation matrix $\boldsymbol{V}_h' \in \mathbb{R}^{n\times n}$ as well as the vectors $\boldsymbol{p} \in \mathbb{R}^{\mathcal{N}}$ and $\boldsymbol{q} \in \mathbb{R}^n$ are extracted as in Algorithm 1, which leads to the adapted POD basis

$$\boldsymbol{V}_h = \boldsymbol{V}_{h-1}\boldsymbol{V}_h' + \boldsymbol{p}\boldsymbol{q}^T \,. \tag{12}$$

In Algorithm 1 we make use of MATLAB's slicing notation by selecting the first n columns and the first n rows of $\overline{\boldsymbol{V}}_h \in \mathbb{R}^{n+1\times n+1}$ with $\overline{\boldsymbol{V}}_h(1:n, 1:n) \in \mathbb{R}^{n\times n}$. The adapted POD basis (12) is an approximation of the POD basis that we would obtain by recomputing the SVD of $\boldsymbol{Y}_h$ from scratch. The approximation error decreases with the dimension n of the ROM [32]. Note that Algorithm 1 and the update in (12) could easily be modified to permit a change in the dimension n of the ROM after each rank-one update to the snapshot matrix. We therefore would set the rotation matrix $\boldsymbol{V}_h'$ in line 8 of Algorithm 1 to $\overline{\boldsymbol{V}}_h(1:n, 1:n+1) \in \mathbb{R}^{n\times n+1}$ and the additive update $\boldsymbol{q}$ to $\overline{\boldsymbol{V}}_h(n+1, 1:n+1)^T \in \mathbb{R}^{n+1\times 1}$. This then would lead in (12) to a POD basis $\boldsymbol{V}_h$ with $n+1$ basis vectors, see [32] for details; however, we do not further pursue this option in the following.

3.2. Adapting the reduced operators with low-rank updates

We now adapt the reduced operators. We cannot directly construct the true reduced operators

$$\tilde{\boldsymbol{A}}_{\boldsymbol{\eta}'}^{(i)} = \boldsymbol{V}_h^T \boldsymbol{A}_{\boldsymbol{\eta}'}^{(i)} \boldsymbol{V}_h \,, \qquad i = 1, \ldots, l_A \,, \tag{13}$$

because the matrix-matrix products in (13) rely on the full-order matrices $\boldsymbol{A}_{\boldsymbol{\eta}'}^{(1)}, \dots, \boldsymbol{A}_{\boldsymbol{\eta}'}^{(l_A)} \in \mathbb{R}^{\mathcal{N} \times \mathcal{N}}$ for the changed parameter $\boldsymbol{\eta}'$. These full-order matrices are not available and inferring the latent parameter to assemble them is often too expensive in the online phase, see Section 2.3 and Figures 1-2. We therefore approximate (13) at step h by the adapted reduced operators

$$\tilde{\boldsymbol{A}}_h^{(i)} = \boldsymbol{V}_h^T \boldsymbol{A}_0^{(i)} \boldsymbol{V}_h + \delta \tilde{\boldsymbol{A}}_h^{(i)}\,, \qquad i = 1, \dots, l_A\,,$$

where $\boldsymbol{V}_h$ is the adapted POD basis given by (12), the operators

$$\boldsymbol{A}_0^{(i)} = \boldsymbol{A}_{\boldsymbol{\eta}_0}^{(i)}\,, \qquad i = 1, \dots, l_A\,,$$

are the full-order matrices with initial parameter $\boldsymbol{\eta}_0$, and

$$\delta \tilde{\boldsymbol{A}}_h^{(1)}, \dots, \delta \tilde{\boldsymbol{A}}_h^{(l_A)} \in \mathbb{R}^{n \times n} \tag{14}$$

are additive updates. Adapting the reduced operators requires that we first construct

$$\boldsymbol{V}_h^T \boldsymbol{A}_0^{(i)} \boldsymbol{V}_h\,, \qquad i = 1, \dots, l_A\,, \tag{15}$$

and then derive the additive updates (14). Since we adapt the ROM online, both steps must be conducted efficiently. We discuss these two steps in the Sections 3.2.1 and 3.2.2, and then summarize them in the computational procedure in Section 3.2.3.

3.2.1. Basis transformation

We exploit the structure $\boldsymbol{V}_h = \boldsymbol{V}_{h-1}\boldsymbol{V}_h' + \boldsymbol{p}\boldsymbol{q}^T$ of the adapted POD basis $\boldsymbol{V}_h$, cf. Section 3.1 and (12), to avoid the costly matrix-matrix product with the full-order matrices for the construction of (15). We represent (15) as

$$\begin{aligned}
\boldsymbol{V}_h^T \boldsymbol{A}_0^{(i)} \boldsymbol{V}_h &= \left(\boldsymbol{V}_{h-1}\boldsymbol{V}_h' + \boldsymbol{p}\boldsymbol{q}^T\right)^T \boldsymbol{A}_0^{(i)} \left(\boldsymbol{V}_{h-1}\boldsymbol{V}_h' + \boldsymbol{p}\boldsymbol{q}^T\right) \\
&= \boldsymbol{V}_h'^T \boldsymbol{V}_{h-1}^T \boldsymbol{A}_0^{(i)} \boldsymbol{V}_{h-1} \boldsymbol{V}_h' + \boldsymbol{q}\boldsymbol{p}^T \underbrace{\boldsymbol{A}_0^{(i)} \boldsymbol{V}_{h-1}}_{\boldsymbol{B}_{h-1}^{(i)}} \boldsymbol{V}_h' + \underbrace{\boldsymbol{V}_h'^T \boldsymbol{V}_{h-1}^T \boldsymbol{A}_0^{(i)} \boldsymbol{p}\boldsymbol{q}^T + \boldsymbol{q}\boldsymbol{p}^T \boldsymbol{A}_0^{(i)} \boldsymbol{p}\boldsymbol{q}^T}_{\boldsymbol{C}_h^{(i)} \boldsymbol{p}\boldsymbol{q}^T} \\
&= \underbrace{\boldsymbol{V}_h'^T}_{n \times n} \underbrace{\boldsymbol{V}_{h-1}^T \boldsymbol{A}_0^{(i)} \boldsymbol{V}_{h-1}}_{n \times n} \underbrace{\boldsymbol{V}_h'}_{n \times n} + \underbrace{\boldsymbol{q}\boldsymbol{p}^T}_{n \times \mathcal{N}} \underbrace{\boldsymbol{B}_{h-1}^{(i)}}_{\mathcal{N} \times n} \underbrace{\boldsymbol{V}_h'}_{n \times n} + \underbrace{\boldsymbol{C}_h^{(i)}}_{n \times \mathcal{N}} \underbrace{\boldsymbol{p}\boldsymbol{q}^T}_{\mathcal{N} \times n}
\end{aligned} \tag{16}$$

where we reuse the operator $\boldsymbol{V}_{h-1}^T \boldsymbol{A}_0^{(i)} \boldsymbol{V}_{h-1}$ of the previous adaptivity step $h-1$ and where $\boldsymbol{B}_{h-1}^{(i)} = \boldsymbol{A}_0^{(i)} \boldsymbol{V}_{h-1} \in \mathbb{R}^{\mathcal{N} \times n}$ and $\boldsymbol{C}_h^{(i)} = \boldsymbol{V}_h^T \boldsymbol{A}_0^{(i)} \in \mathbb{R}^{n \times \mathcal{N}}$ are auxiliary quantities for $i = 1, \dots, l_A$. Note that the computational complexity of all matrix-matrix products in (16) is linear in $\mathcal{N}$. The auxiliary quantities are constructed recursively following

$$\boldsymbol{B}_h^{(i)} = \boldsymbol{B}_{h-1}^{(i)} \boldsymbol{V}_h' + \boldsymbol{A}_0^{(i)} \boldsymbol{p}\boldsymbol{q}^T\,, \qquad i = 1, \dots, l_A\,, \tag{17}$$

$$\boldsymbol{C}_h^{(i)} = \boldsymbol{V}_h'^T \boldsymbol{C}_{h-1}^{(i)} + \boldsymbol{q}\boldsymbol{p}^T \boldsymbol{A}_0^{(i)}\,, \qquad i = 1, \dots, l_A\,, \tag{18}$$

where $\boldsymbol{B}_0^{(i)} = \boldsymbol{A}_0^{(i)} \boldsymbol{V}_0 \in \mathbb{R}^{\mathcal{N} \times n}$ and $\boldsymbol{C}_0^{(i)} = \boldsymbol{V}_0^T \boldsymbol{A}_0^{(i)} \in \mathbb{R}^{n \times \mathcal{N}}$ are pre-computed in the offline phase. Algorithm 2 summarizes the steps to compute (17) and (18).

3.2.2. Additive updates to the reduced operators

Recall that $\hat{\boldsymbol{y}}_{\boldsymbol{\eta}'}(\boldsymbol{\mu}_{m+1}), \dots, \hat{\boldsymbol{y}}_{\boldsymbol{\eta}'}(\boldsymbol{\mu}_{m+h}) \in \mathbb{R}^{\mathcal{N}}$ are the sensor samples that have been received at adaptivity step h. We compute the additive updates (14) by solving the minimization problem

$$\min_{\delta \tilde{\boldsymbol{A}}_h^{(1)}, \dots, \delta \tilde{\boldsymbol{A}}_h^{(l_A)} \in \mathbb{R}^{n \times n}} \sum_{j=1}^{h} \left\| \sum_{i=1}^{l_A} \theta_A^{(i)}(\boldsymbol{\mu}_{m+j}) \left(\boldsymbol{V}_h^T \boldsymbol{A}_0^{(i)} \boldsymbol{V}_h + \delta \tilde{\boldsymbol{A}}_h^{(i)}\right) \boldsymbol{V}_h^T \hat{\boldsymbol{y}}_{\boldsymbol{\eta}'}(\boldsymbol{\mu}_{m+j}) - \tilde{\boldsymbol{f}}_h(\boldsymbol{\mu}_{m+j}) \right\|_2^2 . \tag{19}$$

Algorithm 2 Update auxiliary quantities

```
1: procedure AUXQU($\boldsymbol{B}_{h-1}^{(1)}, \boldsymbol{C}_{h-1}^{(1)}, \dots, \boldsymbol{C}_{h-1}^{(l_A)}, \boldsymbol{V}_h', \boldsymbol{p}, \boldsymbol{q}$)
2:     for $i = 1, \dots, l_A$ do
3:         Compute quantity $\boldsymbol{B}_h^{(i)} = \boldsymbol{B}_{h-1}^{(i)} \boldsymbol{V}_h' + \boldsymbol{A}_0^{(i)} \boldsymbol{p}\boldsymbol{q}^T$
4:         Compute quantity $\boldsymbol{C}_h^{(i)} = \boldsymbol{V}_h'^T \boldsymbol{C}_{h-1}^{(i)} + \boldsymbol{q}\boldsymbol{p}^T \boldsymbol{A}_0^{(i)}$
5:     end for
6: return $[\boldsymbol{B}_h^{(1)}, \boldsymbol{C}_h^{(1)}, \dots, \boldsymbol{C}_h^{(l_A)}]$
7: end procedure
```

We show in Theorem 1 that we recover the true reduced operators (13) if the solution of (19) is used to adapt the reduced operators from noise-free sensor samples. We therefore first show in Lemma 1 that (19) can be represented as n independent least-squares problems. This result is then used to prove Theorem 1.

Lemma 1. *Let $\hat{\boldsymbol{y}}_{\boldsymbol{\eta}'}(\boldsymbol{\mu}_{m+1}), \dots, \hat{\boldsymbol{y}}_{\boldsymbol{\eta}'}(\boldsymbol{\mu}_{m+h})$ be the columns of the sensor sample matrix $\boldsymbol{S}_h$, and let $\boldsymbol{V}_h$ be the adapted POD basis. The minimization problem* (19) *is a least-squares problem*

$$\min_{\delta\tilde{\boldsymbol{A}}_h \in \mathbb{R}^{l_A n \times n}} \|\tilde{\boldsymbol{U}}_h \delta\tilde{\boldsymbol{A}}_h - \tilde{\boldsymbol{R}}_h\|_F^2 \tag{20}$$

with system matrix

$$\tilde{\boldsymbol{U}}_h = \begin{bmatrix} \theta_A^{(1)}(\boldsymbol{\mu}_{m+1})\hat{\boldsymbol{y}}_{\boldsymbol{\eta}'}(\boldsymbol{\mu}_{m+1})^T\boldsymbol{V}_h & \dots & \theta_A^{(l_A)}(\boldsymbol{\mu}_{m+1})\hat{\boldsymbol{y}}_{\boldsymbol{\eta}'}(\boldsymbol{\mu}_{m+1})^T\boldsymbol{V}_h \\ \vdots & \ddots & \vdots \\ \theta_A^{(1)}(\boldsymbol{\mu}_{m+h})\hat{\boldsymbol{y}}_{\boldsymbol{\eta}'}(\boldsymbol{\mu}_{m+h})^T\boldsymbol{V}_h & \dots & \theta_A^{(l_A)}(\boldsymbol{\mu}_{m+h})\hat{\boldsymbol{y}}_{\boldsymbol{\eta}'}(\boldsymbol{\mu}_{m+h})^T\boldsymbol{V}_h \end{bmatrix} \in \mathbb{R}^{h \times l_A n}, \tag{21}$$

and right-hand side

$$\tilde{\boldsymbol{R}}_h = \begin{bmatrix} \boldsymbol{f}(\boldsymbol{\mu}_{m+1})^T\boldsymbol{V}_h - \hat{\boldsymbol{y}}_{\boldsymbol{\eta}'}(\boldsymbol{\mu}_{m+1})^T\boldsymbol{V}_h \sum_{i=1}^{l_A} \theta_A^{(i)}(\boldsymbol{\mu}_{m+1})\boldsymbol{V}_h^T\boldsymbol{A}_0^{(i)T}\boldsymbol{V}_h \\ \vdots \\ \boldsymbol{f}(\boldsymbol{\mu}_{m+h})^T\boldsymbol{V}_h - \hat{\boldsymbol{y}}_{\boldsymbol{\eta}'}(\boldsymbol{\mu}_{m+h})^T\boldsymbol{V}_h \sum_{i=1}^{l_A} \theta_A^{(i)}(\boldsymbol{\mu}_{m+1})\boldsymbol{V}_h^T\boldsymbol{A}_0^{(i)T}\boldsymbol{V}_h \end{bmatrix} \in \mathbb{R}^{h \times n}, \tag{22}$$

where the solution

$$\delta\tilde{\boldsymbol{A}}_h^T = \begin{bmatrix} \delta\tilde{\boldsymbol{A}}_h^{(1)} & \dots & \delta\tilde{\boldsymbol{A}}_h^{(l_A)} \end{bmatrix} \in \mathbb{R}^{n \times l_A n}$$

contains the updates $\delta\tilde{\boldsymbol{A}}_h^{(1)}, \dots, \delta\tilde{\boldsymbol{A}}_h^{(l_A)} \in \mathbb{R}^{n \times n}$ as blocks. Furthermore, the columns of $\delta\tilde{\boldsymbol{A}}_h$ are the solution of n independent least-squares problems

$$\min_{\boldsymbol{a}_i} \|\tilde{\boldsymbol{U}}_h\boldsymbol{a}_i - \boldsymbol{r}_i\|_2^2, \qquad i = 1, \dots, n, \tag{23}$$

where $\boldsymbol{a}_i \in \mathbb{R}^{l_A n}$ and $\boldsymbol{r}_i \in \mathbb{R}^h$ are the i-th column of $\delta\tilde{\boldsymbol{A}}_h$ and $\tilde{\boldsymbol{R}}_h$, respectively.

Proof. We transform the objective of (19) into $\|\tilde{\boldsymbol{U}}_h\delta\tilde{\boldsymbol{A}}_h - \tilde{\boldsymbol{R}}_h\|_F^2$ by exploiting that for a matrix $\boldsymbol{Z} \in \mathbb{R}^{n \times h}$, the following holds:

$$\sum_{j=1}^{h} \|\boldsymbol{z}_j\|_2^2 = \sum_{j=1}^{h}\sum_{i=1}^{n} Z_{ij}^2 = \|\boldsymbol{Z}\|_F^2,$$

where $\boldsymbol{z}_j \in \mathbb{R}^n$ is the j-th column of $\boldsymbol{Z}$ and Z_{ij} is the element of $\boldsymbol{Z}$ in row i and column j. Let $\boldsymbol{a}_i$ and $\boldsymbol{r}_i$ be the i-th column of $\delta\tilde{\boldsymbol{A}}_h$ and $\tilde{\boldsymbol{R}}_h$, respectively, then

$$\|\tilde{\boldsymbol{U}}_h\delta\tilde{\boldsymbol{A}}_h - \tilde{\boldsymbol{R}}_h\|_F^2 = \sum_{i=1}^{n} \|\tilde{\boldsymbol{U}}_h\boldsymbol{a}_i - \boldsymbol{r}_i\|_2^2$$

holds from which the splitting into n independent least-squares problems follows. □

Theorem 1. *If we have $h = l_A n$ noise-free sensor samples available, i.e., if the sensor sample $\hat{\boldsymbol{y}}_{\boldsymbol{\eta}'}(\boldsymbol{\mu}_{m+i})$ equals the FOM solution $\boldsymbol{y}_{\boldsymbol{\eta}'}(\boldsymbol{\mu}_{m+i})$ for all $i = 1, \ldots, h$, the adapted operators match the true reduced operators* (13) *that we would obtain by directly computing the (computationally expensive) matrix-matrix products with the POD basis $\boldsymbol{V}_h$ and full-order matrices $\boldsymbol{A}^{(1)}_{\boldsymbol{\eta}'}, \ldots, \boldsymbol{A}^{(l_A)}_{\boldsymbol{\eta}'} \in \mathbb{R}^{\mathcal{N}\times\mathcal{N}}$ corresponding to the latent parameter $\boldsymbol{\eta}' \in \mathcal{E}$.*

Proof. According to Lemma 1, the minimization problem (19) can be represented as n least-squares problems each with $l_A n$ unknowns and h equations. After we have collected $h = l_A n$ linearly independent sensor samples (cf. the problem description in Section 2.3), the least-squares problems become systems of linear equations of full rank and thus have a unique solution. If the dimension of the ROM n is large enough, then the noise-free projected sensor samples $\boldsymbol{V}_h^T \hat{\boldsymbol{y}}_{\boldsymbol{\eta}'}(\boldsymbol{\mu}_{m+j})$ are a solution of the ROM. This is indeed the case because the snapshot matrix contains the sensor samples and thus the projection error $\|\hat{\boldsymbol{y}}_{\boldsymbol{\eta}'}(\boldsymbol{\mu}_{m+j}) - \boldsymbol{V}_h \boldsymbol{V}_h^T \hat{\boldsymbol{y}}_{\boldsymbol{\eta}'}(\boldsymbol{\mu}_{m+j})\|_2^2$ can be made arbitrarily small (up to numerical tolerance) by retaining enough POD vectors. Therefore, if the dimension of the ROM is large enough, the true reduced operators (13) minimize

$$\sum_{j=1}^{h} \left\| \sum_{i=1}^{l_A} \theta_A^{(i)}(\boldsymbol{\mu}_{m+j}) \tilde{\boldsymbol{A}}^{(i)}_{\boldsymbol{\eta}'} \boldsymbol{V}_h^T \hat{\boldsymbol{y}}_{\boldsymbol{\eta}'}(\boldsymbol{\mu}_{m+j}) - \tilde{\boldsymbol{f}}_h(\boldsymbol{\mu}_{m+j}) \right\|_2^2 .$$

Then, the true updates

$$\delta\tilde{\boldsymbol{A}}^{(i)}_{\boldsymbol{\eta}'} = \tilde{\boldsymbol{A}}^{(i)}_{\boldsymbol{\eta}'} - \boldsymbol{V}_h^T \boldsymbol{A}_0^{(i)} \boldsymbol{V}_h , \qquad i = 1, \ldots, l_A , \tag{24}$$

are a solution of the minimization problem (19). Since the solution of (19) is unique, the updated and the true updates (24) must be equal. It follows that the updated and the true reduced operators are equal. □

3.2.3. Low-rank updates and computational procedure

If we have fewer than $l_A n$ linearly independent sensor samples available, Theorem 1 is not applicable. In this case, the system (20) becomes underdetermined. We therefore introduce low-rank updates of the block form

$$\delta\tilde{\boldsymbol{A}}^{(i)}_h = \begin{bmatrix} * & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{0} \end{bmatrix} \in \mathbb{R}^{n\times n} , \qquad i = 1, \ldots, l_A , \tag{25}$$

where only the block $* \in \mathbb{R}^{r\times r}$ can contain non-zero elements. We call $r \in \mathbb{N}$, with $r \leq n$, the rank of the update and call (25) low-rank if $r < n$. This reduces the number of unknowns in (20) from $l_A n^2$ to $l_A r^2$. The following Corollary 1 to Theorem 1 shows how to choose r to ensure a full-rank or an overdetermined least-squares problem.

Corollary 1. *Let $n \in \mathbb{N}$ be the dimension of the ROM, and let $l_A \in \mathbb{N}$ be the number of $\boldsymbol{\mu}$-independent operators of the FOM. If the rank $r \in \mathbb{N}$ of the update* (25) *is chosen as the floor of the ratio of the number of sensor samples h and the number of $\boldsymbol{\mu}$-independent operators l_A, i.e.,*

$$r = \left\lfloor \frac{h}{l_A} \right\rfloor , \tag{26}$$

then the least-squares problems (20) *cannot be underdetermined.*

Proof. With (26) we obtain $h \geq l_A r$. If $r < n$, the least-squares problem (20) corresponds to r independent least-squares problems of the form (23) because $n - r$ columns in all updates are set to zero. Each of these independent problems has $l_A r$ unknowns in h equations, and thus they are not underdetermined. □

Note that Corollary 1 also holds in the case $h < l_A$, where fewer sensor samples than operators are available. The rank of the update is then zero and the update (25) becomes the zero matrix. This means that no update is performed until at least l_A sensor samples are read. The dynamic ROM approach therefore cannot capture changes in the latent parameter if fewer than l_A sensor samples are read before another change occurs; thus, our approach is appropriate for situations in which changes in the latent parameter do not occur

Algorithm 3 Adapt reduced operators with sensor data

```
1: procedure ADAPTOPERATOR($\boldsymbol{V}_h, \boldsymbol{S}_h, \boldsymbol{H}_{h-1}$)
2:     Determine rank of additive update $r = \lfloor h/l_A \rfloor$
3:     If $h = 1$ then set auxiliary matrix $\boldsymbol{H}_1 = [\hat{\boldsymbol{y}}_{\boldsymbol{\eta}'}^T(\boldsymbol{\mu}_{m+1})\boldsymbol{V}_1] \in \mathbb{R}^{1\times n}$ else update matrix
           $$\boldsymbol{H}_h = \begin{bmatrix} \boldsymbol{H}_{h-1}\boldsymbol{V}_{h-1}^T\boldsymbol{V}_h \\ \hat{\boldsymbol{y}}_{\boldsymbol{\eta}'}^T(\boldsymbol{\mu}_{m+h})\boldsymbol{V}_h \end{bmatrix} \in \mathbb{R}^{h\times n}$$
4:     Assemble system matrix
           $$\tilde{\boldsymbol{U}}_h = \begin{bmatrix} \theta_A^{(1)}(\boldsymbol{\mu}_{m+1})\boldsymbol{H}_h(1,1:r) & \dots & \theta_A^{(l_A)}(\boldsymbol{\mu}_{m+1})\boldsymbol{H}_h(1,1:r) \\ \vdots & \ddots & \vdots \\ \theta_A^{(1)}(\boldsymbol{\mu}_{m+h})\boldsymbol{H}_h(h,1:r) & \dots & \theta_A^{(l_A)}(\boldsymbol{\mu}_{m+h})\boldsymbol{H}_h(h,1:r) \end{bmatrix} \in \mathbb{R}^{h\times l_A r}$$
5:     Assemble right-hand side
           $$\tilde{\boldsymbol{R}}_h = \begin{bmatrix} \boldsymbol{f}(\boldsymbol{\mu}_{m+1})^T\boldsymbol{V}_h(:,1:r) - \hat{\boldsymbol{y}}_{\boldsymbol{\eta}'}(\boldsymbol{\mu}_{m+1})^T\boldsymbol{V}_h\sum_{i=1}^{l_A}\theta_A^{(i)}(\boldsymbol{\mu}_{m+1})\boldsymbol{V}_h(:,1:r)^T\boldsymbol{A}_0^{(i)T}\boldsymbol{V}_h(:,1:r) \\ \vdots \\ \boldsymbol{f}(\boldsymbol{\mu}_{m+h})^T\boldsymbol{V}_h(:,1:r) - \hat{\boldsymbol{y}}_{\boldsymbol{\eta}'}(\boldsymbol{\mu}_{m+h})^T\boldsymbol{V}_h\sum_{i=1}^{l_A}\theta_A^{(i)}(\boldsymbol{\mu}_{m+h})\boldsymbol{V}_h(:,1:r)^T\boldsymbol{A}_0^{(i)T}\boldsymbol{V}_h(:,1:r) \end{bmatrix} \in \mathbb{R}^{h\times r}$$
6:     Solve minimization problem to derive additive updates
           $$\underset{\delta\tilde{\boldsymbol{A}}_h\in\mathbb{R}^{l_A r\times r}}{\arg\min}\ \|\tilde{\boldsymbol{U}}_h\delta\tilde{\boldsymbol{A}}_h - \tilde{\boldsymbol{R}}_h\|_F^2$$
7:     for $i = 1,\dots,l_A$ do
8:         Construct adapted operator $\tilde{\boldsymbol{A}}_h^{(i)}$ by using the additive updates and (16)
9:     end for
10:    for $i = 1,\dots,l_f$ do
11:        Adapt right-hand side $\tilde{\boldsymbol{f}}_h^{(i)} = \boldsymbol{V}_h'\tilde{\boldsymbol{f}}_{h-1}^{(i)} + \boldsymbol{p}\boldsymbol{q}^T\boldsymbol{f}^{(i)}$
12:    end for
13: return $[\boldsymbol{H}_h, \tilde{\boldsymbol{A}}_h^{(1)},\dots,\tilde{\boldsymbol{A}}_h^{(l_A)}, \tilde{\boldsymbol{f}}_h^{(1)},\dots,\tilde{\boldsymbol{f}}_h^{(l_f)}]$
14: end procedure
```

too rapidly and where there is both time and benefit to online data-informed decision making (e.g., mission replanning in the face of mild to moderate wing damage).

The computational procedure to construct the adapted reduced operators is summarized in Algorithm 3. It closely follows Lemma 1; however, it reuses the system matrix of the previous step, instead of assembling it from scratch at each step h. The auxiliary matrix $\boldsymbol{H}_0$ is initialized to an arbitrary scalar value (which is not used) and then is extended to $\boldsymbol{H}_h \in \mathbb{R}^{h\times n}$ with new sensor samples at each adaptivity step. Note that Algorithm 3 exploits that the right-hand sides are independent of the latent parameter and are therefore known to assemble matrix $\tilde{\boldsymbol{R}}_h$.

3.3. Dynamic reduced-order models and complexity analysis

We now combine the POD basis update of Section 3.1 and the reduced operator update of Section 3.2 into the dynamic ROM method summarized in Algorithm 4. The procedure is called at each adaptivity step $h = 1,\dots,m'$ during the online phase. It first updates the snapshot and sensor sample matrices and then adapts the POD basis with Algorithm 1. With the adapted basis $\boldsymbol{V}_h$, the auxiliary quantities are constructed with Algorithm 2. These are then used to construct the adapted reduced operators and right-hand sides with Algorithm 3.

Algorithm 4 Adaptivity procedure for dynamic ROMs

```
1: procedure ADAPTROM($\boldsymbol{V}_{h-1}$, $\tilde{\boldsymbol{A}}_{h-1}^{(1)}, \dots, \tilde{\boldsymbol{A}}_{h-1}^{(l_A)}$, $\tilde{\boldsymbol{f}}_{h-1}^{(1)}, \dots, \tilde{\boldsymbol{f}}_{h-1}^{(l_f)}$, $\boldsymbol{A}_0^{(1)}, \dots, \boldsymbol{A}_0^{(l_A)}$, $\boldsymbol{f}^{(1)}, \dots, \boldsymbol{f}^{(l_f)}$)
2:     Receive new sensor sample $\hat{\boldsymbol{y}}_{\boldsymbol{\eta}'}(\boldsymbol{\mu}_{m+h})$
3:     Update snapshot matrix $\boldsymbol{Y}_h = [\hat{\boldsymbol{y}}_{\boldsymbol{\eta}'}(\boldsymbol{\mu}_{m+1}), \dots, \hat{\boldsymbol{y}}_{\boldsymbol{\eta}'}(\boldsymbol{\mu}_{m+h}), \boldsymbol{y}_{\boldsymbol{\eta}_0}(\boldsymbol{\mu}_{h+1}), \dots, \boldsymbol{y}_{\boldsymbol{\eta}_0}(\boldsymbol{\mu}_m)]$
4:     Update sensor window $\boldsymbol{S}_h = [\hat{\boldsymbol{y}}_{\boldsymbol{\eta}'}(\boldsymbol{\mu}_{m+1}), \dots, \hat{\boldsymbol{y}}_{\boldsymbol{\eta}'}(\boldsymbol{\mu}_{m+h})]$
5:     Adapt POD basis to $\boldsymbol{V}_h$ with snapshot matrix $\boldsymbol{Y}_h$          ▷ Algorithm 1
6:     Update auxiliary quantities          ▷ Algorithm 2
7:     Compute low-rank updates $\delta\tilde{\boldsymbol{A}}_h^{(1)}, \dots, \delta\tilde{\boldsymbol{A}}_h^{(l_A)}$ from sensor samples in $\boldsymbol{S}_h$          ▷ Algorithm 3
8:     Adapt reduced operators to $\tilde{\boldsymbol{A}}_h^{(1)}, \dots, \tilde{\boldsymbol{A}}_h^{(l_A)}$ with updates $\delta\tilde{\boldsymbol{A}}_h^{(1)}, \dots, \delta\tilde{\boldsymbol{A}}_h^{(l_A)}$          ▷ Algorithm 3
9:     Adapt right-hand sides to $\tilde{\boldsymbol{f}}_h^{(1)}, \dots, \tilde{\boldsymbol{f}}_h^{(l_f)}$          ▷ Algorithm 3
10: end procedure
```

3.4. Complexity analysis

We analyze the runtime of adapting a dynamic ROM and show that it scales only linearly with the dimension $\mathcal{N}$ of the FOM if the full-order matrices with the initial latent parameter $\boldsymbol{\eta}_0$ are sparse.

Let us first consider the POD basis adaptivity procedure in Algorithm 1. Its runtime scales only linearly with the dimension $\mathcal{N}$ of the FOM. The computational costs of the POD adaptation are dominated by the (full) SVD of the $(n+1) \times (n+1)$ matrix $\boldsymbol{K}$ which is in $\mathcal{O}(n^3)$ [32].

Algorithm 2 updates the auxiliary quantities. This is the only computation in the dynamic ROM update that includes a matrix-vector product with the full-order matrices for the initial latent parameter. In general, this leads to runtime costs scaling quadratically with the dimension $\mathcal{N}$ of the FOM; however, most PDE discretization schemes lead to sparse matrices and thus in these situations the matrix-vector products in Algorithm 2 can be provided with linear runtime costs in $\mathcal{N}$. We emphasize that only the operators corresponding to the initial latent parameters have to be sparse.

Finally we consider adapting the reduced operators with Algorithm 3. Assembling the matrix $\boldsymbol{H}_h$ in line 3 requires the matrix-matrix product $\boldsymbol{V}_{h-1}^T\boldsymbol{V}_h$ which is in $\mathcal{O}(\mathcal{N}n^2)$ because $\boldsymbol{V}_{h-1}^T \in \mathbb{R}^{n\times\mathcal{N}}$ and $\boldsymbol{V}_h \in \mathbb{R}^{\mathcal{N}\times n}$. The product with $\boldsymbol{H}_{h-1} \in \mathbb{R}^{h-1\times n}$ is in $\mathcal{O}(hn^2)$. The reduction $\hat{\boldsymbol{y}}_{\boldsymbol{\eta}'}^T(\boldsymbol{\mu}_{m+h})\boldsymbol{V}_h$ scales linearly with $\mathcal{O}(\mathcal{N})$, and thus extending the matrix $\boldsymbol{H}_h$ has costs in $\mathcal{O}(\mathcal{N})$. The costs of assembling the system matrix $\tilde{\boldsymbol{U}}_h \in \mathbb{R}^{h\times l_A r}$ are independent of the dimension $\mathcal{N}$. Forming the right-hand sides in line 5 requires reducing $\boldsymbol{f}(\boldsymbol{\mu}_1), \dots, \boldsymbol{f}(\boldsymbol{\mu}_h) \in \mathbb{R}^{\mathcal{N}}$ with the adapted POD basis $\boldsymbol{V}_h$ and subtracting the sensor samples applied to the full-order operators. For that, we reuse the operators (15) computed with the auxiliary quantities to achieve a linear runtime with respect to $\mathcal{N}$. Finally, in line 6, r least squares problems are solved. Each of these problems has a system matrix of size $h \times rl_A$, and thus the costs of solving each problem is in $\mathcal{O}(hr^2l_A^2)$ because $h \geq rl_A$. Note that since the rank r is chosen depending on the number h of available sensor samples, we usually have $h \approx rl_A$, as well as $l_A \ll r$ and $r \leq n \ll \mathcal{N}$. Thus, the runtime of Algorithm 3 scales linearly with the dimension $\mathcal{N}$.

Because Algorithms 1–3 have linear runtime with respect to the dimension $\mathcal{N}$ of the FOM, the overall runtime of one adaptivity step is also linear in $\mathcal{N}$; however, this assumes the auxiliary quantities can be computed with costs linear in $\mathcal{N}$ due to the structure of the full-order matrices for latent parameter $\boldsymbol{\eta}_0$.

The linear dependence of the update runtime on the number of degrees of freedom $\mathcal{N}$ of the FOM might render the presented adaptivity scheme computationally infeasible in the online phase for certain applications; however, compared to rebuilding the ROM from scratch, significant runtime savings are obtained with the presented updating scheme, see the numerical results and runtime measurements in Section 4.3.

4. Numerical Results

We demonstrate the dynamic ROM approach on the deflection model of a plate where the latent parameter $\boldsymbol{\eta} \in \mathcal{E}$ controls local damage in the structure. Damage is modeled as a decrease in the thickness of the material. The plate model is based on the Mindlin plate theory [33] that takes into account transverse shear deformations and is therefore applicable to thick plates. The Mindlin plate theory is linear and neglects nonlinear effects such as postbuckling behavior [34]. We first build a ROM for the plate model with isotropic thickness in subregions defined by the initial latent parameter $\boldsymbol{\eta}_0$, i.e., no damage. We then consider a notional scenario in which the plate undergoes a local change of thickness, i.e., due to some damage event. We generate synthetic sensor data by changing the latent parameter of the FOM to $\boldsymbol{\eta}'$, generating the corresponding state solutions, and corrupting them with noise. We then use these synthetic sensor samples in our dynamic data-driven approach and show how the dynamic ROM adapts to the thickness change with no knowledge of the underlying latent parameter. The results confirm that the dynamic ROM quickly adapts to the changed situation and that the runtime of one adaptivity step scales only linearly with the number of degrees of freedom $\mathcal{N}$ of the FOM. In our example, one adaptivity step of the dynamic ROM is up to 3.8×10^4 times faster than rebuilding the ROM from scratch. The following subsections give more details on the problem setup and the results.

4.1. Problem setup

We consider the static analysis of a plate in bending. Our discretization and implementation is an extension of the implementation in [33]. Figure 3 shows the geometry of the plate. The plate is clamped into a frame and a pressure load is applied. The spatial domain $\Omega = [0,1]^2 \subset \mathbb{R}^2$ is split into four subregions $\Omega = \Omega_1 \uplus \Omega_2 \uplus \Omega_3 \uplus \Omega_4$. The model has eight observable $\boldsymbol{\mu} = [\mu_1, \dots, \mu_8]^T \in \mathcal{D} = [0.05, 0.1]^4 \times [1, 100]^4 \subset \mathbb{R}^8$ and two latent parameters $\boldsymbol{\eta} = [\eta_1, \eta_2]^T \in \mathcal{E} = [0, 0.2] \times (0, 0.05] \subset \mathbb{R}^2$. The thickness at position $\boldsymbol{x} \in \Omega$ is given by the function $t : \Omega \times \mathcal{D} \times \mathcal{E} \to \mathbb{R}$ with

$$t(\boldsymbol{x}; \boldsymbol{\mu}, \boldsymbol{\eta}) = t_0(\boldsymbol{x}; \boldsymbol{\mu}) - t_0(\boldsymbol{x}; \boldsymbol{\mu})\eta_1 \exp\left(-\frac{1}{2\eta_2^2}\|\boldsymbol{x} - \boldsymbol{z}\|_2^2\right) \tag{27}$$

and

$$t_0(\boldsymbol{x}; \boldsymbol{\mu}) = \begin{cases} \mu_1 & \text{if } x_1 < 0.5 \text{ and } x_2 < 0.5 \\ \mu_2 & \text{if } x_1 < 0.5 \text{ and } x_2 \geq 0.5 \\ \mu_3 & \text{if } x_1 \geq 0.5 \text{ and } x_2 < 0.5 \\ \mu_4 & \text{if } x_1 \geq 0.5 \text{ and } x_2 \geq 0.5 \end{cases} \tag{28}$$

with a pre-defined position $\boldsymbol{z} = [0.7, 0.4]$ in the spatial domain Ω. Thus, the parameters $\mu_1, \dots, \mu_4$ correspond to the nominal thickness in the subregions $\Omega_1, \dots, \Omega_4$ and the latent parameter $\boldsymbol{\eta}$ describes the decrease of the thickness due to damage at position $\boldsymbol{z}$ in the domain. The function (27) is nonlinear with respect to $\boldsymbol{x}$ and $\boldsymbol{\eta}$. The initial latent parameter is given as $\boldsymbol{\eta}_0 = [\eta_1, \eta_2]^T = [0, \epsilon]^T \in \mathcal{E}$ and leads to no decrease of the thickness at position $\boldsymbol{z}$. The constant ϵ can be set to any positive value, since it has no influence if $\eta_1 = 0$, see (27). The pressure load on each subregion can vary and is described by four observable parameters $\mu_5, \dots, \mu_8 \in [1, 100]^4$. We set the length of the plate to 1, Young's modulus to $E = 10920$, and the Poisson ratio to $\nu = 0.3$. This leads to a flexural rigidity of one and is convenient for non-dimensional results [33]. Figures 4a and 4b visualize the thickness of the plate for the observable parameters $[\mu_1, \mu_2, \mu_3, \mu_4]^T = [0.08, 0.060, 0.07, 0.065]^T$, and the initial parameter $\boldsymbol{\eta}_0$ and the latent parameter $\boldsymbol{\eta}' = [0.2, 0.05]^T \in \mathcal{E}$, respectively.

We follow [33] and discretize with the finite element method with four-noded Q4 elements with homogeneous Dirichlet boundary conditions. Each finite element node has three degrees of freedom. These are the deflection of the plate, the shear stress in x_1 direction, and the shear stress in x_2 direction. Taking the Dirichlet boundary conditions into account, the discretization leads to $\mathcal{N} \approx 3N^2$ degrees of freedom where $N \in \mathbb{N}$ is the number of equidistant grid points in each dimension x_1 and x_2. We obtain the full-order operator $\boldsymbol{A}_{\boldsymbol{\eta}}(\boldsymbol{\mu}) \in \mathbb{R}^{\mathcal{N} \times \mathcal{N}}$ depending on the observable $\boldsymbol{\mu} \in \mathcal{D}$ and the latent parameter $\boldsymbol{\eta} \in \mathcal{E}$. The pressure load is described by the right-hand side which depends on $\boldsymbol{\mu} \in \mathcal{D}$ only. We refer to [33] for details. The corresponding system of discrete equations is $\boldsymbol{A}_{\boldsymbol{\eta}}(\boldsymbol{\mu})\boldsymbol{y}_{\boldsymbol{\eta}}(\boldsymbol{\mu}) = \boldsymbol{f}(\boldsymbol{\mu})$ with the solution vector $\boldsymbol{y}_{\boldsymbol{\eta}}(\boldsymbol{\mu}) \in \mathbb{R}^{\mathcal{N}}$ that contains the deflection and the shear stress in x_1 and x_2 directions at each grid point.

The operator $\boldsymbol{A}_{\boldsymbol{\eta}}(\boldsymbol{\mu})$ has an affine parameter dependence (2) with respect to the observable parameter $\boldsymbol{\mu}$. It is given for $l_A = 8$ with the eight functions $\theta_A^{(1)}, \dots, \theta_A^{(8)} : \mathcal{D} \to \mathbb{R}$

$$\theta_A^{(1)}(\boldsymbol{\mu}) = \theta_A^{(2)}(\boldsymbol{\mu}) = \theta_A^{(3)}(\boldsymbol{\mu}) = \theta_A^{(4)}(\boldsymbol{\mu}) = (t_0(\boldsymbol{\mu}))^3$$

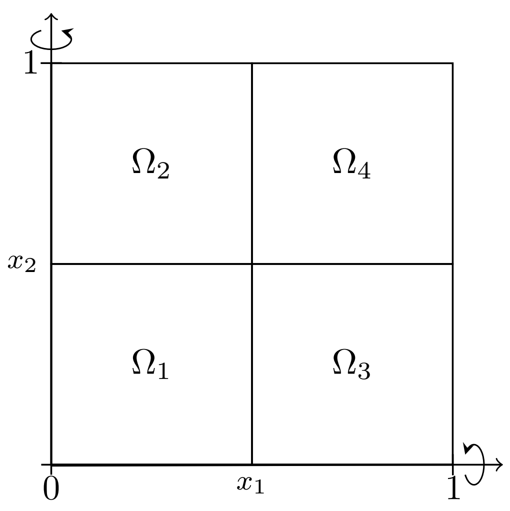


Figure 3: The plot shows the geometry of the plate. The plate has a different thickness in each of its four subregions $\Omega_1, \dots, \Omega_4$.

and

$$\theta_A^{(5)}(\boldsymbol{\mu}) = \theta_A^{(6)}(\boldsymbol{\mu}) = \theta_A^{(7)}(\boldsymbol{\mu}) = \theta_A^{(8)}(\boldsymbol{\mu}) = t_0(\boldsymbol{\mu}) \,.$$

The four operators $\boldsymbol{A}_{\boldsymbol{\eta}}^{(1)}, \dots, \boldsymbol{A}_{\boldsymbol{\eta}}^{(4)}$ describe the bending of the plate in the subregions $\Omega_1, \dots, \Omega_4$, and the operators $\boldsymbol{A}_{\boldsymbol{\eta}}^{(5)}, \dots, \boldsymbol{A}_{\boldsymbol{\eta}}^{(8)}$ the shear stress. The right-hand side can be decomposed into $l_f = 4$ $\boldsymbol{\mu}$-independent components $\boldsymbol{f}^{(1)}, \dots, \boldsymbol{f}^{(4)} \in \mathbb{R}^{\mathcal{N}}$ with the functions $\theta_f^{(1)}, \dots, \theta_f^{(4)} : \mathcal{D} \to \mathbb{R}$ as $\theta_f^{(i)}(\boldsymbol{\mu}) = \mu_{4+i}$ for $i = 1, \dots, 4$. In this problem setup, we emphasize that the operator $\boldsymbol{A}_{\boldsymbol{\eta}}(\boldsymbol{\mu})$ has no affine parameter dependence with respect to the latent parameter $\boldsymbol{\eta}$. Since the latent parameter is not represented explicitly in our dynamic ROM, this poses no problem for our approach. The solution of the plate model with observable parameter

$$\boldsymbol{\mu} = [0.08, 0.060, 0.07, 0.065, 50, 50, 100, 50]^T \in \mathcal{D}$$

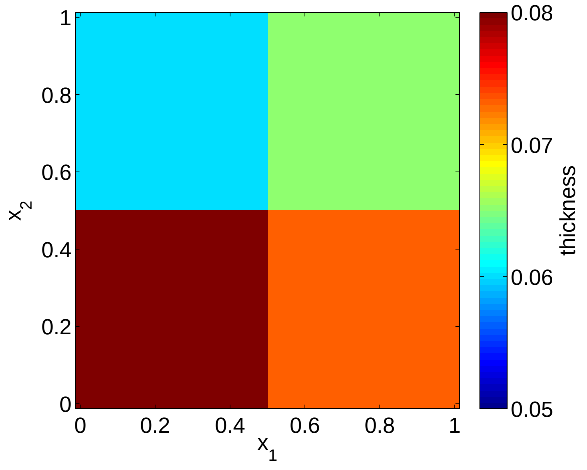
and $\boldsymbol{\eta} = \boldsymbol{\eta}_0$ is shown in Figure 4c. The solution in case of damage $\boldsymbol{\eta}' = [0.2, 0.05]^T$ is visualized in Figure 4d. Note that the difference between the solutions corresponding to the undamaged, Figure 4c, and the damaged, Figure 4d, plate may seem small visually, but the numerical results in Section 4.3 will show that the ROM without updates quickly fails to provide a valid approximation as the thickness of the plate decreases.

In the offline phase, we sample the FOM at $\boldsymbol{\mu}_1, \dots, \boldsymbol{\mu}_m \in \mathcal{D}$ randomly selected observable parameters and fixed latent parameter $\boldsymbol{\eta}_0$ and assemble the snapshot matrix $\boldsymbol{Y}_0$. We then derive the POD basis $\boldsymbol{V}_0 = [\boldsymbol{v}_1, \dots, \boldsymbol{v}_n] \in \mathbb{R}^{\mathcal{N} \times n}$ and construct the $\boldsymbol{\mu}$-independent reduced operators $\tilde{\boldsymbol{A}}_0^{(i)} = \boldsymbol{V}_0^T \boldsymbol{A}_0^{(i)} \boldsymbol{V}_0 \in \mathbb{R}^{n \times n}$ for $i = 1, \dots, 8$, and the reduced right-hand sides $\tilde{\boldsymbol{f}}_0^{(1)} = \boldsymbol{V}_0^T \boldsymbol{f}^{(1)}, \dots, \tilde{\boldsymbol{f}}_0^{(4)} = \boldsymbol{V}_0^T \boldsymbol{f}^{(4)} \in \mathbb{R}^n$. They lead to a ROM with a reduced system as in (8) with n degrees of freedom.

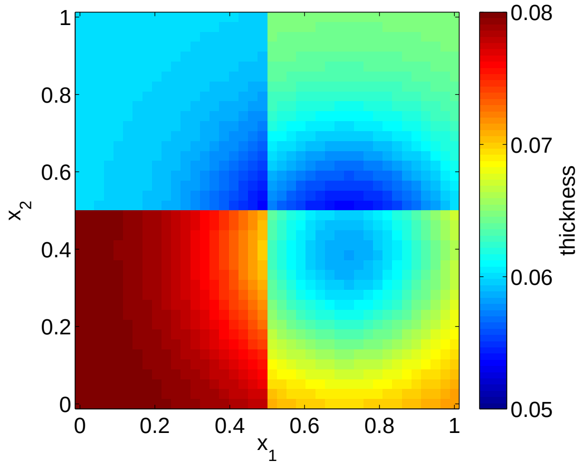
4.2. Singular values and latent parameters

Let us first consider the decay of the singular values of the snapshot matrix $\boldsymbol{Y}_{\boldsymbol{\eta}_0}$, of the corresponding $\boldsymbol{Y}_{\boldsymbol{\eta}'}$ with latent parameter $\boldsymbol{\eta}'$, and of a general snapshot matrix with varying $\boldsymbol{\eta} \in \mathcal{E}$. The number of snapshots in each case is $m = 1000$. The number of grid points in each direction of the finite element discretization is $N = 81$ and thus the number of degrees of freedom of the FOM is $\mathcal{N} = 19039$.

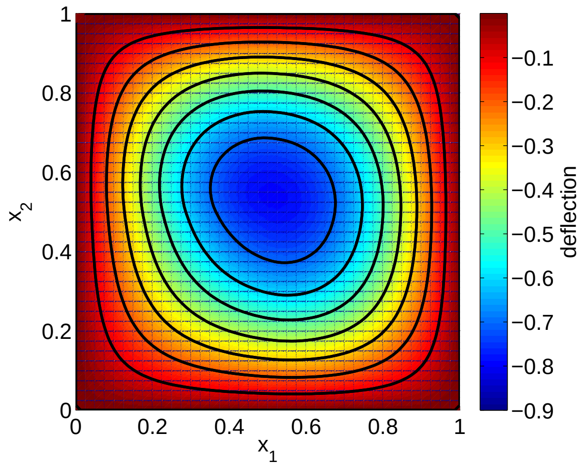
The plot in Figure 5a shows that the singular values decay with about the same rate if $\boldsymbol{\eta}$ is fixed, i.e., if either $\boldsymbol{\eta} = \boldsymbol{\eta}_0 = [0, \epsilon]^T$ or $\boldsymbol{\eta} = \boldsymbol{\eta}' = [0.2, 0.05]^T$. Figure 5b shows the decay of the singular values corresponding to snapshots with varying latent parameter $\boldsymbol{\eta}$. The decay is slower than in case of Figure 5a. Therefore, in this example, it is unnecessary to modify the number of POD basis vectors after changes in the latent parameter if the POD basis is constructed (or updated) with respect to a single latent parameter only, cf. the discussion of Algorithm 1 in Section 3.1.



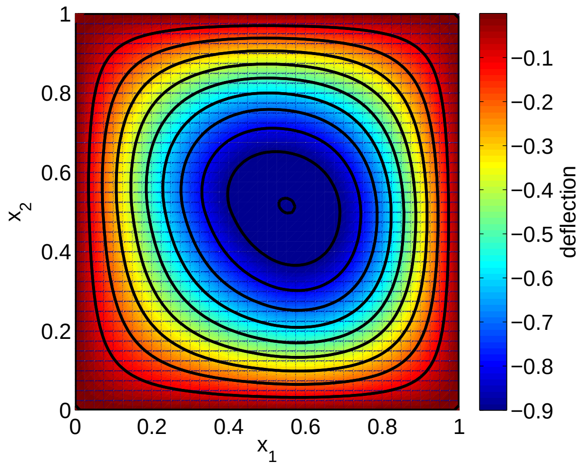
(a) thickness, no damage



(b) thickness, damage up to 20%

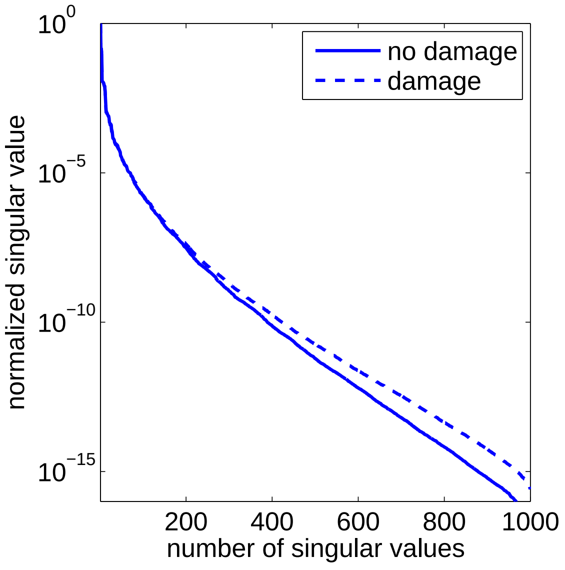


(c) deflection, no damage

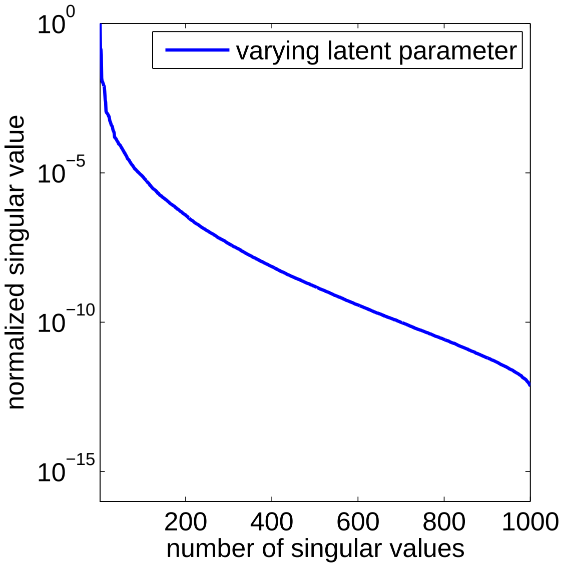


(d) deflection, damage up to 20%

Figure 4: The plot shows the thickness without damage in (a) and with damage at $\boldsymbol{z} = [0.7, 0.4] \in \Omega$ in (b). The corresponding deflection of the plate is shown in (c) and (d), respectively.



(a) singular values of $\boldsymbol{Y}_{\boldsymbol{\eta}_0}$ and $\boldsymbol{Y}_{\boldsymbol{\eta}'}$



(b) singular values of snapshots with varying latent parameter

Figure 5: The plots show the decay of the singular values corresponding to snapshot matrices with $\boldsymbol{\eta} = \boldsymbol{\eta}_0$ (no damage) and $\boldsymbol{\eta} = \boldsymbol{\eta}'$ (damage) in (a) and for varying $\boldsymbol{\eta}$ in (b).

4.3. Numerical experiments with dynamic reduced-order models

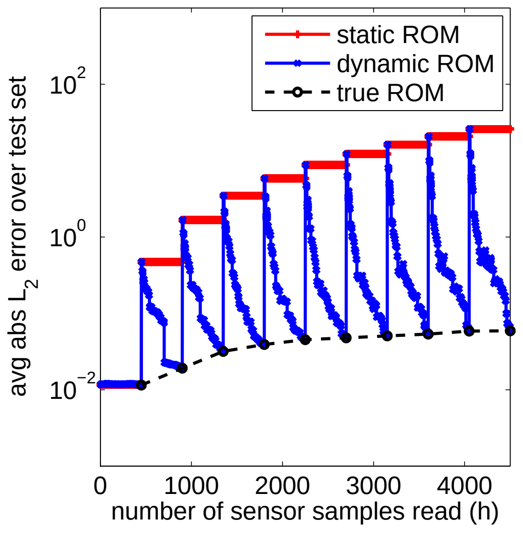
We now present numerical results to demonstrate the performance of our dynamic ROM approach for the deflection model of the clamped plate. The FOM has again $N = 81$ grid points in each direction which leads to $\mathcal{N} = 19039$ degrees of freedom. We create $m = 1000$ snapshots with the initial latent parameter $\boldsymbol{\eta}_0$ and build a ROM with $n = 50$ POD basis vectors. If not otherwise noted, we change the latent parameter $\boldsymbol{\eta}$ ten times with a linear decrease of the thickness

$$\boldsymbol{\eta} \in \{\boldsymbol{\eta}_0, [2/90, 2/360], [4/90, 4/360], \dots, [18/90, 18/360]\} \subset \mathcal{E} \tag{29}$$

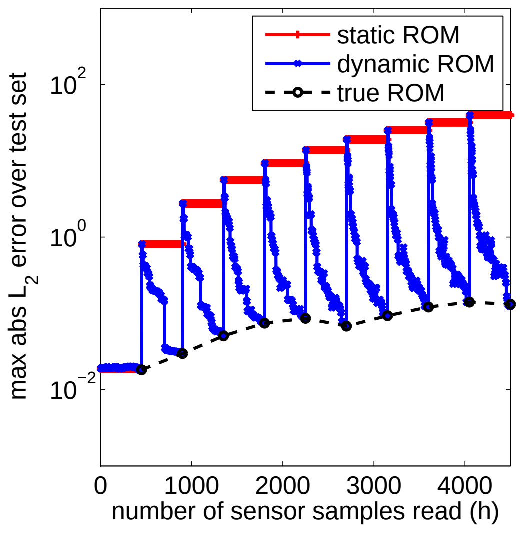
at position $\boldsymbol{z} = [0.7, 0.4] \in \Omega$. This corresponds to a maximum decrease of the thickness of the plate by 20%. After each change of the latent parameter $\boldsymbol{\eta}$, the sensor window (9) is flushed and reset to the empty matrix. For each change, we read $m' = 450$ sensor samples. Note that even though the window is flushed after a parameter change occurs, it is unnecessary to know the value of the latent parameter; it is sufficient to know just that it has changed. There are many methods available to detect such a change from sensor data. For example, novelty detection methods are widely studied in signal processing and machine learning, see the survey papers [35, 36, 37]. In the results, we will also investigate the effects of flushing the sensor window too early.

We first demonstrate dynamic ROMs with synthetic sensor samples that are not corrupted with noise. We therefore generate sensor samples that are the solutions of the FOM for randomly selected observable parameters $\boldsymbol{\mu}_{m+1}, \dots, \boldsymbol{\mu}_{m+m'} \in \mathcal{D}$ and for the respective latent parameters in (29). We then generate a test set with ten randomly chosen full-order solutions and compare the L_2 error of the states of the static ROM, the true ROM, and our dynamic ROM:

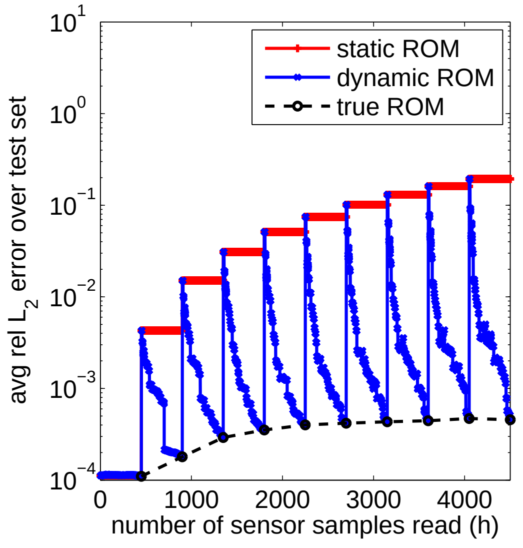
- The static ROM uses the POD basis $\boldsymbol{V}_0$ and the reduced system (8) computed in the offline phase. It is not adapted online.
- The true ROM is rebuilt from scratch with the adapted POD basis $\boldsymbol{V}_h$ and the true reduced operators (13) at each adaptivity step $h = 1, \dots, m'$.
- The dynamic ROM is adapted with Algorithm 4 at each step $h = 1, \dots, m'$.



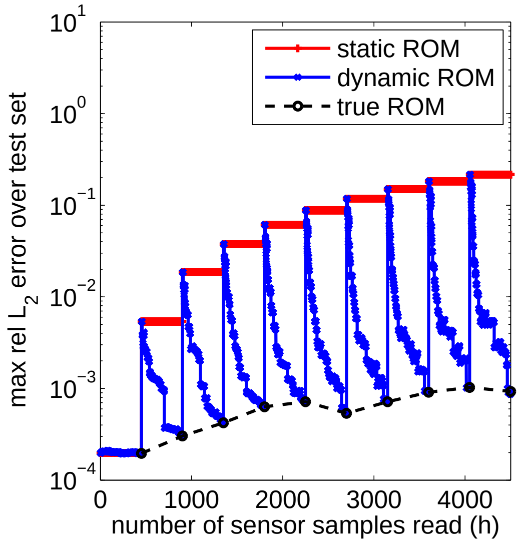
(a) averaged absolute L_2 error



(b) maximum absolute L_2 error



(c) averaged relative L_2 error



(d) maximum relative L_2 error

Figure 6: Whereas we obtain large errors with the static ROM, the dynamic ROM is able to adapt to the changed latent parameter $\boldsymbol{\eta}$ quickly. The plots also show that if enough sensor information is available, the dynamic ROM recovers the true ROM.

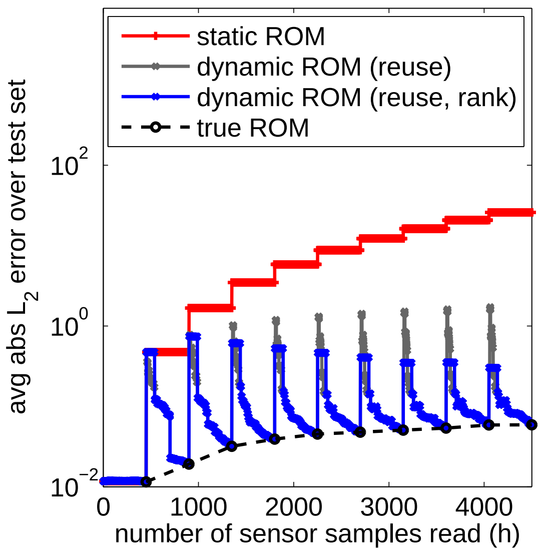
Figure 6a shows that the averaged absolute L_2 error of the static ROM increases as the latent parameter is changed. The dynamic ROM on the other hand quickly adapts to the new situation and finally recovers the true ROM. The rank r of the update (25) is increased with h. To obtain a well-conditioned system (20), we use the slightly more conservative ratio $r = \lfloor h/(l_A + 1) \rfloor$ rather than $\lfloor h/l_A \rfloor$. This means that an update with full rank $r = \lfloor h/(l_A + 1) \rfloor = \lfloor 450/(9 + 1) \rfloor = 50$ is performed after $h = 450$ sensor samples are read, and therefore that the true ROM is recovered after 450 sensor samples instead of after $l_A n = 400$ sensor samples. The maximum absolute L_2 error is reported in Figure 6b, which shows that the solutions for all parameters in the test set are approximated well. The relative L_2 errors in Figures 6c and 6d show a similar behavior as the corresponding absolute errors but are about two orders of magnitude lower.

The operator update as introduced in Section 3.2 and Algorithm 3 constructs the additive updates with respect to the reduced operators $\tilde{\boldsymbol{A}}_0^{(1)}, \dots, \tilde{\boldsymbol{A}}_0^{(l_A)} \in \mathbb{R}^{n \times n}$ which were computed in the offline phase. It is straightforward to extend Algorithm 3 such that the additive updates take previously adapted operators into account. For that, the right-hand sides in $\tilde{\boldsymbol{R}}_h$ are not only computed with respect to the reduced operators with $\boldsymbol{\eta}_0$ but with respect to the adapted operators. The accuracy results for the corresponding dynamic ROM are shown in Figure 7a. This reusing of the previously adapted operators prevents the error peaks after the sensor window was flushed. Additionally, we can impose a minimum rank of $r_{\min} \in \mathbb{N}$ such that the rank r has to be larger than $r_{\min}$ before an update is applied to the reduced operators. This avoids the error that is introduced if the rank of the update is low, cf. the blue and gray error curves in Figure 7a for $r_{\min} = 9$. For the same setting as in Figure 7a, we report in Figure 7b the L_2 error corresponding to a plate where the thickness is decreased in ten equidistant steps from $\boldsymbol{\eta}_0$ to $[0.5, 0.1] \in \mathbb{R}^2$. This corresponds to a maximum decrease of the thickness by 50%. The results confirm that the dynamic ROM approach still recovers the true ROM and also provides valid intermediate ROMs. Thus, the results show that the dynamic ROM recovers the true ROM independent of the difference between the solutions corresponding to the changed latent parameters, see Theorem 1.

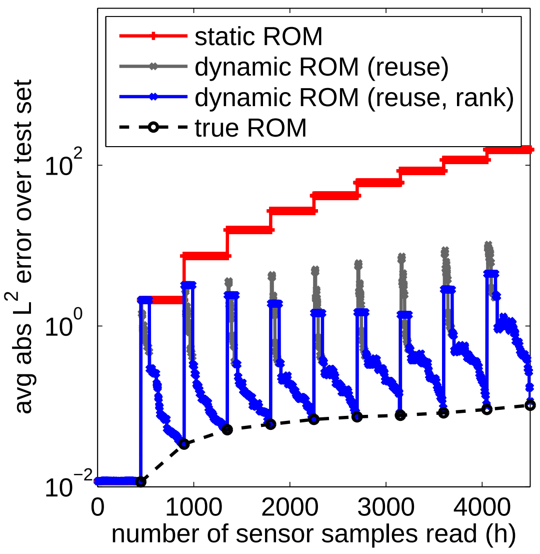
We discussed in Section 3.2.3 that it is necessary to increase the rank of the update depending on how many sensor samples are available to obtain a smooth transition to the true ROM. The results in Figure 8a demonstrate that if the rank is set too high the system (20) becomes underdetermined and thus the updates can lead to large errors. We now present an experiment where the dynamic ROM fails to recover the true ROM. Figure 8b reports the L_2 error when the dynamic ROM receives only 225 sensor samples per damage step, instead of $m' = 450$. Therefore, the dynamic ROM cannot completely recover the true ROM; however, it still provides more accurate results than the static ROM.

Besides the operators, we also adapt the POD basis at each step $h = 1, \dots, m'$. For this we use Algorithm 1. It reuses large parts of the adapted POD of the previous step $h - 1$ and only requires an SVD of a small and sparse matrix of size $(n + 1) \times (n + 1)$, see Section 3.1. However, since we only have a truncated SVD, the adapted POD basis provided by Algorithm 1 is not exact and thus differs from the basis computed from scratch. The error introduced by the approximate SVD is shown in Figure 9a. Even though the true ROM based on the approximate SVD achieves a slightly lower accuracy than the true ROM based on the rebuilt SVD, the difference is small compared to the error due to the change in the latent parameter. This also holds for the dynamic ROM because the dynamic ROM recovers the true ROM after sufficiently many sensor samples are read. Figure 9a also shows that the accuracy difference between the dynamic ROMs based on the rebuilt and on the approximate SVD increases only in the first few adaptivity steps but then stays constant during the rest of the adaptation. This indicates that the error incurred by the approximate SVD update does not accumulate when performing multiple updates.

Let us now consider sensor samples corrupted with noise. With low-rank updates, not only do we avoid an underdetermined system (20), but also we force the update to focus on the system characteristics corresponding to the first, and thus more important, POD basis vectors. Let us consider the system matrix and the right-hand side of the least-squares problem in Algorithm 3. Both are assembled by taking only the first r POD basis vectors into account. This means, in case of $h < m'$, they only capture the most important modes and thus tend to ignore noise. We demonstrate this with sensor samples where we add noise. We therefore generate noise vectors $\boldsymbol{y}_1^{\text{noise}}, \dots, \boldsymbol{y}_{m'}^{\text{noise}} \in \mathbb{R}^{\mathcal{N}}$ where each component of these vectors contains independent Gaussian noise with mean 0 and standard deviation 10^{-4}. To better reflect the situation in,

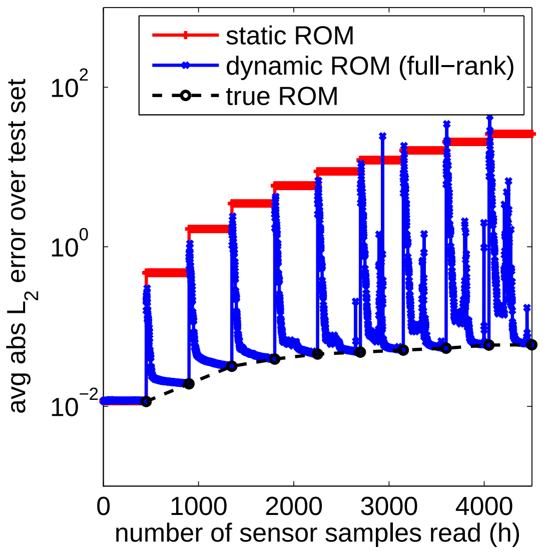


(a) reusing previously made updates

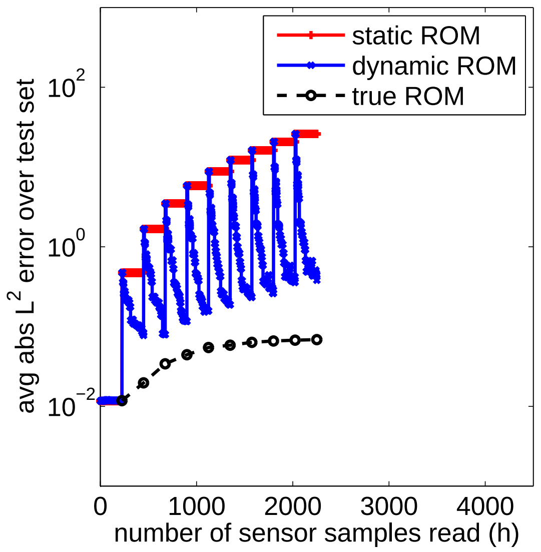


(b) recovering from damage with up to 50% decrease of thickness

Figure 7: In (a) the L_2 error of the states is shown for a dynamic ROM which reuses previously adapted operators (gray curve) and thus prevents the error peaks as observed in Figure 6. The error peaks can be further reduced by additionally imposing a minimum rank on the update (blue curve). The plot in (b) shows for the same setting as in (a) that the dynamic ROM also recovers from a damage with an up to 50% decrease of the thickness of the plate.

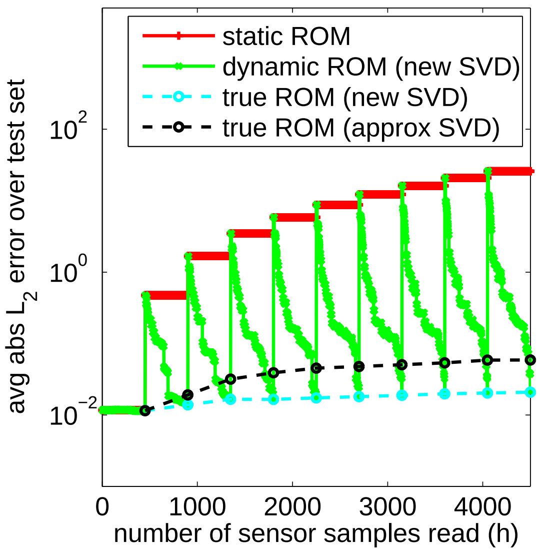


(a) full-rank updates

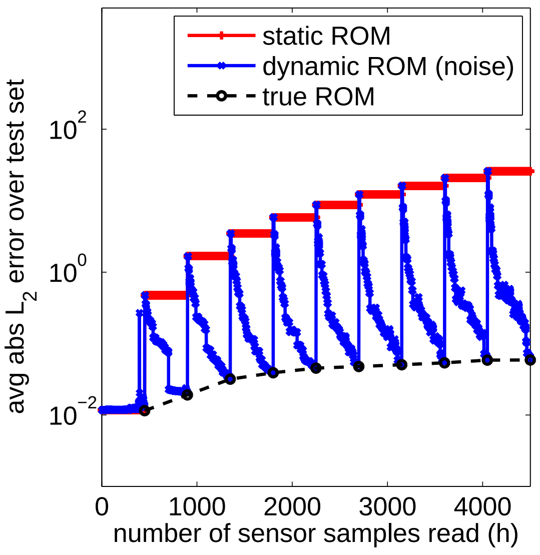


(b) not enough sensor samples to fully recover the true ROM

Figure 8: The plot in (a) shows that low-rank updates are necessary because full-rank updates can lead to large errors if not enough data is available. The results in (b) show that if insufficient sensor samples are available to obtain a full-rank update following Corollary 1, then the dynamic ROM cannot recover the true ROM but still provides more accurate results than the static ROM.



(a) rebuilding SVD from scratch



(b) dynamic ROM with sensor samples corrupted with noise

Figure 9: The left plot shows the error incurred by the approximate SVD update compared to rebuilding the SVD from scratch. The right plot demonstrates that the dynamic ROM also adapts to the new latent parameters if the sensor samples are polluted with noise.

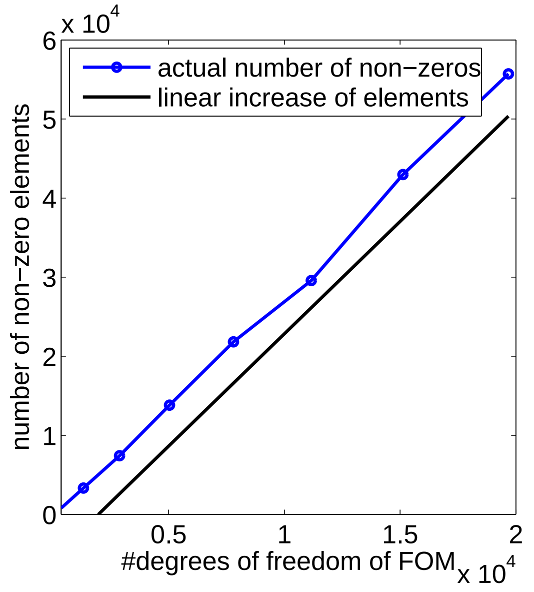
e.g., sensor networks, we introduce a spatial correlation by applying a moving window of size 5. This leads to noise in the range of 10^{-6}. This is the same range as reported for current fiber optic sensor systems[1], see, e.g., [38]. The sensor samples $\hat{\boldsymbol{y}}_{\boldsymbol{\eta}}(\boldsymbol{\mu}_{m+i}) = \boldsymbol{y}_{\boldsymbol{\eta}}(\boldsymbol{\mu}_{m+i}) + \boldsymbol{y}_i^{\text{noise}}$ for $i = 1, \dots, m'$ are obtained by adding the noise vectors to the FOM solutions $\boldsymbol{y}_{\boldsymbol{\eta}}(\boldsymbol{\mu}_{m+1}), \dots, \boldsymbol{y}_{\boldsymbol{\eta}}(\boldsymbol{\mu}_{m+m'}) \in \mathbb{R}^{\mathcal{N}}$. These sensor samples are then used to adapt the dynamic ROM. The results in Figure 9b show that our dynamic ROM approach is still able to adapted to the changed latent parameters from the sensor samples corrupted with noise. It even recovers the true ROM because the range of the absolute values of the components of the noise vectors is below the error of the ROM.

Finally, let us consider the runtime of the online phase of the dynamic ROMs. All of the following time measurements where performed on compute nodes with Intel Xeon E5-1620 CPUs and 32GB RAM using a MATLAB implementation.

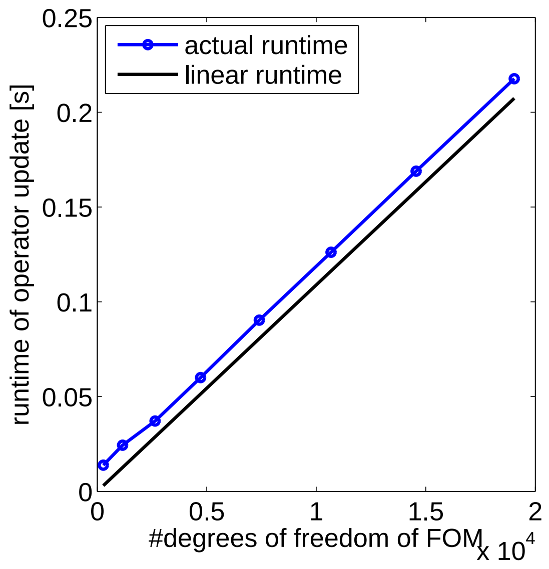
Figure 10 shows the runtime of adapting the reduced operator in the online phase. It scales linearly with the number $\mathcal{N}$ of degrees of freedom of the FOM because the full-order matrices corresponding to the initial latent parameter $\boldsymbol{\eta}_0$ are sparse.

Let us now compare the runtime of our dynamic ROM approach to classical model order reduction which rebuilds the ROM from scratch if the latent parameter of the underlying system changes. Rebuilding the ROM requires first inferring the latent parameter from the sensor samples and then rerunning the offline phase. To simplify the parameter inference, we consider here synthetic sensor samples without noise and therefore can infer the latent parameter exactly with a nonlinear least-squares problem. Note that this is indeed a nonlinear problem because the thickness function (27), and thus the operator, is nonlinear in the latent parameter $\boldsymbol{\eta}$. We use MATLAB's `lsqnonlin` method. Figure 11a reports the runtime of rebuilding the ROM and of adapting a dynamic ROM. For the dynamic ROM, we distinguish between the case where the basis is derived from a rebuilt SVD and where it is updated with an approximate SVD as discussed in Section 3.1.

[1]See the data sheet to the optical distributed sensor interrogator by Luna Inc available at http://lunainc.com/.



(a) number of non-zeros in full operators



(b) runtime of operator update

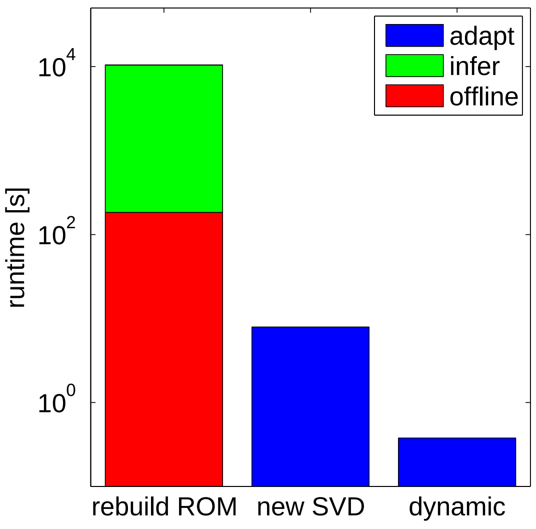
Figure 10: Because the full-order operators corresponding to the initial latent parameter $\boldsymbol{\eta}_0$ are sparse, the runtime of one adaptivity step of a dynamic ROM scales only linearly with the number of degrees of freedom of the FOM.

The results in Figure 11a show that an adaptivity step of the dynamic ROM with the approximate SVD (0.27 seconds) is about 28 times faster than recomputing the SVD (7.82 seconds), and about 3.8×10^4 times faster than rebuilding the ROM from scratch (10457 seconds). Even though the runtime of the dynamic ROM update scales linearly with the dimension of the FOM, the dynamic ROM achieves a large speedup here. As shown in Figure 11a, the speedup is in large part due to the avoidance of the inference of the latent parameter. The speedup of one dynamic ROM update versus rebuilding the ROM from scratch is shown for increasing $\mathcal{N}$ in Figure 11b. The speedup obtained with the dynamic ROM increases with the dimension $\mathcal{N}$ in this example. Therefore, these results indicate that the runtime for rebuilding the ROM grows faster with the dimension $\mathcal{N}$ of the FOM than the runtime of the dynamic ROM update. Overall, the results in Figure 11 show that significant speedups are obtained with the dynamic ROM compared to rebuilding from scratch, even though the adaptation of the dynamic ROM depends linearly on the dimension $\mathcal{N}$ of the FOM. We finally note that if the latent parameter had been known and thus the runtime of inferring the latent parameter had been excluded from the runtime of rebuilding the ROM, we would have obtained a speedup of 6.7×10^2 with our dynamic ROM.

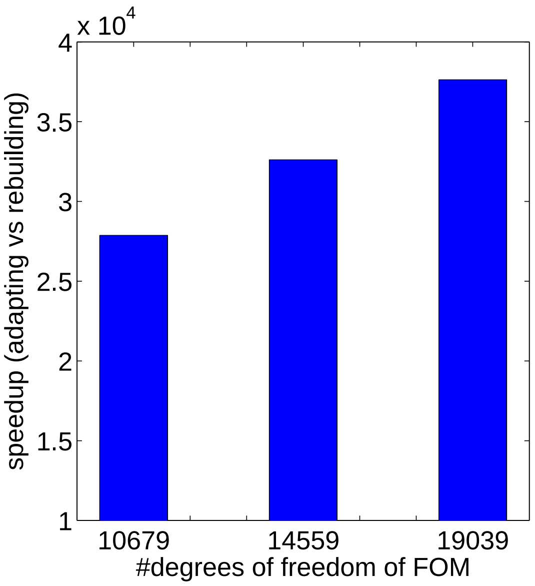
5. Conclusions

The key novel idea of dynamic ROMs is to adapt to changes in the underlying system by updating the reduced basis and the reduced operators directly from sensor data in the online phase. This avoids the computationally expensive FOM of the changed system.

The POD basis is adapted to the sensor data with an approximate SVD updating scheme and the reduced operators are adapted by inferring additive updates from the sensor data with a highly structured optimization problem. We ensure valid ROMs in case of limited sensor information by adapting with low-rank updates, where the rank is chosen such that the corresponding least-squares problem cannot become underdetermined. If sufficient and accurate data are available, our update scheme guarantees that we eventually recover the true ROM that we would obtain by rebuilding it from scratch. The runtime costs scale only linearly with the number of degrees of freedom of the FOM if the full-order matrices for the initial parameter configuration are sparse.



(a) runtime



(b) speedup of one adaptation compared to rebuilding

Figure 11: Adapting the dynamic ROM with the approximate SVD ("dynamic") is about 28 times faster than if the SVD is recomputed from scratch ("new SVD"), and about 3.8×10^4 times faster than rebuilding the ROM from scratch ("rebuild ROM"). The speedup of one adaptivity step compared to rebuilding increases as the number $\mathcal{N}$ of the degrees of freedom of the FOM is increased.

Because dynamic ROMs learn from sensor data instead of relying on the FOM, it is not necessary to evaluate the FOM at any other latent parameter than the one describing the initial state of the system. This is not only advantageous with respect to the runtime of updating the ROM but it also has at least two implications for the FOM. First, the discretization and solver routines of the FOM have to be available only for this initial parameter configuration. This often simplifies the FOM implementation or even allows reuse of available codes. Second, the latent parameters do not have to be selected when modeling the FOM. This means that the FOM does not have to anticipate all possible system changes. From a more general point of view, our dynamic ROM approach shows once more that models and data act in a symbiotic way and should not be considered as separate entities during modeling, implementation, and evaluation.

Dynamic ROMs are applicable if sensor data are available. We only considered real-time structural assessment and decision making but modern sensor technology is also advancing for control systems and, in general, for dynamic data-driven application systems (DDDAS). With the availability of accurate and massive amounts of sensor data, these are all possible further applications of dynamic ROMs. Besides the situation where sensor samples are available, dynamic ROMs are also applicable if sporadic FOM evaluations are feasible during the online phase. Dynamic ROMs then successively adapt to changing latent parameters, whereas rebuilding the ROM would require stopping the online phase until sufficient snapshot data are generated. Dynamic ROMs also avoid inferring the latent parameters and avoid assembling full-order matrices. In the setting where additional data might be provided by sporadic FOM evaluations, this would be particularly advantageous if the FOM is quick to evaluate, e.g., due to a fast (matrix-free) forward solver, compared to assembling the full-order operators to rebuild the ROM.

Future work includes an extension to derive updates from partial sensor samples. Such an extension could rely on gappy POD [24, 39] or the adaptivity scheme introduced in [15]. Another topic of future research is an approach to avoid the computationally expensive initialization phase that requires access to the FOM operators. For example, one could start with an initial operator that is, e.g., zero, and infer updates to this operator from data.

Acknowledgment

The authors would like to acknowledge the funding for this research, supported by AFOSR grant FA9550-11-1-0339 under the Dynamic Data-Driven Application Systems (DDDAS) Program (Program Manager Dr. Frederica Darema). Several examples were computed on the computer clusters of the Munich Centre of Advanced Computing.

References

[1] D. Kordonowy, O. Toupet, Composite airframe condition-aware maneuverability and survivability for unmanned aerial vehicles, in: Infotech@Aerospace 2011, AIAA Paper 2011-1496, 2011, pp. 1–10.

[2] D. Allaire, J. Chambers, R. Cowlagi, D. Kordonowy, M. Lecerf, L. Mainini, F. Ulker, K. Willcox, An Offline/Online DDDAS capability for self-aware aerospace vehicles, Procedia Computer Science 18 (0) (2013) 1959–1968.

[3] J. Degroote, J. Vierendeels, K. Willcox, Interpolation among reduced-order matrices to obtain parameterized models for design, optimization and probabilistic analysis, International Journal for Numerical Methods in Fluids 63 (2) (2010) 207–230.

[4] H. Panzer, J. Mohring, R. Eid, B. Lohmann, Parametric model order reduction by matrix interpolation, at – Automatisierungstechnik 58 (8) (2010) 475–484.

[5] D. Amsallem, C. Farhat, An online method for interpolating linear parametric reduced-order models, SIAM Journal on Scientific Computing 33 (5) (2011) 2169–2198.

[6] J. Burkardt, M. Gunzburger, H.-C. Lee, POD and CVT-based reduced-order modeling of Navier–Stokes flows, Computer Methods in Applied Mechanics and Engineering 196 (1–3) (2006) 337–355.

[7] J. Eftang, B. Stamm, Parameter multi-domain hp empirical interpolation, International Journal for Numerical Methods in Engineering 90 (4) (2012) 412–428.

[8] M. Dihlmann, M. Drohmann, B. Haasdonk, Model reduction of parametrized evolution problems using the reduced basis method with adaptive time-partitioning, in: D. Aubry, P. Díez, B. Tie, N. Parés (Eds.), Proceedings of the International Conference on Adaptive Modeling and Simulation, 2011, pp. 156–167.

[9] D. Amsallem, M. Zahr, C. Farhat, Nonlinear model order reduction based on local reduced-order bases, International Journal for Numerical Methods in Engineering 92 (10) (2012) 891–916.

[10] B. Peherstorfer, D. Butnaru, K. Willcox, H.-J. Bungartz, Localized discrete empirical interpolation method, SIAM Journal on Scientific Computing 36 (1) (2014) A168–A192.

[11] J. Eftang, A. Patera, Port reduction in parametrized component static condensation: approximation and a posteriori error estimation, International Journal for Numerical Methods in Engineering 96 (5) (2013) 269–302.

[12] S. Kaulmann, B. Haasdonk, Online greedy reduced basis construction using dictionaries, in: I. Troch, F. Breitenecker (Eds.), Proceedings of 7th Vienna International Conference on Mathematical Modelling, 2012, pp. 112–117.

[13] Y. Maday, B. Stamm, Locally adaptive greedy approximations for anisotropic parameter reduced basis spaces, SIAM Journal on Scientific Computing 35 (6) (2013) A2417–A2441.

[14] K. Washabaugh, D. Amsallem, M. Zahr, C. Farhat, Nonlinear model reduction for CFD problems using local reduced-order bases, in: 42nd AIAA Fluid Dynamics Conference and Exhibit, Fluid Dynamics and co-located Conferences, AIAA Paper 2012-2686, AIAA, 2012, pp. 1–16.

[15] D. Amsallem, M. Zahr, K. Washabaugh, Fast local reduced basis updates for the efficient reduction of nonlinear systems with hyper-reduction, Special issue on Model Reduction of Parameterized Systems (MoRePaS), Advances in Computational Mathematics (in review).

[16] K. Carlberg, Adaptive h-refinement for reduced-order models, International Journal for Numerical Methods in Engineering (accepted).

[17] C. Gogu, Improving the efficiency of large scale topology optimization through on-the-fly reduced order model construction, International Journal for Numerical Methods in Engineering 101 (4) (2015) 281–304.

[18] Y. Maday, O. Mula, A generalized empirical interpolation method: Application of reduced basis techniques to data assimilation, in: F. Brezzi, P. C. Franzone, U. Gianazza, G. Gilardi (Eds.), Analysis and Numerics of Partial Differential Equations, no. 4 in Springer INdAM Series, Springer, 2013, pp. 221–235.

[19] M. Yano, J. Penn, A. Patera, A model-data weak formulation for simultaneous estimation of state and model bias, Comptes Rendus Mathematique 351 (23-24) (2013) 937–941.

[20] R. Kalman, A new approach to linear filtering and prediction problems, Journal of Fluids Engineering 82 (1) (1960) 35–45.

[21] G. Evensen, Sequential data assimilation with a nonlinear quasi-geostrophic model using Monte Carlo methods to forecast error statistics, Journal of Geophysical Research: Oceans 99 (C5) (1994) 10143–10162.

[22] E. Constantinescu, A. Sandu, T. Chai, G. Carmichael, Assessment of ensemble-based chemical data assimilation in an idealized setting, Atmospheric Environment 41 (1) (2007) 18 – 36.

[23] C. Johns, J. Mandel, A two-stage ensemble Kalman filter for smooth data assimilation, Environmental and Ecological Statistics 15 (1) (2008) 101–110.

[24] R. Everson, L. Sirovich, Karhunen-Loève procedure for gappy data, Journal of the Optical Society of America A: Optics, Image Science & Vision 12 (8) (1995) 1657–1664.

[25] P. Astrid, S. Weiland, K. Willcox, T. Backx, Missing point estimation in models described by proper orthogonal decomposition, Automatic Control, IEEE Transactions on 53 (10) (2008) 2237–2251.

[26] M. Barrault, Y. Maday, N. Nguyen, A. Patera, An 'empirical interpolation' method: application to efficient reduced-basis discretization of partial differential equations, Comptes Rendus Mathematique 339 (9) (2004) 667–672.

[27] S. Chaturantabut, D. Sorensen, Nonlinear model reduction via discrete empirical interpolation, SIAM Journal on Scientific Computing 32 (5) (2010) 2737–2764.

[28] K. Veroy, A. Patera, Certified real-time solution of the parametrized steady incompressible Navier-Stokes equations: rigorous reduced-basis a posteriori error bounds, International Journal for Numerical Methods in Fluids 47 (8-9) (2005) 773–788.

[29] C. Lieberman, K. Willcox, O. Ghattas, Parameter and state model reduction for large-scale statistical inverse problems, SIAM Journal on Scientific Computing 32 (5) (2010) 2523–2542.

[30] T. Bui-Thanh, K. Willcox, O. Ghattas, Model reduction for large-scale systems with high-dimensional parametric input space, SIAM Journal on Scientific Computing 30 (6) (2008) 3270–3288.

[31] B. Peherstorfer, S. Zimmer, H.-J. Bungartz, Model reduction with the reduced basis method and sparse grids, in: J. Garcke, M. Griebel (Eds.), Sparse Grids and Applications, Vol. 88 of Lecture Notes in Computational Science and Engineering, Springer, 2013, pp. 223–242.

[32] M. Brand, Fast low-rank modifications of the thin singular value decomposition, Linear Algebra and its Applications 415 (1) (2006) 20–30.

[33] A. Ferreira, MATLAB Codes for Finite Element Analysis, Springer, 2008.

[34] E. Ventsel, T. Krauthammer, Thin Plates and Shells, CRC Press, 2001.

[35] M. Markou, S. Singh, Novelty detection: a review-part 1: statistical approaches, Signal Processing 83 (12) (2003) 2481–2497.

[36] M. Markou, S. Singh, Novelty detection: a review-part 2: neural network based approaches, Signal Processing 83 (12) (2003) 2499–2521.

[37] M. Pimentel, D. Clifton, L. Clifton, L. Tarassenko, A review of novelty detection, Signal Processing 99 (2014) 215–249.

[38] M. Lecerf, A data-driven approach to online flight capability estimation, S.M. Thesis, Massachusetts Institute of Technology, Cambridge, MA (2014).

[39] L. Mainini, K. Willcox, Sensitivity analysis of surrogate-based methodology for real time structural assessment, in: AIAA Modeling and Simulation Technologies Conference, AIAA SciTech 2015, AIAA Paper 2015-1362, AIAA, 2015.