SAMPLING LOW-DIMENSIONAL MARKOVIAN DYNAMICS FOR PREASYMPTOTICALLY RECOVERING REDUCED MODELS FROM DATA WITH OPERATOR INFERENCE[*]

BENJAMIN PEHERSTORFER[†]

Abstract. This work introduces a method for learning low-dimensional models from data of high-dimensional black-box dynamical systems. The novelty is that the learned models are exactly the reduced models that are traditionally constructed with classical projection-based model reduction techniques. Thus, the proposed approach learns models that are guaranteed to have the well-studied properties of reduced models known from model reduction, without requiring full knowledge of the governing equations and without requiring the operators of the high-dimensional systems. The key ingredient is a new data sampling scheme to obtain re-projected trajectories of high-dimensional systems that correspond to Markovian dynamics in low-dimensional subspaces. The exact recovery of reduced models from these re-projected trajectories is guaranteed preasymptotically under certain conditions for finite amounts of data and for a large class of systems with polynomial nonlinear terms. Numerical results demonstrate that the low-dimensional models learned with the proposed approach match reduced models from traditional model reduction up to numerical errors in practice. The numerical results further indicate that low-dimensional models fitted to re-projected trajectories are predictive even in situations where models fitted to trajectories without re-projection are inaccurate and unstable.

Key words. operator inference, data-driven modeling, nonintrusive model reduction, proper orthogonal decomposition, reduced basis method

AMS subject classifications. 65M60, 68T10, 65P99, 65Y99, 65F99

DOI. 10.1137/19M1292448

1. Introduction.

Reduced models have become a ubiquitous tool to make tractable computations that require large numbers of model evaluations in, e.g., uncertainty quantification, optimization, and inverse problems. Traditional model reduction derives reduced models from high-dimensional (full) models of systems that typically are given in the form of partial differential equations (PDEs) and their corresponding discretized operators, which have to be available either in assembled form or implicitly through methods that return the action of the operators on given vectors. The properties of reduced models have been extensively studied by the model reduction community [2, 49, 4, 9], and even rigorous error estimation has been established for certain classes of problems [63, 62, 23, 27, 55, 49]. The aim of nonintrusive model reduction methods is to learn reduced models from data of the high-dimensional systems alone and so to extend the scope of model reduction to settings where the operators of the high-dimensional systems are unavailable; however, the models learned from data alone typically are only approximations of the reduced models obtained with traditional model reduction, and thus establishing the same rigor for the learned models as for reduced models is challenging. In contrast, this work presents an approach to learn low-dimensional models from data that exactly match the reduced

[*] Submitted to the journal's Methods and Algorithms for Scientific Computing section October 10, 2019; accepted for publication (in revised form) August 4, 2020; published electronically October 27, 2020.

https://doi.org/10.1137/19M1292448

Funding: This work was partially supported by the U.S. Department of Energy, Office of Advanced Scientific Computing Research, Applied Mathematics Program (Program Manager Dr. Steven Lee), DOE award DESC0019334. This work was also partially supported by NSF grant IIS-1901091.

[†] Courant Institute of Mathematical Sciences, New York University, New York, NY 10012 USA (pehersto@cims.nyu.edu).

models that are obtained with traditional model reduction as if the operators of the high-dimensional systems were available. This guarantee of exactly recovering reduced models from data holds preasymptotically in the number of data points and for a wide class of high-dimensional systems with polynomial nonlinear terms under certain conditions. Thus, models learned with the proposed approach *are* the reduced models of traditional model reduction and therefore directly inherit their well-studied properties.

There is a large body of literature on learning dynamical-system models from data. First, there are data-fit surrogate-modeling approaches that use interpolation and regression to learn maps from parameters to coefficients of approximations of high-dimensional solutions in low-dimensional subspaces [36, 6, 66, 26, 29, 67]. There is a wide range of applications where such data-fit surrogate models have been employed successfully [59]; however, it is challenging to formulate systems and control theory concepts such as stability, attractors, and eigenmodes for such surrogate models. We do not consider data-fit surrogate models in the following. Second, there is system identification that originated in the systems and control community [35]. The Loewner approach was introduced by Antoulas and collaborators [3, 34, 37, 7] and has been extended from linear time-invariant systems to parametrized [30], bilinear [5], and quadratic-bilinear systems [20]. Under certain conditions, the models learned with the Loewner approach are the reduced models that are obtained with interpolatory model reduction; however, Loewner models are learned from frequency-response data rather than from time-domain data. The work [41] builds on Loewner to learn reduced models of linear time-invariant systems from time-domain data; however, learning from time-domain data can introduce errors, and so the learned models can differ from the corresponding Loewner models derived from frequency-response data. Third, there is dynamic mode decomposition [54, 53, 48, 61, 32] that best-fits linear operators to state trajectories with respect to the L_2 norm. Methods based on the Koopman operator have been developed as one path to extending dynamic mode decomposition to nonlinear dynamical systems [38, 65, 10]. Fourth, there are methods that learn parsimonious models by exploiting sparsity in the high-dimensional systems; see, e.g., [51, 52, 11, 50]. The learned models typically are either continuous in the sense that terms of PDEs are learned from a dictionary or high-dimensional models are learned that inherit sparsity from, e.g., finite-element discretizations of the governing equations of the systems of interest. In contrast, we aim to learn low-dimensional models that help to reduce computational costs in applications that require many model evaluations [9, 45].

We aim to exactly recover reduced models from data so that the learned models have the same well-studied properties as reduced models constructed with traditional model reduction methods. Our approach is based on operator inference [44], which has been derived from [43] and is a data-driven model reduction approach that learns approximations of reduced models from state trajectories. In [44], operator inference has been introduced for systems with polynomial nonlinear terms, and in [47] operator inference is combined with the "transform & learn" approach to obtain models of systems with more general nonlinear terms. Operator inference projects trajectories of systems of interest onto low-dimensional subspaces of the high-dimensional state spaces and then fits operators to the projected trajectories via least-squares regression. However, as is known from, e.g., the Mori–Zwanzig formalism from statistical physics [19, 16], the projected trajectories correspond to non-Markovian dynamics in the low-dimensional subspaces even though the high-dimensional trajectories and the corresponding high-dimensional systems are Markovian. Dynamical systems are

Markovian if future states depend only on the current state and not on previous ones. In contrast, non-Markovian systems can be thought of having a memory so that future states depend on the current and previous states. A formal definition follows in section 3.1. The non-Markovian dynamics are related to the closure error in model reduction [64, 18, 39, 21, 57, 68]. To account for the non-Markovian dynamics, methods have been proposed that learn non-Markovian terms [15, 39, 21, 68] and that use time-delay and other embeddings [60, 33, 14]; however, since we aim to exactly recover the Markovian reduced models that are obtained with traditional model reduction, neither of these remedies is applicable in our situation. Instead, we propose a data sampling scheme that iterates between time stepping the high-dimensional systems and projecting onto low-dimensional subspaces to generate trajectories that correspond to low-dimensional Markovian dynamics. To generate these re-projected trajectories, it is necessary to time step the high-dimensional systems at selected initial conditions, which is in contrast to other data-driven modeling techniques that are applicable even if only data are available. We show that, under certain conditions, applying operator inference to re-projected trajectories gives the same operators that are obtained with traditional model reduction methods. The result is a preasymptotic guarantee to exactly recover reduced models from finite amounts of data for a wide class of systems with polynomial nonlinear terms. Our numerical results demonstrate these theoretical results in practice by learning low-dimensional models that match the reduced models from traditional model reduction up to numerical errors.

Section 2 discusses preliminaries on dynamical systems, traditional model reduction, and operator inference and formulates the problem. Section 3 introduces data sampling with re-projection to obtain trajectories that correspond to low-dimensional Markovian dynamics and provides an analysis that shows that operators fitted to these re-projected trajectories are the reduced operators obtained with traditional model reduction. The overall computational approach is presented in Algorithm 2 in section 4, and numerical results are given in section 5. Conclusions are drawn in section 6.

2. Preliminaries.

The focus of this work is on dynamical systems with polynomial nonlinear terms, which we introduce in section 2.1 together with traditional model reduction for these systems in section 2.2. A building block of our approach is operator inference [44] for learning reduced models from data, which we discuss in section 2.3. The problem we aim to address is formulated in section 2.4.

2.1. Dynamical systems with polynomial nonlinear terms.

Let $K \in \mathbb{N}$, and consider a dynamical system of the form

$$\boldsymbol{x}_{k+1}(\boldsymbol{\mu}) = \boldsymbol{f}(\boldsymbol{x}_k(\boldsymbol{\mu}), \boldsymbol{u}_k(\boldsymbol{\mu}); \boldsymbol{\mu}) \,, \qquad k = 0, \dots, K-1 \,, \tag{2.1}$$

with state $\boldsymbol{x}_k(\boldsymbol{\mu}) \in \mathbb{R}^N$ of dimension $N \in \mathbb{N}$ and input $\boldsymbol{u}_k(\boldsymbol{\mu}) \in \mathbb{R}^p$ of dimension $p \in \mathbb{N}$ at time steps $k = 0, \dots, K-1$. The parameter $\boldsymbol{\mu} \in \mathcal{D} \subset \mathbb{R}^d$ is independent of the time step. The initial condition is $\boldsymbol{x}_0(\boldsymbol{\mu}) \in \mathbb{R}^N$. The potentially nonlinear function $\boldsymbol{f} : \mathbb{R}^N \times \mathbb{R}^p \times \mathcal{D} \to \mathbb{R}^N$ describes the dynamics of system (2.1). Set $N_i = \binom{N+i-1}{i}$ for $i \in \mathbb{N}$. In the following, we consider systems (2.1) that are polynomial of order $\ell \in \mathbb{N}$, which means that there exists $\boldsymbol{A}_i(\boldsymbol{\mu}) \in \mathbb{R}^{N \times N_i}$ for $i = 1, \dots, \ell$ and $\boldsymbol{B}(\boldsymbol{\mu}) \in \mathbb{R}^{N \times p}$ for $\boldsymbol{\mu} \in \mathcal{D}$ such that

$$\boldsymbol{f}(\boldsymbol{x}_k(\boldsymbol{\mu}), \boldsymbol{u}_k(\boldsymbol{\mu}); \boldsymbol{\mu}) = \sum_{i=1}^{\ell} \boldsymbol{A}_i(\boldsymbol{\mu}) \boldsymbol{x}_k^i(\boldsymbol{\mu}) + \boldsymbol{B}(\boldsymbol{\mu}) \boldsymbol{u}_k(\boldsymbol{\mu}) \,, \qquad k = 0, \dots, K-1 \,. \tag{2.2}$$

The vector $\boldsymbol{x}_k^i(\boldsymbol{\mu}) \in \mathbb{R}^{N_i}$ is the ith power of $\boldsymbol{x}_k(\boldsymbol{\mu})$, which is constructed from the Kronecker product $\boldsymbol{x}_k(\boldsymbol{\mu}) \otimes \cdots \otimes \boldsymbol{x}_k(\boldsymbol{\mu})$ by removing all duplicate entries due to commutativity of the multiplication [44]. Note that $\boldsymbol{x}_k^i(\boldsymbol{\mu})$ preserves the ordering of $\boldsymbol{x}_k(\boldsymbol{\mu}) \otimes \cdots \otimes \boldsymbol{x}_k(\boldsymbol{\mu})$ in the sense that for two components of $\boldsymbol{x}_k^i(\boldsymbol{\mu})$ with indices j_1 and l_1 with $j_1 < l_1$, the indices j_2 and l_2 of the corresponding two components in $\boldsymbol{x}_k(\boldsymbol{\mu}) \otimes \cdots \otimes \boldsymbol{x}_k(\boldsymbol{\mu})$ satisfy $j_2 < l_2$. Note further that $\boldsymbol{x}_k^1(\boldsymbol{\mu}) = \boldsymbol{x}_k(\boldsymbol{\mu})$. Define the trajectories $\boldsymbol{X}(\boldsymbol{\mu}) = [\boldsymbol{x}_0(\boldsymbol{\mu}), \dots, \boldsymbol{x}_{K-1}(\boldsymbol{\mu})] \in \mathbb{R}^{N \times K}$ and $\boldsymbol{Y}(\boldsymbol{\mu}) = [\boldsymbol{x}_1(\boldsymbol{\mu}), \dots, \boldsymbol{x}_K(\boldsymbol{\mu})] \in \mathbb{R}^{N \times K}$. Let further $\boldsymbol{X}^i(\boldsymbol{\mu}) = [\boldsymbol{x}_0^i(\boldsymbol{\mu}), \dots, \boldsymbol{x}_{K-1}^i(\boldsymbol{\mu})] \in \mathbb{R}^{N_i \times K}$, for $i = 1, \dots, \ell$, be the trajectories corresponding to the ith powers of the states at times $k = 0, \dots, K-1$. More details on systems with polynomial nonlinear terms and their relevance in computational science and engineering can be found in, e.g., [24, 8, 31, 22, 44].

Systems of the form (2.1) are typically obtained from single-step time discretizations of systems of ordinary differential equations [28]. Discretizations based on the explicit Euler method lead to systems (2.1). Discretizing polynomial systems with higher-order explicit Runge–Kutta schemes leads to polynomial time-discrete systems of the form (2.2) as well if the inputs are interpolated between the time steps. Note, however, that the degree ℓ of the polynomial (2.2) grows exponentially with the order of the scheme, which might make computations challenging in practice. We will consider a linear system ($\ell = 1$) that is derived from a fourth-order explicit Runge–Kutta scheme in section 2.4. We will further demonstrate with a numerical example in section 5.3.3 that the proposed approach is applicable to multistep schemes such as the Adams–Bashforth schemes as well if the first time step is computed with the explicit Euler method and thus the system has form (2.2) in the first time step. If the system is linear ($\ell = 1$), then implicit time discretizations with the implicit Euler method can be represented in the form (2.2) and the proposed approach is applicable. Extensions to implicit discretizations for $\ell > 1$ are future work, even though we expect that the principles described in the following remain the same.

2.2. Model reduction of systems with polynomial nonlinear terms. If operators $\boldsymbol{A}_1(\boldsymbol{\mu}), \dots, \boldsymbol{A}_\ell(\boldsymbol{\mu}), \boldsymbol{B}(\boldsymbol{\mu})$ of (2.1) for $\boldsymbol{\mu} \in \mathcal{D}$ are available either explicitly in assembled form or implicitly through a method that provides the action of the operators on given vectors, then traditional projection-based model reduction can be applied to find a reduced model; see, e.g., [49, 9]. Traditional projection-based model reduction typically first constructs a reduced space and then projects the operators of the high-dimensional system to obtain the reduced operators and to assemble the reduced model. Consider first the construction of a reduced space. Let $\boldsymbol{\mu}_1, \dots, \boldsymbol{\mu}_m \in \mathcal{D}$, and let $\boldsymbol{X}(\boldsymbol{\mu}_1), \dots, \boldsymbol{X}(\boldsymbol{\mu}_m) \in \mathbb{R}^{N \times K}$ be the corresponding trajectories of length K. Applying proper orthogonal decomposition (POD) [9, 56] to the set of columns of the snapshot matrix $[\boldsymbol{X}(\boldsymbol{\mu}_1), \dots, \boldsymbol{X}(\boldsymbol{\mu}_m)] \in \mathbb{R}^{N \times mK}$ yields an orthonormal basis $\boldsymbol{v}_1, \dots, \boldsymbol{v}_n$, with $n \ll N$, that spans an n-dimensional subspace $\mathcal{V}_n \subset \mathbb{R}^N$. Let $\boldsymbol{V}_n = [\boldsymbol{v}_1, \dots, \boldsymbol{v}_n] \in \mathbb{R}^{N \times n}$ be the basis matrix that has as columns the basis vectors $\boldsymbol{v}_1, \dots, \boldsymbol{v}_n$. Note that $\boldsymbol{V}_n$ is independent of the parameter $\boldsymbol{\mu}$ in the following. There are other methods for constructing reduced spaces, such as greedy methods [46, 63] and interpolatory model reduction [4, 25, 2]. We refer the reader to [49, 9] for details on how to select the parameters $\boldsymbol{\mu}_1, \dots, \boldsymbol{\mu}_m$ and how to select the dimension n of the space $\mathcal{V}_n$.

The reduced operators are constructed via, e.g., Galerkin projection. In case of $\ell = 1$ and Galerkin projection, the reduced operators are

$$\tilde{\boldsymbol{A}}_1(\boldsymbol{\mu}_j) = \boldsymbol{V}_n^T \boldsymbol{A}_1(\boldsymbol{\mu}_j) \boldsymbol{V}_n\,, \quad \tilde{\boldsymbol{B}}(\boldsymbol{\mu}_j) = \boldsymbol{V}_n^T \boldsymbol{B}(\boldsymbol{\mu}_j)\,, \quad j = 1, \dots, m\,. \tag{2.3}$$

To introduce the reduced operators $\tilde{\boldsymbol{A}}_i(\boldsymbol{\mu}_j)$ for $i > 1$ and $j = 1, \dots, m$, consider the matrix $\mathsf{I}_N^{(i)} \in \{0,1\}^{N_i \times N^i}$ and $\mathsf{J}_n^{(i)} \in \{0,1\}^{n^i \times n_i}$ such that $\boldsymbol{z}^i = \mathsf{I}_N^{(i)}(\boldsymbol{z} \otimes \cdots \otimes \boldsymbol{z})$ for $\boldsymbol{z} \in \mathbb{R}^N$ and $\tilde{\boldsymbol{z}} \otimes \cdots \otimes \tilde{\boldsymbol{z}} = \mathsf{J}_n^{(i)} \tilde{\boldsymbol{z}}^i$ for $\tilde{\boldsymbol{z}} \in \mathbb{R}^n$, respectively. The Kronecker product is taken i times for $i = 2, \dots, \ell$, and n_i is defined as

$$n_i = \binom{n + i - 1}{i} . \tag{2.4}$$

Note that the ith power $\boldsymbol{z}^i$ of a vector $\boldsymbol{z}$ is defined in section 2.1. The reduced operator $\tilde{\boldsymbol{A}}_i(\boldsymbol{\mu}_j)$ is

$$\tilde{\boldsymbol{A}}_i(\boldsymbol{\mu}_j) = \boldsymbol{V}_n^T \boldsymbol{A}_i(\boldsymbol{\mu}_j) \mathsf{I}_N^{(i)} (\boldsymbol{V}_n \otimes \cdots \otimes \boldsymbol{V}_n) \mathsf{J}_n^{(i)} \in \mathbb{R}^{n \times n_i} , \qquad j = 1, \dots, m ,$$

where the Kronecker product is taken i times and $i = 2, \dots, \ell$. The reduced model for $\boldsymbol{\mu}_j$ is

$$\begin{aligned} \tilde{\boldsymbol{x}}_{k+1}(\boldsymbol{\mu}_j) &= \tilde{\boldsymbol{f}}(\tilde{\boldsymbol{x}}_k(\boldsymbol{\mu}_j), \boldsymbol{u}_k(\boldsymbol{\mu}_j); \boldsymbol{\mu}_j) \\ &= \sum_{i=1}^{\ell} \tilde{\boldsymbol{A}}_i(\boldsymbol{\mu}_j) \tilde{\boldsymbol{x}}_k^i(\boldsymbol{\mu}_j) + \tilde{\boldsymbol{B}}(\boldsymbol{\mu}_j) \boldsymbol{u}_k(\boldsymbol{\mu}_j) , \qquad k = 0, \dots, K-1 , \end{aligned} \tag{2.5}$$

with the reduced state $\tilde{\boldsymbol{x}}_k(\boldsymbol{\mu}_j) \in \mathbb{R}^n$ and its ith power $\tilde{\boldsymbol{x}}_k^i(\boldsymbol{\mu}_j) \in \mathbb{R}^{n_i}$ for $i \in \mathbb{N}$. The initial condition is $\tilde{\boldsymbol{x}}_0(\boldsymbol{\mu}_j) = \boldsymbol{V}_n^T \boldsymbol{x}_0(\boldsymbol{\mu}_j)$. Once the reduced models $\tilde{\boldsymbol{f}}(\cdot, \cdot; \boldsymbol{\mu}_1), \dots, \tilde{\boldsymbol{f}}(\cdot, \cdot; \boldsymbol{\mu}_m)$ are constructed for all m parameters $\boldsymbol{\mu}_1, \dots, \boldsymbol{\mu}_m$, a reduced model for $\boldsymbol{\mu} \in \mathcal{D}$ is derived by elementwise interpolation of the reduced operators corresponding to $\boldsymbol{\mu}_1, \dots, \boldsymbol{\mu}_m$. If the structure of the reduced operators is known, e.g., symmetry and positive definiteness, then this structure can be preserved in the interpolation. We refer the reader to [1, 40, 17] for details on interpolating reduced operators in model reduction.

2.3. Operator inference.

The traditional model reduction approach described in section 2.2 to construct a reduced model (2.5) is intrusive in the sense that the operators $\boldsymbol{A}_1(\boldsymbol{\mu}_j), \dots, \boldsymbol{A}_\ell(\boldsymbol{\mu}_j), \boldsymbol{B}(\boldsymbol{\mu}_j)$ of system (2.2) for $j = 1, \dots, m$ are required in the projection step (2.3) either in assembled form or implicitly through methods that return the action of the operators on given vectors. Operator inference is introduced in [44] to derive approximations of the reduced operators $\tilde{\boldsymbol{A}}_1(\boldsymbol{\mu}_j), \dots, \tilde{\boldsymbol{A}}_\ell(\boldsymbol{\mu}_j), \tilde{\boldsymbol{B}}(\boldsymbol{\mu}_j)$ from data of the high-dimensional system without requiring the high-dimensional operators $\boldsymbol{A}_1(\boldsymbol{\mu}_j), \dots, \boldsymbol{A}_\ell(\boldsymbol{\mu}_j), \boldsymbol{B}(\boldsymbol{\mu}_j)$.

2.3.1. Operator inference.

Operator inference proceeds in three steps. First, state trajectories $\boldsymbol{X}(\boldsymbol{\mu}_1), \dots, \boldsymbol{X}(\boldsymbol{\mu}_m)$ and $\boldsymbol{Y}(\boldsymbol{\mu}_1), \dots, \boldsymbol{Y}(\boldsymbol{\mu}_m)$ are obtained by querying the system (2.1) at parameters $\boldsymbol{\mu}_1, \dots, \boldsymbol{\mu}_m \in \mathcal{D}$ to derive a reduced space spanned by the columns of $\boldsymbol{V}_n = [\boldsymbol{v}_1, \dots, \boldsymbol{v}_n]$. Many of the basis construction techniques developed in traditional model reduction can be applied; see references given in section 2.2. In the following, we will use POD to construct $\boldsymbol{V}_n$ as described in section 2.2. The second step of operator inference is to project the trajectories onto the reduced space $\mathcal{V}_n$ spanned by the columns of $\boldsymbol{V}_n$ and so to obtain the projected trajectories

$$\breve{\boldsymbol{X}}(\boldsymbol{\mu}_j) = \boldsymbol{V}_n^T \boldsymbol{X}(\boldsymbol{\mu}_j) , \quad \breve{\boldsymbol{Y}}(\boldsymbol{\mu}_j) = \boldsymbol{V}_n^T \boldsymbol{Y}(\boldsymbol{\mu}_j) , \quad j = 1, \dots, m .$$

In the third step of operator inference, operators

$$\hat{\boldsymbol{A}}_1(\boldsymbol{\mu}_j) \in \mathbb{R}^{n \times n_1}, \dots, \hat{\boldsymbol{A}}_\ell(\boldsymbol{\mu}_j) \in \mathbb{R}^{n \times n_\ell}, \ \hat{\boldsymbol{B}}(\boldsymbol{\mu}_j) \in \mathbb{R}^{n \times p} \tag{2.6}$$

are learned via least-squares regression

$$\min_{\hat{\boldsymbol{A}}_1(\boldsymbol{\mu}_j),\dots,\hat{\boldsymbol{A}}_\ell(\boldsymbol{\mu}_j),\hat{\boldsymbol{B}}(\boldsymbol{\mu}_j)} \sum_{k=0}^{K-1} \left\| \sum_{i=1}^{\ell} \hat{\boldsymbol{A}}_i(\boldsymbol{\mu}_j)\breve{\boldsymbol{x}}_k^i(\boldsymbol{\mu}_j) + \hat{\boldsymbol{B}}(\boldsymbol{\mu}_j)\boldsymbol{u}_k(\boldsymbol{\mu}_j) - \breve{\boldsymbol{x}}_{k+1}(\boldsymbol{\mu}_j) \right\|_2^2 \tag{2.7}$$

to obtain the model

$$\begin{aligned} \hat{\boldsymbol{x}}_{k+1}(\boldsymbol{\mu}_j) &= \sum_{i=1}^{\ell} \hat{\boldsymbol{A}}_i(\boldsymbol{\mu}_j)\hat{\boldsymbol{x}}_k^i(\boldsymbol{\mu}_j) + \hat{\boldsymbol{B}}(\boldsymbol{\mu}_j)\boldsymbol{u}_k(\boldsymbol{\mu}_j) \\ &= \hat{\boldsymbol{f}}(\hat{\boldsymbol{x}}_k(\boldsymbol{\mu}_j), \boldsymbol{u}_k(\boldsymbol{\mu}_j); \boldsymbol{\mu}_j) \end{aligned} \tag{2.8}$$

for $j = 1, \dots, m$. Note that the least-squares problem (2.7) is solved for each parameter $\boldsymbol{\mu}_j$ with $j = 1, \dots, m$. The state of the learned model at time k is $\hat{\boldsymbol{x}}_k(\boldsymbol{\mu}_j) \in \mathbb{R}^n$ with its ith power $\hat{\boldsymbol{x}}_k^i(\boldsymbol{\mu}_j)$. Note that the state $\hat{\boldsymbol{x}}_k(\boldsymbol{\mu}_j)$ is obtained by time stepping the learned model (2.8), whereas the projected state $\breve{\boldsymbol{x}}_k(\boldsymbol{\mu}_j)$ is obtained by projecting the high-dimensional state $\boldsymbol{x}_k(\boldsymbol{\mu}_j)$ at time k onto the reduced space $\mathcal{V}_n$. The initial condition is $\hat{\boldsymbol{x}}_0(\boldsymbol{\mu}_j) = \boldsymbol{V}_n^T \boldsymbol{x}_0(\boldsymbol{\mu}_j)$. To obtain a model for $\boldsymbol{\mu} \in \mathcal{D}$, the operators of the learned models corresponding to $\boldsymbol{\mu}_1, \dots, \boldsymbol{\mu}_m$ are interpolated as in traditional model reduction; see section 2.2. We refer the reader to [44, 47, 58, 43] for details on operator inference.

2.3.2. Data matrix.

It will be convenient to write (2.7) for each $j = 1, \dots, m$ as

$$\min_{\hat{\boldsymbol{O}}(\boldsymbol{\mu}_j)} \left\| \breve{\boldsymbol{D}}^T(\boldsymbol{\mu}_j)\hat{\boldsymbol{O}}^T(\boldsymbol{\mu}_j) - \breve{\boldsymbol{Y}}^T(\boldsymbol{\mu}_j) \right\|_F^2 \tag{2.9}$$

with the data matrix

$$\breve{\boldsymbol{D}}(\boldsymbol{\mu}_j) = \begin{bmatrix} \breve{\boldsymbol{X}}(\boldsymbol{\mu}_j) \\ \breve{\boldsymbol{X}}^2(\boldsymbol{\mu}_j) \\ \vdots \\ \breve{\boldsymbol{X}}^\ell(\boldsymbol{\mu}_j) \\ \boldsymbol{U}(\boldsymbol{\mu}_j) \end{bmatrix} \in \mathbb{R}^{\left(p + \sum_{i=1}^{\ell} n_i\right) \times K} \tag{2.10}$$

and $\breve{\boldsymbol{X}}^i(\boldsymbol{\mu}_j) = [\breve{\boldsymbol{x}}_0^i(\boldsymbol{\mu}_j), \dots, \breve{\boldsymbol{x}}_{K-1}^i(\boldsymbol{\mu}_j)] \in \mathbb{R}^{n_i \times K}$ for $i = 1, \dots, \ell$ and $\boldsymbol{U}(\boldsymbol{\mu}_j) = [\boldsymbol{u}_0(\boldsymbol{\mu}_j), \dots, \boldsymbol{u}_{K-1}(\boldsymbol{\mu}_j)] \in \mathbb{R}^{p \times K}$. Note that $\| \cdot \|_F$ is the Frobenius norm in (2.9). The operator matrix is

$$\hat{\boldsymbol{O}}(\boldsymbol{\mu}_j) = \begin{bmatrix} \hat{\boldsymbol{A}}_1(\boldsymbol{\mu}_j) & \dots & \hat{\boldsymbol{A}}_\ell(\boldsymbol{\mu}_j) & \hat{\boldsymbol{B}}(\boldsymbol{\mu}_j) \end{bmatrix} \in \mathbb{R}^{n \times \left(p + \sum_{i=1}^{\ell} n_i\right)}.$$

2.4. Problem formulation.

Our goal is to exactly recover the reduced operators (2.3) from data of the high-dimensional system without having the high-dimensional operators (2.3) available in assembled or implicit form. The learned operators (2.6) obtained with operator inference from the projected trajectories, as described in section 2.3, equal the reduced operators (2.3) obtained with intrusive model reduction in the limit of $n \to N$ under certain conditions described in [44]. However, typically, one is interested in reduced models with $n \ll N$, in which case the learned operators can differ significantly from the reduced operators. To explain the origin of the difference between the reduced and the learned operators, consider the trajectory $\tilde{\boldsymbol{X}}(\boldsymbol{\mu}) = [\tilde{\boldsymbol{x}}_0(\boldsymbol{\mu}), \dots, \tilde{\boldsymbol{x}}_{K-1}(\boldsymbol{\mu})] \in \mathbb{R}^{n \times K}$ obtained by time stepping the reduced model (2.5) that is constructed with intrusive model reduction. Even if $\boldsymbol{x}_0(\boldsymbol{\mu}) \in \mathcal{V}_n$, and thus $\tilde{\boldsymbol{x}}_0(\boldsymbol{\mu}) = \breve{\boldsymbol{x}}_0(\boldsymbol{\mu})$, the projected trajectory $\breve{\boldsymbol{X}}(\boldsymbol{\mu})$ can be different from the trajectory $\tilde{\boldsymbol{X}}(\boldsymbol{\mu})$ of the reduced model; i.e., there is a nonzero closure error

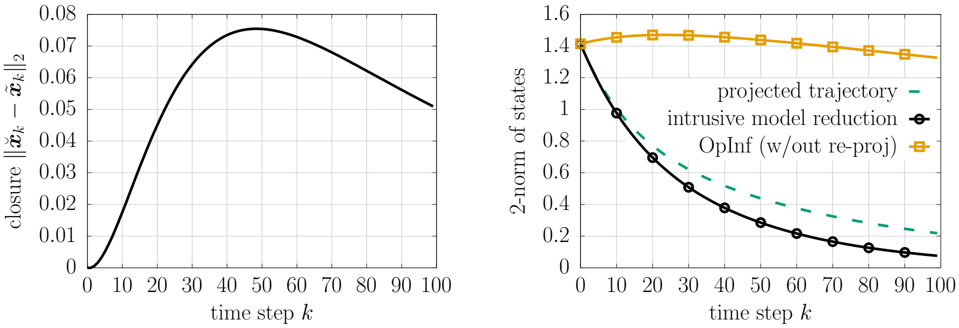
$$\|\breve{\boldsymbol{X}}(\boldsymbol{\mu}) - \tilde{\boldsymbol{X}}(\boldsymbol{\mu})\|_F \,. \tag{2.11}$$

By fitting operators to projected trajectories with operator inference as described in section 2.3 and in [44], the closure error (2.11) is introduced into the learned operators, which means that the learned operators can fail to approximate the dynamics of the reduced models constructed with intrusive model reduction.

Toy example. We demonstrate the effect of the closure error on operator inference with a toy example.[1] First, a symmetric negative definite matrix $\mathcal{A} \in \mathbb{R}^{N \times N}$ is generated with $N = 10$. The eigenvalues of the matrix $\mathcal{A}$ are logarithmically spaced between -10^{-1} and -10^{-2}. The eigenvectors are derived by orthonormalizing an $N \times N$ matrix that has standard normal entries; see Algorithm 3 for details. Then the time-continuous linear autonomous system $\dot{\boldsymbol{x}}(t) = \mathcal{A}\boldsymbol{x}(t)$ with system matrix $\mathcal{A}$ is discretized in time with the explicit Runge–Kutta scheme of fourth order, which leads to an autonomous (time-discrete) system $\boldsymbol{x}_{k+1} = \boldsymbol{A}_1\boldsymbol{x}_k$ of the form (2.2) with degree $\ell = 1$ and system matrix $\boldsymbol{A}_1$. The time-step size is 1, and the number of time steps is $K = 100$. The system $\boldsymbol{x}_{k+1} = \boldsymbol{A}_1\boldsymbol{x}_k$ is asymptotically stable. The initial condition $\boldsymbol{x}_0$ is the first column of the identity matrix of dimension $N \times N$, and $\boldsymbol{X}$ is the corresponding trajectory of length K. Set $n = 2$, and consider the two-dimensional space $\mathcal{V}_n$ that is spanned by the initial condition and the canonical unit vector with 1 at component 2. Let $\boldsymbol{V}_n$ be the corresponding basis matrix, and let $\breve{\boldsymbol{X}} = \boldsymbol{V}_n^T\boldsymbol{X}$ be the projected trajectory. The reduced model obtained with intrusive model reduction is $\tilde{\boldsymbol{x}}_{k+1} = \tilde{\boldsymbol{A}}_1\tilde{\boldsymbol{x}}_k$ with $\tilde{\boldsymbol{A}}_1 = \boldsymbol{V}_n^T\boldsymbol{A}_1\boldsymbol{V}_n$, $\tilde{\boldsymbol{x}}_0 = \boldsymbol{V}_n^T\boldsymbol{x}_0 = [1, 0]^T$, and the trajectory $\tilde{\boldsymbol{X}}$. The reduced model is asymptotically stable. Figure 1(a) shows the difference $\|\breve{\boldsymbol{x}}_k - \tilde{\boldsymbol{x}}_k\|_2$ for time steps $k = 0, \dots, K-1$, which is the 2-norm of the difference of the projected state and the state of the reduced model at time step k. Consider now the learned model obtained with operator inference from the projected trajectory $\breve{\boldsymbol{X}}$ as described in section 2.3. The learned model is asymptotically stable. We consider an initial condition $\boldsymbol{x}_0^{\text{test}} = [1, 1, 0, \dots, 0]^T \in \mathcal{V}_n$, which is different from the initial condition $\boldsymbol{x}_0$ used for training, and then time step the learned model with initial condition $\boldsymbol{V}_n^T\boldsymbol{x}_0^{\text{test}}$ to obtain the corresponding trajectory $\hat{\boldsymbol{X}}^{\text{test}}$. The trajectory $\hat{\boldsymbol{X}}^{\text{test}}$ obtained with the learned model differs from the trajectory $\tilde{\boldsymbol{X}}^{\text{test}}$ of the reduced model obtained with the same initial condition $\boldsymbol{V}_n^T\boldsymbol{x}_0^{\text{test}}$ as shown in Figure 1(b). The figure also shows that the trajectory $\hat{\boldsymbol{X}}^{\text{test}}$ of the learned model differs from the projected trajectory $\breve{\boldsymbol{X}}^{\text{test}} = \boldsymbol{V}_n^T\boldsymbol{X}^{\text{test}}$ that is obtained by projecting the trajectory $\boldsymbol{X}^{\text{test}}$ of the high-dimensional system $\boldsymbol{x}_{k+1} = \boldsymbol{A}_1\boldsymbol{x}_k$ with initial condition $\boldsymbol{x}_0^{\text{test}}$.

The toy example demonstrates that the closure error can have a significant polluting effect on fitting operators to projected trajectories and so lead to models that exhibit dynamics different from the corresponding reduced models obtained with intrusive model reduction. Thus, if the aim is to learn from data the same reduced models that intrusive model reduction constructs, then there is a need for revising operator inference to guarantee the recovery of the reduced operators (2.3) from trajectories with finite length $K < \infty$ and for dimensions $n \ll N$.

[1] MATLAB code for this example is available online from https://github.com/pehersto/reproj.



(a) closure error for training initial condition $\boldsymbol{x}_0$ (b) trajectories for test initial condition $\boldsymbol{x}_0^{\text{test}}$

FIG. 1. *Toy example: The closure error* (2.11) *pollutes operators that are fitted to projected trajectories with operator inference, which can lead to models that only poorly approximate the corresponding reduced models that are obtained with intrusive model reduction.*

3. Sampling Markovian dynamics via re-projection.

We present a data sampling scheme that generates trajectories $\bar{\boldsymbol{X}}(\boldsymbol{\mu}) \in \mathbb{R}^{n\times K}$ from the high-dimensional system (2.1) so that the closure error $\|\bar{\boldsymbol{X}}(\boldsymbol{\mu}) - \tilde{\boldsymbol{X}}(\boldsymbol{\mu})\|_F = 0$ is zero. Applying operator inference to these trajectories $\bar{\boldsymbol{X}}(\boldsymbol{\mu})$ of sufficient length $K < \infty$ exactly recovers the corresponding reduced model obtained with intrusive model reduction under certain conditions. In section 3.1, we build on the Mori–Zwanzig formalism [19, 16] to clarify that the closure error (2.11) corresponds to non-Markovian dynamics of the projected trajectories in $\mathcal{V}_n$. These non-Markovian dynamics cannot be represented by a reduced model of the form (2.5). Section 3.2 describes a data sampling scheme that cancels these non-Markovian dynamics after each time step to obtain trajectories that correspond to Markovian dynamics in the reduced space $\mathcal{V}_n$. Section 3.3 shows that these re-projected trajectories equal the trajectories that are obtained with a reduced model and thus that operator inference exactly recovers the reduced model from these re-projected trajectories. Note that a requirement of the proposed data sampling scheme is that the high-dimensional system (2.1) can be time stepped at selected initial conditions; more details on requirements and limitations of the proposed data sampling scheme follow in section 4.

In this section, we focus on learning reduced models corresponding to a single parameter $\boldsymbol{\mu}_j$, which then is subsequently repeated for all parameters $j = 1, \dots, m$. To ease exposition, we drop the dependence on $\boldsymbol{\mu}_j$ in this section.

3.1. Non-Markovian dynamics of projected states.

To motivate our data sampling scheme, we first discuss the closure error $\|\breve{\boldsymbol{X}} - \tilde{\boldsymbol{X}}\|_F$ on the toy example given in the problem formulation in section 2.4. The arguments in this section are not new; we refer the reader to the literature from the statistical physics community on the Mori–Zwanzig formalism, which describes the arguments in this section for more general systems and in stochastic settings; see, e.g., the surveys [19, 16] for more details.

Our toy example is an autonomous linear system, which corresponds to system (2.1) with $\ell = 1$ and $\boldsymbol{B} = \boldsymbol{0}$, i.e.,

$$\boldsymbol{x}_{k+1} = \boldsymbol{A}_1 \boldsymbol{x}_k\,, \qquad k = 0, \dots, K-1\,. \tag{3.1}$$

Consider now the orthogonal complement $\mathcal{V}_n^\perp$ of $\mathcal{V}_n$ that is spanned by the orthonormal columns of the basis matrix $\boldsymbol{V}_n^\perp \in \mathbb{R}^{N\times N-n}$ such that $\mathbb{R}^N = \mathcal{V}_n \oplus \mathcal{V}_n^\perp$. Define

$\boldsymbol{x}_k^{\parallel} = \boldsymbol{V}_n^T \boldsymbol{x}_k$ and $\boldsymbol{x}_k^{\perp} = (\boldsymbol{V}_n^{\perp})^T \boldsymbol{x}_k$ so that $\boldsymbol{x}_k = \boldsymbol{V}_n \boldsymbol{x}_k^{\parallel} + \boldsymbol{V}_n^{\perp} \boldsymbol{x}_k^{\perp}$. Correspondingly, (3.1) is split into

$$\begin{aligned} \boldsymbol{x}_{k+1}^{\parallel} &= \boldsymbol{A}_1^{\parallel\parallel} \boldsymbol{x}_k^{\parallel} + \boldsymbol{A}_1^{\parallel\perp} \boldsymbol{x}_k^{\perp}, \\ \boldsymbol{x}_{k+1}^{\perp} &= \boldsymbol{A}_1^{\perp\parallel} \boldsymbol{x}_k^{\parallel} + \boldsymbol{A}_1^{\perp\perp} \boldsymbol{x}_k^{\perp} \end{aligned}$$

with the matrices

$$\boldsymbol{A}_1^{\parallel\parallel} = \boldsymbol{V}_n^T \boldsymbol{A}_1 \boldsymbol{V}_n\,,\ \boldsymbol{A}_1^{\parallel\perp} = \boldsymbol{V}_n^T \boldsymbol{A}_1 \boldsymbol{V}_n^{\perp}\,,\ \boldsymbol{A}_1^{\perp\parallel} = (\boldsymbol{V}_n^{\perp})^T \boldsymbol{A}_1 \boldsymbol{V}_n\,,\ \boldsymbol{A}_1^{\perp\perp} = (\boldsymbol{V}_n^{\perp})^T \boldsymbol{A}_1 \boldsymbol{V}_n^{\perp}\,.$$

Model reduction as described in section 2.2 constructs the reduced operator $\tilde{\boldsymbol{A}}_1 = \boldsymbol{A}_1^{\parallel\parallel}$ via projection. Consider now the trajectory $\boldsymbol{X} = [\boldsymbol{x}_0, \boldsymbol{x}_1, \dots, \boldsymbol{x}_{K-1}]$ and its projection $\breve{\boldsymbol{X}} = [\breve{\boldsymbol{x}}_0, \dots, \breve{\boldsymbol{x}}_{K-1}]$ with $\boldsymbol{V}_n$. Then we obtain

$$\boldsymbol{V}_n^T \boldsymbol{x}_{k+1} = \breve{\boldsymbol{x}}_{k+1} = \boldsymbol{x}_{k+1}^{\parallel} = \boldsymbol{A}_1^{\parallel\parallel} \boldsymbol{x}_k^{\parallel} + \boldsymbol{A}_1^{\parallel\perp} \boldsymbol{x}_k^{\perp}\,, \qquad k = 0, \dots, K-1\,,$$

which gives with an inductive argument that

$$\breve{\boldsymbol{x}}_{k+1} = \boldsymbol{x}_{k+1}^{\parallel} = \underbrace{\boldsymbol{A}_1^{\parallel\parallel} \boldsymbol{x}_k^{\parallel}}_{\text{Markovian term}} \underbrace{+ \boldsymbol{A}_1^{\parallel\perp} \sum_{i=0}^{k-1} \left(\boldsymbol{A}_1^{\perp\perp}\right)^{k-1-i} \boldsymbol{A}_1^{\perp\parallel} \boldsymbol{x}_i^{\parallel}}_{\text{non-Markovian term}} + \boldsymbol{A}_1^{\parallel\perp} \left(\boldsymbol{A}_1^{\perp\perp}\right)^k \boldsymbol{x}_0^{\perp}\,.$$

Thus, the projected state $\breve{\boldsymbol{x}}_{k+1} = \boldsymbol{x}_{k+1}^{\parallel}$ at time $k+1$ depends on the time history of projected states $\boldsymbol{x}_0^{\parallel}, \boldsymbol{x}_1^{\parallel}, \dots, \boldsymbol{x}_k^{\parallel}$ instead of only on the previous time step $\boldsymbol{x}_k^{\parallel}$. This means that the dynamics described by the trajectory $\boldsymbol{X}$ become non-Markovian if projected onto the reduced space $\mathcal{V}_n$ in the sense that going from $\boldsymbol{x}_k^{\parallel}$ to $\boldsymbol{x}_{k+1}^{\parallel}$ requires knowledge of the time history $\boldsymbol{x}_0^{\parallel}, \dots, \boldsymbol{x}_{k-1}^{\parallel}$ in general. Therefore, the reduced model (2.5), which is derived with traditional model reduction, cannot describe well the projected trajectory $\breve{\boldsymbol{X}}$ because the reduced model (2.5) is Markovian in the sense that the state $\tilde{\boldsymbol{x}}_{k+1}$ at time step $k+1$ depends on the state $\tilde{\boldsymbol{x}}_k$ of the previous time step k alone, instead of on the history $\tilde{\boldsymbol{x}}_0, \dots, \tilde{\boldsymbol{x}}_{k-1}$.

3.2. Data sampling with re-projection to avoid non-Markovian dynamics. We now describe our sampling scheme with re-projection. Consider an initial condition $\boldsymbol{x}_0 \in \mathcal{V}_n$ and set $\bar{\boldsymbol{x}}_0 = \boldsymbol{V}_n^T \boldsymbol{x}_0$. Note that $\boldsymbol{V}_n \bar{\boldsymbol{x}}_0 = \boldsymbol{x}_0$ because $\boldsymbol{x}_0 \in \mathcal{V}_n$. Our scheme proceeds iteratively; see Figure 2. In the first iteration, system (2.1) is queried at initial condition $\boldsymbol{V}_n \bar{\boldsymbol{x}}_0$ and input $\boldsymbol{u}_0$ to obtain

$$\boldsymbol{x}_{\text{tmp}} = \boldsymbol{f}(\boldsymbol{V}_n \bar{\boldsymbol{x}}_0, \boldsymbol{u}_0)\,.$$

Then the re-projected state $\bar{\boldsymbol{x}}_1 = \boldsymbol{V}_n^T \boldsymbol{x}_{\text{tmp}}$ is computed by projecting $\boldsymbol{x}_{\text{tmp}}$ onto $\mathcal{V}_n$. In the second iteration, system (2.1) is queried for a single time step at the initial condition $\boldsymbol{V}_n \bar{\boldsymbol{x}}_1$ and input $\boldsymbol{u}_1$ to obtain $\boldsymbol{f}(\boldsymbol{V}_n \bar{\boldsymbol{x}}_1, \boldsymbol{u}_1)$ and to compute $\bar{\boldsymbol{x}}_2$ via projection $\bar{\boldsymbol{x}}_2 = \boldsymbol{V}_n^T \boldsymbol{f}(\boldsymbol{V}_n \bar{\boldsymbol{x}}_1, \boldsymbol{u}_1)$. This process is repeated to generate the re-projected states $\bar{\boldsymbol{x}}_0, \bar{\boldsymbol{x}}_1, \dots, \bar{\boldsymbol{x}}_K$ and to collect them into the re-projected trajectories $\bar{\boldsymbol{X}} = [\bar{\boldsymbol{x}}_0, \bar{\boldsymbol{x}}_1, \dots, \bar{\boldsymbol{x}}_{K-1}]$ and $\bar{\boldsymbol{Y}} = [\bar{\boldsymbol{x}}_1, \dots, \bar{\boldsymbol{x}}_K]$.

Algorithm 1 summarizes our data sampling scheme with re-projection. The inputs to Algorithm 1 are the high-dimensional system $\boldsymbol{f}$, a basis matrix $\boldsymbol{V}_n$, an initial condition $\boldsymbol{x}_0 \in \mathcal{V}_n$, a parameter $\boldsymbol{\mu} \in \mathcal{D}$, and inputs $\boldsymbol{u}_0, \dots, \boldsymbol{u}_{K-1}$. Line 2 projects the initial condition $\boldsymbol{x}_0$ to obtain $\bar{\boldsymbol{x}}_0$. The `for` loop on line 3 iterates over the time steps $k = 0, \dots, K-1$ and generates the re-projected state $\bar{\boldsymbol{x}}_{k+1}$ by querying the high-dimensional system for a single time step in line 4. The re-projected trajectories $\bar{\boldsymbol{X}}$ and $\bar{\boldsymbol{Y}}$ are returned in line 7.

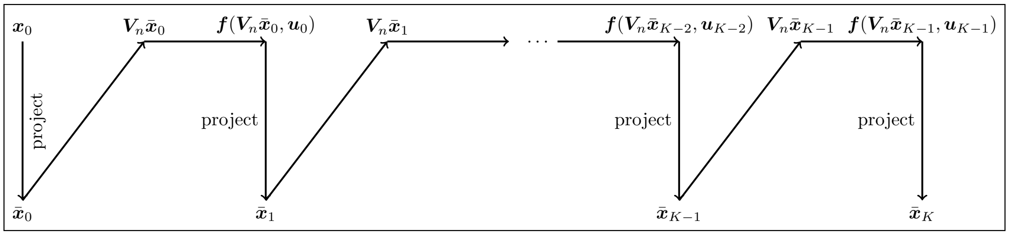

$\boldsymbol{x}_0$
$\boldsymbol{V}_n\bar{\boldsymbol{x}}_0$
$\boldsymbol{f}(\boldsymbol{V}_n\bar{\boldsymbol{x}}_0, \boldsymbol{u}_0)$
$\boldsymbol{V}_n\bar{\boldsymbol{x}}_1$
$\cdots$
$\boldsymbol{f}(\boldsymbol{V}_n\bar{\boldsymbol{x}}_{K-2}, \boldsymbol{u}_{K-2})$
$\boldsymbol{V}_n\bar{\boldsymbol{x}}_{K-1}$
$\boldsymbol{f}(\boldsymbol{V}_n\bar{\boldsymbol{x}}_{K-1}, \boldsymbol{u}_{K-1})$
project
project
project
project
$\bar{\boldsymbol{x}}_0$
$\bar{\boldsymbol{x}}_1$
$\bar{\boldsymbol{x}}_{K-1}$
$\bar{\boldsymbol{x}}_K$

FIG. 2. *The scheme shows data sampling with re-projection. Under certain conditions that are discussed in section* 3.3*, the re-projected trajectories $\bar{\boldsymbol{X}} = [\bar{\boldsymbol{x}}_0, \bar{\boldsymbol{x}}_1, \dots, \bar{\boldsymbol{x}}_{K-1}]$ are the trajectories $\tilde{\boldsymbol{X}}$ of the reduced models that are obtained with intrusive model reduction. Thus, the closure error* (2.11) *of the re-projected trajectories is zero.*

Algorithm 1. Data sampling with re-projection.

```
procedure $\textsc{ReProj}(\boldsymbol{f}, \boldsymbol{V}_n, \boldsymbol{x}_0, \boldsymbol{u}_0, \dots, \boldsymbol{u}_{K-1})$
    Set $\bar{\boldsymbol{x}}_0 = \boldsymbol{V}_n^T \boldsymbol{x}_0$
    for $k = 0, \dots, K-1$ do
        Query system for a single time step $\boldsymbol{x}_{\text{tmp}} = \boldsymbol{f}(\boldsymbol{V}_n \bar{\boldsymbol{x}}_k, \boldsymbol{u}_k)$
        Set $\bar{\boldsymbol{x}}_{k+1} = \boldsymbol{V}_n^T \boldsymbol{x}_{\text{tmp}}$
    end for
    Return $\bar{\boldsymbol{X}} = [\bar{\boldsymbol{x}}_0, \bar{\boldsymbol{x}}_1, \dots, \bar{\boldsymbol{x}}_{K-1}]$ and $\bar{\boldsymbol{Y}} = [\bar{\boldsymbol{x}}_1, \dots, \bar{\boldsymbol{x}}_K]$
end procedure
```

3.3. Exact recovery of reduced models from re-projected trajectories.

Proposition 3.1 shows that the trajectories $\bar{\boldsymbol{X}}$ and $\bar{\boldsymbol{Y}}$ obtained with sampling with re-projection are the trajectories $\tilde{\boldsymbol{X}}$ and $\tilde{\boldsymbol{Y}}$ obtained from time stepping the corresponding reduced models. Proposition 3.1 leads to Corollary 3.2, which shows that the reduced operators (2.3) from intrusive model reduction are exactly recovered from re-projected trajectories in the sense that $\|\tilde{\boldsymbol{A}}_i - \hat{\boldsymbol{A}}_i\|_F = \|\tilde{\boldsymbol{B}} - \hat{\boldsymbol{B}}\|_F = 0$ for $i = 1, \dots, \ell$ under certain conditions. This is a preasymptotic result in the sense that it holds for finite-length trajectories, i.e., for finite number of data points, and for reduced spaces $\mathcal{V}_n$ of dimensions $n < N$.

PROPOSITION 3.1. *Consider a system* (2.2) *with polynomial nonlinear terms. Let $\boldsymbol{x}_0 \in \mathcal{V}_n$ be an initial condition, and let $\boldsymbol{u}_0, \dots, \boldsymbol{u}_{K-1}$ be inputs. Generate trajectories $\bar{\boldsymbol{X}}$ and $\bar{\boldsymbol{Y}}$ from system* (2.2) *with re-projection as described in Algorithm* 1*. Then $\bar{\boldsymbol{X}} = \tilde{\boldsymbol{X}}$ and $\bar{\boldsymbol{Y}} = \tilde{\boldsymbol{Y}}$ hold, where $\tilde{\boldsymbol{X}} = [\tilde{\boldsymbol{x}}_0, \dots, \tilde{\boldsymbol{x}}_{K-1}]$ and $\tilde{\boldsymbol{Y}} = [\tilde{\boldsymbol{x}}_1, \dots, \tilde{\boldsymbol{x}}_K]$ are trajectories obtained with the reduced model* (2.5) *from intrusive model reduction with initial condition $\tilde{\boldsymbol{x}}_0 = \boldsymbol{V}_n^T \boldsymbol{x}_0$ and inputs $\boldsymbol{u}_0, \dots, \boldsymbol{u}_{K-1}$.*

Proof. With zero padding, the operators $\boldsymbol{A}_1, \dots, \boldsymbol{A}_\ell$ of (2.2) can be reformulated to $\mathsf{A}_1, \dots, \mathsf{A}_\ell$ so that

$$\boldsymbol{x}_{k+1} = \sum_{i=1}^{\ell} \mathsf{A}_i(\underbrace{\boldsymbol{x}_k \otimes \cdots \otimes \boldsymbol{x}_k}_{i-\text{times}}) + \boldsymbol{B}\boldsymbol{u}_k\,, \qquad k = 0, \dots, K-1\,,$$

where $\otimes$ is the Kronecker product. Note that $\mathsf{A}_i \in \mathbb{R}^{N \times N^i}$ for $i = 1, \dots, \ell$. Similarly, operators $\tilde{\mathsf{A}}_1, \dots, \tilde{\mathsf{A}}_\ell$ are obtained via projection

$$\tilde{\mathsf{A}}_i = \boldsymbol{V}_n^T \mathsf{A}_i(\underbrace{\boldsymbol{V}_n \otimes \cdots \otimes \boldsymbol{V}_n}_{i-\text{times}})\,, \qquad i = 1, \dots, \ell\,,$$

so that $\tilde{\boldsymbol{A}}_i\tilde{\boldsymbol{x}}_k^i = \tilde{\mathsf{A}}_i(\tilde{\boldsymbol{x}}_k \otimes \cdots \otimes \tilde{\boldsymbol{x}}_k)$ holds for $i = 1, \dots, \ell$. Set $\bar{\boldsymbol{x}}_0 = \boldsymbol{V}_n^T\boldsymbol{x}_0$, and note that $\boldsymbol{x}_0 \in \mathcal{V}_n$ and thus $\bar{\boldsymbol{x}}_0 = \tilde{\boldsymbol{x}}_0$. Querying system (2.2) at initial condition $\boldsymbol{V}_n\bar{\boldsymbol{x}}_0 = \boldsymbol{x}_0$ as described in line 4 of Algorithm 1 leads to

$$\begin{align}
\boldsymbol{x}_{\text{tmp}} &= \sum_{i=1}^{\ell} \boldsymbol{A}_i\boldsymbol{x}_0^k + \boldsymbol{B}\boldsymbol{u}_0 \notag\\
&= \sum_{i=1}^{\ell} \mathsf{A}_i(\boldsymbol{x}_0 \otimes \cdots \otimes \boldsymbol{x}_0) + \boldsymbol{B}\boldsymbol{u}_0 \notag\\
&= \sum_{i=1}^{\ell} \mathsf{A}_i(\boldsymbol{V}_n\tilde{\boldsymbol{x}}_0 \otimes \cdots \otimes \boldsymbol{V}_n\tilde{\boldsymbol{x}}_0) + \boldsymbol{B}\boldsymbol{u}_0 \tag{3.2}\\
&= \sum_{i=1}^{\ell} \mathsf{A}_i(\boldsymbol{V}_n \otimes \cdots \otimes \boldsymbol{V}_n)(\tilde{\boldsymbol{x}}_0 \otimes \cdots \otimes \tilde{\boldsymbol{x}}_0) + \boldsymbol{B}\boldsymbol{u}_0\,, \tag{3.3}
\end{align}$$

where we used that $\boldsymbol{x}_0 = \boldsymbol{V}_n\tilde{\boldsymbol{x}}_0$ in (3.2) and where we exploited the mixed-product property of the Kronecker product in (3.3). We now project (3.3) to obtain

$$\begin{aligned}
\boldsymbol{V}_n^T\boldsymbol{x}_{\text{tmp}} &= \sum_{i=1}^{\ell} \boldsymbol{V}_n^T\mathsf{A}_i(\boldsymbol{V}_n \otimes \cdots \otimes \boldsymbol{V}_n)(\tilde{\boldsymbol{x}}_0 \otimes \cdots \otimes \tilde{\boldsymbol{x}}_0) + \boldsymbol{V}_n^T\boldsymbol{B}\boldsymbol{u}_0\\
&= \sum_{i=1}^{\ell} \tilde{\mathsf{A}}_i(\tilde{\boldsymbol{x}}_0 \otimes \cdots \otimes \tilde{\boldsymbol{x}}_0) + \tilde{\boldsymbol{B}}\boldsymbol{u}_0,
\end{aligned}$$

and thus $\tilde{\boldsymbol{x}}_1 = \boldsymbol{V}_n^T\boldsymbol{x}_{\text{tmp}}$. According to line 5 in Algorithm 1, the re-projected state is $\bar{\boldsymbol{x}}_1 = \boldsymbol{V}_n^T\boldsymbol{x}_{\text{tmp}}$, and thus $\tilde{\boldsymbol{x}}_1 = \bar{\boldsymbol{x}}_1$ holds. The same steps can be repeated for time step k with $\bar{\boldsymbol{x}}_k = \tilde{\boldsymbol{x}}_k$ to obtain $\tilde{\boldsymbol{x}}_{k+1} = \bar{\boldsymbol{x}}_{k+1}$. Then with induction it follows that $\bar{\boldsymbol{X}} = \tilde{\boldsymbol{X}}$ and $\bar{\boldsymbol{Y}} = \tilde{\boldsymbol{Y}}$ hold. □

COROLLARY 3.2. *Let the trajectories* $\bar{\boldsymbol{X}}$ *and* $\bar{\boldsymbol{Y}}$ *of length* K *be generated with Algorithm* 1 *from a system with polynomial nonlinear terms up to degree* ℓ*. Further, let*

$$K \geq p + \sum_{i=1}^{\ell} n_i\,, \tag{3.4}$$

with n_i *defined in* (2.4) *for* $i = 1, \dots, \ell$*. Consider the data matrix*

$$\bar{\boldsymbol{D}} = \begin{bmatrix} \bar{\boldsymbol{X}} \\ \bar{\boldsymbol{X}}^2 \\ \vdots \\ \bar{\boldsymbol{X}}^j \\ \boldsymbol{U} \end{bmatrix} \in \mathbb{R}^{\left(p+\sum_{i=1}^{\ell} n_i\right)\times K} \tag{3.5}$$

derived from the re-projected trajectory $\bar{\boldsymbol{X}}$*; cf. the data matrix* $\breve{\boldsymbol{D}}$ *derived from the projected trajectory* $\breve{\boldsymbol{X}}$ *defined in* (2.10)*. If* $\bar{\boldsymbol{D}}$ *has full rank, then the least-squares problem*

$$\min_{\hat{\boldsymbol{O}}} \|\bar{\boldsymbol{D}}^T\hat{\boldsymbol{O}}^T - \bar{\boldsymbol{Y}}^T\|_F^2 \tag{3.6}$$

has a unique solution $\hat{\boldsymbol{O}}^*$ *with objective* 0; *that solution is* $\hat{\boldsymbol{O}}^* = [\tilde{\boldsymbol{A}}_1, \tilde{\boldsymbol{A}}_2, \dots, \tilde{\boldsymbol{A}}_\ell, \tilde{\boldsymbol{B}}]$, *where* $\tilde{\boldsymbol{A}}_1, \dots, \tilde{\boldsymbol{A}}_\ell, \tilde{\boldsymbol{B}}$ *are the reduced operators* (2.3) *from intrusive model reduction.*

Proof. First, because of Proposition 3.1, we have $\tilde{\boldsymbol{X}} = \bar{\boldsymbol{X}}$ and $\tilde{\boldsymbol{Y}} = \bar{\boldsymbol{Y}}$, and thus the states of $\bar{\boldsymbol{X}}$ and $\bar{\boldsymbol{Y}}$ satisfy the equations corresponding to the reduced model (2.5) obtained with intrusive model reduction. This means that the matrix $\tilde{\boldsymbol{O}} = [\tilde{\boldsymbol{A}}_1, \tilde{\boldsymbol{A}}_2, \dots, \tilde{\boldsymbol{A}}_\ell, \tilde{\boldsymbol{B}}]$ is a solution of (3.6) because it achieves objective 0. To show uniqueness, note that (3.6) corresponds to $i = 1, \dots, n$ independent least-squares problems

$$\min_{\hat{\boldsymbol{O}}_i} \|\bar{\boldsymbol{D}}\hat{\boldsymbol{O}}_i^T - \bar{\boldsymbol{Y}}_i^T\|_2^2\,, \tag{3.7}$$

with $\hat{\boldsymbol{O}} = [\hat{\boldsymbol{O}}_1^T, \dots, \hat{\boldsymbol{O}}_n^T]^T$ and $\bar{\boldsymbol{Y}} = [\bar{\boldsymbol{Y}}_1^T, \dots, \bar{\boldsymbol{Y}}_n^T]^T$. Each of the rows of $\hat{\boldsymbol{O}}$ has length $p + \sum_{i=1}^{\ell} n_i$ and thus each of the least-squares problems (3.7) has $p + \sum_{i=1}^{\ell} n_i$ unknowns. Condition (3.4) guarantees that the number of equations in each least-squares problem (3.7) is at least $K \geq p + \sum_{i=1}^{\ell} n_i$. Thus, if $\bar{\boldsymbol{D}}$ has full rank, then there is at most one solution that solves (3.7). Since $\tilde{\boldsymbol{O}}$ leads to objective 0, we obtain $\hat{\boldsymbol{O}}^* = \tilde{\boldsymbol{O}}$. □

A critical assumption of Corollary 3.2 is that the data matrix $\bar{\boldsymbol{D}}$ has full rank. In case of a low-rank $\bar{\boldsymbol{D}}$, the least-squares problem (3.6) is underdetermined and therefore uniqueness of the solution is lost. Regularizing the least-squares problem (3.6) is one way to enforce a unique solution if $\bar{\boldsymbol{D}}$ is low rank. A common choice is solving for the solution with the smallest Frobenius norm. Another option is to add to the least-squares problem a sparsity-promoting regularizer such as an l_1 norm of the inferred operators. Such sparsity-promoting regularizers have been successfully used in the context of other data-driven modeling methods, such as SINDy [11, 50] and related methods [51, 52]. However, in case of operator inference with re-projection, where the aim is to derive small, dense matrices, it remains unclear what the effect of a sparsity-promoting regularizer is on the inferred reduced operators. Yet another option is to reduce the dimension n of the reduced space until the corresponding data matrix becomes full rank and Corollary 3.2 becomes applicable. Reducing the dimension n restricts the set of feasible solutions of the least-squares problem and so has a similar effect as adding a regularization term.

3.4. Interpretation and Petrov–Galerkin reduced models. The reduced model (2.5) can be written as

$$\tilde{\boldsymbol{x}}_{k+1} = \boldsymbol{V}_n^T \boldsymbol{f}(\boldsymbol{V}_n\tilde{\boldsymbol{x}}_k, \boldsymbol{u}_k)\,, \qquad k = 0, \dots, K-1\,, \tag{3.8}$$

where $\boldsymbol{f}$ is the full-model right-hand side function. Consider now a Petrov–Galerkin reduced model, where the test space is different from the trial space [9]. Let the columns of the orthonormal matrix $\boldsymbol{W}_n \in \mathbb{R}^{N \times n}$ span the test space, and let $\boldsymbol{V}_n$ denote the basis matrix for the trial space. The superior behavior of reduced models based on Petrov–Galerkin rather than on Galerkin projection has been shown in terms of stability and long-time integration in many applications; see, e.g., [9, 12] and the references therein. Following (3.8), the Petrov–Galerkin reduced model corresponds to

$$\tilde{\boldsymbol{x}}_{k+1} = (\boldsymbol{W}_n^T\boldsymbol{V}_n)^{-1}\boldsymbol{W}_n^T \boldsymbol{f}(\boldsymbol{V}_n\tilde{\boldsymbol{x}}_k, \boldsymbol{u}_k)\,, \qquad k = 0, \dots, K-1\,,$$

so that one can extend the arguments made in the proof of Proposition 3.1 to Petrov–Galerkin projection as well. However, constructing the test basis matrix $\boldsymbol{W}_n$ is often more involved than constructing $\boldsymbol{V}_n$. In particular, methods for constructing optimal test bases can require quantities derived from the full-model operators in certain situations [13], which is in contrast to the trial basis matrix $\boldsymbol{V}_n$ that typically is derived from snapshots alone. The data sampling with re-projection as described in section 3.2 is restricted to settings where the matrices $\boldsymbol{W}_n$ and $\boldsymbol{V}_n$ are available.

4. Computational procedure and practical aspects.

This section summarizes the overall computational procedure of operator inference with re-projected trajectories in Algorithm 2 and discusses practical aspects as well as limitations of the approach.

4.1. Computational procedure.

Algorithm 2 summarizes the overall procedure of recovering reduced models from re-projected trajectories with operator inference. The inputs to Algorithm 2 are $\boldsymbol{f}$, the degree ℓ, the dimension n of the reduced space, the parameters $\boldsymbol{\mu}_1, \dots, \boldsymbol{\mu}_m$, the initial conditions $\boldsymbol{x}_0(\boldsymbol{\mu}_1), \dots, \boldsymbol{x}_0(\boldsymbol{\mu}_m)$, and the input trajectories $\boldsymbol{U}(\boldsymbol{\mu}_1), \dots, \boldsymbol{U}(\boldsymbol{\mu}_m)$. Algorithm 2 time steps the high-dimensional system to obtain the trajectories $\boldsymbol{X}(\boldsymbol{\mu}_1), \dots, \boldsymbol{X}(\boldsymbol{\mu}_m)$ in the `for` loop on line 2. Then, in line 5, the POD basis matrix $\boldsymbol{V}_n$ is computed from the snapshot matrix $[\boldsymbol{X}(\boldsymbol{\mu}_1), \dots, \boldsymbol{X}(\boldsymbol{\mu}_m)]$. The `for` loop in line 6 calls Algorithm 1 to generate the re-projected trajectories $\bar{\boldsymbol{X}}(\boldsymbol{\mu}_1), \dots, \bar{\boldsymbol{X}}(\boldsymbol{\mu}_m)$ and $\bar{\boldsymbol{Y}}(\boldsymbol{\mu}_1), \dots, \bar{\boldsymbol{Y}}(\boldsymbol{\mu}_m)$. Operator inference as described in Corollary 3.2 is then applied to the re-projected trajectories in lines 8 and 9 to learn operators. Line 11 returns the learned operators.

The computational costs of Algorithm 2 are typically dominated by querying the high-dimensional system. The costs of assembling the data matrix on line 8 and the costs of solving the corresponding least-squares problem on line 9 typically are negligible. In the `for` loop in line 2, the high-dimensional system is time stepped to generate the trajectories for constructing the POD basis matrix, which is similar to traditional, intrusive model reduction. The `for` loop in line 6 requires time stepping the high-dimensional systems once more to sample the re-projected trajectories with Algorithm 1. Thus, re-projection incurs additional costs for generating the re-projected trajectories compared to operator inference without re-projection. The additional costs depend on the number of time steps K used for generating the re-projected trajectories. The requirement on K is given in (3.4) in Corollary 3.2 and depends on the degree ℓ of the system, the dimension n of the reduced space, and the number of inputs p.

4.2. Practical aspects and condition of least-squares problem.

We make three remarks on practical aspects of operator inference with re-projection. First, Corollary 3.2 states that operator inference from re-projected trajectories leads to the reduced models from intrusive model reduction if condition (3.4) is satisfied and if the data matrix $\bar{\boldsymbol{D}}$ defined in (3.5) has full rank. One can numerically verify these two conditions in practice to determine whether the same model as with intrusive model reduction is obtained up to numerical errors. It is important to add that typically numerically computing the rank of a matrix depends on a tolerance for what is considered a zero singular value of the matrix. For example, the tolerances used in the implementation of the `matrix_rank` procedure of numpy[2] and the `rank`[3] method of MATLAB take the size of the matrix and the floating-point relative accuracy into account. Thus, such a numerical check of the conditions of Corollary 3.2 can only be a heuristic for practical purposes and might fail if tolerances are not set properly.

[2] https://docs.scipy.org/doc/numpy/reference/generated/numpy.linalg.matrix_rank.html

[3] https://www.mathworks.com/help/matlab/ref/rank.html

Algorithm 2. Operator inference with re-projected trajectories.

```
procedure OpInfRP($\boldsymbol{f}, \ell, n, \boldsymbol{\mu}_1, \dots, \boldsymbol{\mu}_m, \boldsymbol{x}_0(\boldsymbol{\mu}_1), \dots, \boldsymbol{x}_0(\boldsymbol{\mu}_m), \boldsymbol{U}(\boldsymbol{\mu}_1), \dots, \boldsymbol{U}(\boldsymbol{\mu}_m)$)
    for $j = 1, \dots, m$ do
        Time step $\boldsymbol{f}$ at $\boldsymbol{\mu}_j$ with $\boldsymbol{x}_0(\boldsymbol{\mu}_j)$ and $\boldsymbol{U}(\boldsymbol{\mu}_j)$ to obtain $\boldsymbol{X}(\boldsymbol{\mu}_j)$
    end for
    Derive POD basis matrix $\boldsymbol{V}_n$ from snapshot matrix $[\boldsymbol{X}(\boldsymbol{\mu}_1), \dots, \boldsymbol{X}(\boldsymbol{\mu}_m)]$
    for $j = 1, \dots, m$ do
        Call Algorithm 1 with $\boldsymbol{V}_n, \boldsymbol{x}_0(\boldsymbol{\mu}_j), \boldsymbol{U}(\boldsymbol{\mu}_j)$ to obtain $\bar{\boldsymbol{X}}(\boldsymbol{\mu}_j)$ and $\bar{\boldsymbol{Y}}(\boldsymbol{\mu}_j)$
        Assemble data matrix $\bar{\boldsymbol{D}}(\boldsymbol{\mu}_j)$ defined in (3.5)
        Solve (3.6) to learn operators $\hat{\boldsymbol{A}}_1(\boldsymbol{\mu}_j), \dots, \hat{\boldsymbol{A}}_\ell(\boldsymbol{\mu}_j), \hat{\boldsymbol{B}}(\boldsymbol{\mu}_j)$
    end for
    Return learned operators $\hat{\boldsymbol{A}}_1(\boldsymbol{\mu}_j), \dots, \hat{\boldsymbol{A}}_\ell(\boldsymbol{\mu}_j), \hat{\boldsymbol{B}}(\boldsymbol{\mu}_j)$ for $j = 1, \dots, m$
end procedure
```

Second, to sample the re-projected trajectories with Algorithm 1, it is necessary to have available the high-dimensional system in the sense that it can be time stepped for a single time step with initial condition $\bar{\boldsymbol{x}}_k$ for $k = 0, \dots, K-1$. This is in contrast to operator inference without re-projection, which is applicable even if only the trajectories $\boldsymbol{X}(\boldsymbol{\mu}_1), \dots, \boldsymbol{X}(\boldsymbol{\mu}_m)$ and the corresponding inputs $\boldsymbol{U}(\boldsymbol{\mu}_1), \dots, \boldsymbol{U}(\boldsymbol{\mu}_m)$ are available and the high-dimensional system cannot be queried. However, note that it is unnecessary to time step the high-dimensional system at arbitrary initial conditions. The re-projected states are close to the states of the high-dimensional system if the space $\mathcal{V}_n$ is sufficiently rich, which typically is a necessary requirement for the success of model reduction in any case.

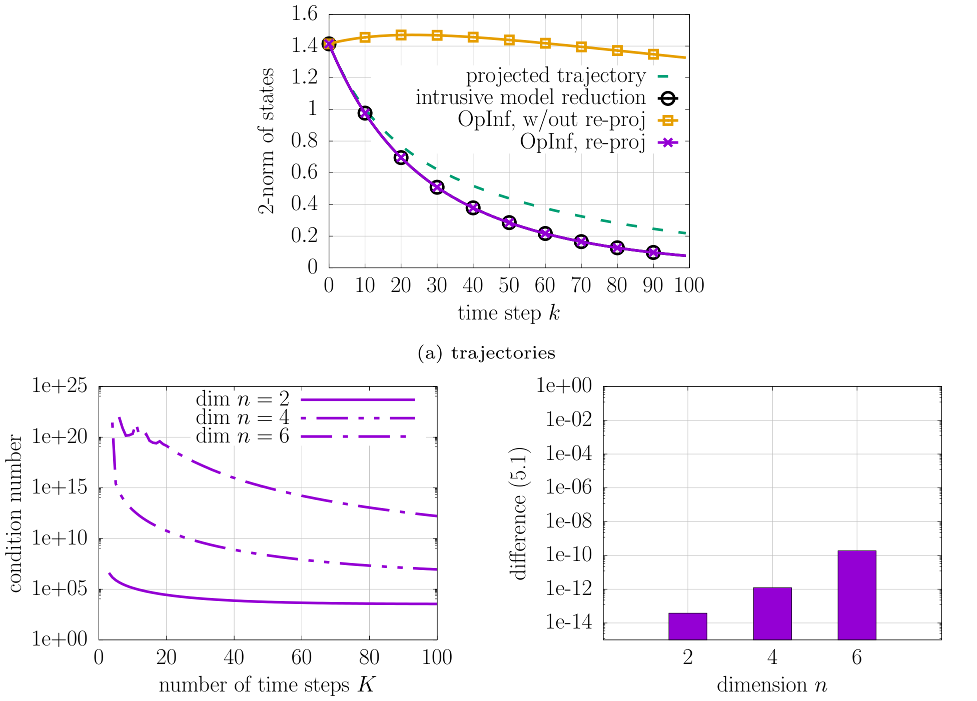
Third, in practice, the condition number of $\bar{\boldsymbol{D}}^T(\boldsymbol{\mu}_j)\bar{\boldsymbol{D}}(\boldsymbol{\mu}_j)$, $j = 1, \dots, m$, can be high, which means that numerical errors are amplified and pollute the learned operators even if all conditions required for Corollary 3.2 are satisfied. It has been empirically observed in [44, section 3.6] that concatenating multiple trajectories corresponding to different inputs helps to keep the condition number of $\bar{\boldsymbol{D}}^T(\boldsymbol{\mu}_j)\bar{\boldsymbol{D}}(\boldsymbol{\mu}_j)$ low. Let $\boldsymbol{U}_1(\boldsymbol{\mu}_j), \dots, \boldsymbol{U}_{m'}(\boldsymbol{\mu}_j)$ be $m' \in \mathbb{N}$ input trajectories, and let $\boldsymbol{X}_1(\boldsymbol{\mu}_j), \dots, \boldsymbol{X}_{m'}(\boldsymbol{\mu}_j)$ be the corresponding trajectories and $\bar{\boldsymbol{X}}_1(\boldsymbol{\mu}_j), \dots, \bar{\boldsymbol{X}}_{m'}(\boldsymbol{\mu}_j)$ be the corresponding re-projected trajectories computed with Algorithm 1. We concatenate the trajectories to obtain

$$\boldsymbol{U}(\boldsymbol{\mu}_j) = \left[\boldsymbol{U}_1(\boldsymbol{\mu}_j), \dots, \boldsymbol{U}_{m'}(\boldsymbol{\mu}_j)\right], \quad \bar{\boldsymbol{X}} = \left[\bar{\boldsymbol{X}}_1(\boldsymbol{\mu}_j), \dots, \bar{\boldsymbol{X}}_{m'}(\boldsymbol{\mu}_j)\right] \tag{4.1}$$

and then use (4.1) and $\bar{\boldsymbol{Y}}(\boldsymbol{\mu}_j)$ obtained from $\bar{\boldsymbol{Y}}_1(\boldsymbol{\mu}_j), \dots, \bar{\boldsymbol{Y}}_{m'}(\boldsymbol{\mu}_j)$ in the least-squares problem (3.6) to learn a model.

5. Numerical results.

The numerical results in this section demonstrate that the proposed data sampling strategy with re-projection leads to low-dimensional models that match reduced models derived with traditional, intrusive model reduction methods up to numerical errors in practice. The toy example introduced in the problem formulation in section 2.4 is revisited in section 5.1. Section 5.2 derives models for the viscous Burgers' equation and section 5.3 for the Chafee–Infante equation. Both of these examples are one dimensional in the spatial domain. Section 5.4 demonstrates learning models from re-projected trajectories on a diffusion-reaction equation with a nonlinear reaction term with two spatial dimensions. All examples were computed with MATLAB version 2019a.



(a) trajectories

(b) condition number of $\bar{\boldsymbol{D}}^T\bar{\boldsymbol{D}}$

(c) difference of reduced model and learned model

Fig. 3. *Toy example: Plot* (a) *shows that time stepping the model fitted to re-projected trajectories gives a trajectory that matches the trajectory obtained with the reduced model from intrusive model reduction. Plot* (b) *shows that the condition number of* $\bar{\boldsymbol{D}}^T\bar{\boldsymbol{D}}$ *can be high, which means that numerical errors are amplified as shown in plot* (c). *Increasing the number of time steps* K *and concatenating multiple trajectories as described in section* 4.2, *and as used in sections* 5.2–5.4, *typically helps to keep the condition number reasonably low in practice.*

5.1. Toy example. We revisit the toy example introduced in section 2.4. Let $\bar{\boldsymbol{X}}$ be the re-projected trajectory obtained with Algorithm 1. Following the least-squares problem (3.6) described in Corollary 3.2, we learn a model from the re-projected trajectory $\bar{\boldsymbol{X}}$ and time step the learned model for the test initial condition $\boldsymbol{V}_n^T\boldsymbol{x}_0^{\text{test}}$ to obtain the trajectory $\hat{\boldsymbol{X}}^{\text{test}}$, which is plotted in Figure 3(a). The trajectory of the model learned from the re-projected trajectory closely follows the trajectory of the reduced model from intrusive model reduction, which is in stark contrast to the model learned from the trajectory $\breve{\boldsymbol{X}}$ without re-projection. Thus, the results in Figure 3(a) are in agreement with Corollary 3.2.

Now consider the data matrix $\bar{\boldsymbol{D}}$ defined in (3.5). Figure 3(b) shows the condition number of $\bar{\boldsymbol{D}}^T\bar{\boldsymbol{D}}$ for dimensions $n \in \{2, 4, 6\}$ and various numbers of time steps K. In this example, the condition number grows with the dimension n. This means that even though condition (3.4) together with a full-rank data matrix is sufficient to recover the same model as is obtained with intrusive model reduction, numerical errors are introduced into the learned operators because of the potentially high condition number of $\bar{\boldsymbol{D}}^T\bar{\boldsymbol{D}}$; cf. section 4.2. Figure 3(c) demonstrates that the difference

$$\frac{\|\hat{\boldsymbol{X}}^{\text{test}} - \tilde{\boldsymbol{X}}^{\text{test}}\|_F}{\|\tilde{\boldsymbol{X}}^{\text{test}}\|_F} \tag{5.1}$$

between the trajectory $\tilde{\boldsymbol{X}}^{\text{test}}$ of the reduced model obtained with intrusive model reduction and the trajectory $\hat{\boldsymbol{X}}^{\text{test}}$ of the model learned from the re-projected trajectory grows with the dimension n as numerical errors are amplified by the increasing condition number of $\bar{\boldsymbol{D}}^T\bar{\boldsymbol{D}}$ in this example. Increasing the number of time steps K seems to help to reduce the condition number, as shown in Figure 3(b). Note that we use a QR solver in the following to solve the least-squares problem (3.6) underlying operator inference, which means that the condition number of $\bar{\boldsymbol{D}}$ enters, rather than the condition number of $\bar{\boldsymbol{D}}^T\bar{\boldsymbol{D}}$; however, the trends shown in Figure 3(b) are still valid because the condition number of $\bar{\boldsymbol{D}}$ is the square root of the condition number of $\bar{\boldsymbol{D}}^T\bar{\boldsymbol{D}}$.

5.2. Burgers' equation. A similar setup as in [44] is used for demonstrating the proposed approach on the viscous Burgers' equation.

5.2.1. Setup. Set the spatial domain to $\Omega = (-1, 1) \subset \mathbb{R}$ and the parameter domain to $\mathcal{D} = [10^{-1}, 1]$. Let $T = 1$ be the end time. Consider the viscous Burgers' equation

$$\frac{\partial}{\partial t}x(\xi, t; \mu) + x(\xi, t; \mu)\frac{\partial}{\partial \xi}x(\xi, t; \mu) - \mu\frac{\partial^2}{\partial \xi^2}x(\xi, t; \mu) = 0\,, \qquad \xi \in \Omega\,,$$

with the spatial coordinate $\xi \in \Omega$, time $t \in [0, T]$, and parameter $\mu \in \mathcal{D}$. Impose Dirichlet boundary conditions $x(-1, t; \mu) = u(\mu)$ and $x(1, t; \mu) = -u(\mu)$ with the input $u(\mu) \in \mathbb{R}$. The initial condition is zero $x(\xi, 0; \mu) = 0$ for $\xi \in \Omega$. We discretize the Burgers' equation with finite differences on an equidistant grid in Ω so that a system of ordinary differential equations of order $N = 128$ is obtained. Time is discretized with the forward Euler method and time step size $\delta t = 10^{-4}$ to obtain

$$\boldsymbol{x}_{k+1}(\mu) = \boldsymbol{A}_1(\mu)\boldsymbol{x}_k(\mu) + \boldsymbol{A}_2\boldsymbol{x}_k^2(\mu) + \boldsymbol{B}u_k(\mu)\,, \qquad k = 0, \dots, K-1\,, \tag{5.2}$$

which is a polynomial nonlinear dynamical system (2.2) of degree $\ell = 2$ with $\boldsymbol{A}_1(\mu) \in \mathbb{R}^{N\times N}$, $\boldsymbol{A}_2 \in \mathbb{R}^{N\times N_2}$, and input matrix $\boldsymbol{B} \in \mathbb{R}^{N\times 1}$, where we now let the input $u_k(\mu)$ vary with the time step k. Note that $\boldsymbol{A}_2$ and $\boldsymbol{B}$ are independent of the parameter μ. Note further that $\boldsymbol{f}(\boldsymbol{x}_k(\mu), u_k(\mu); \mu) = \boldsymbol{A}_1(\mu)\boldsymbol{x}_k(\mu) + \boldsymbol{A}_2\boldsymbol{x}_k^2(\mu) + \boldsymbol{B}u_k(\mu)$ in this example.

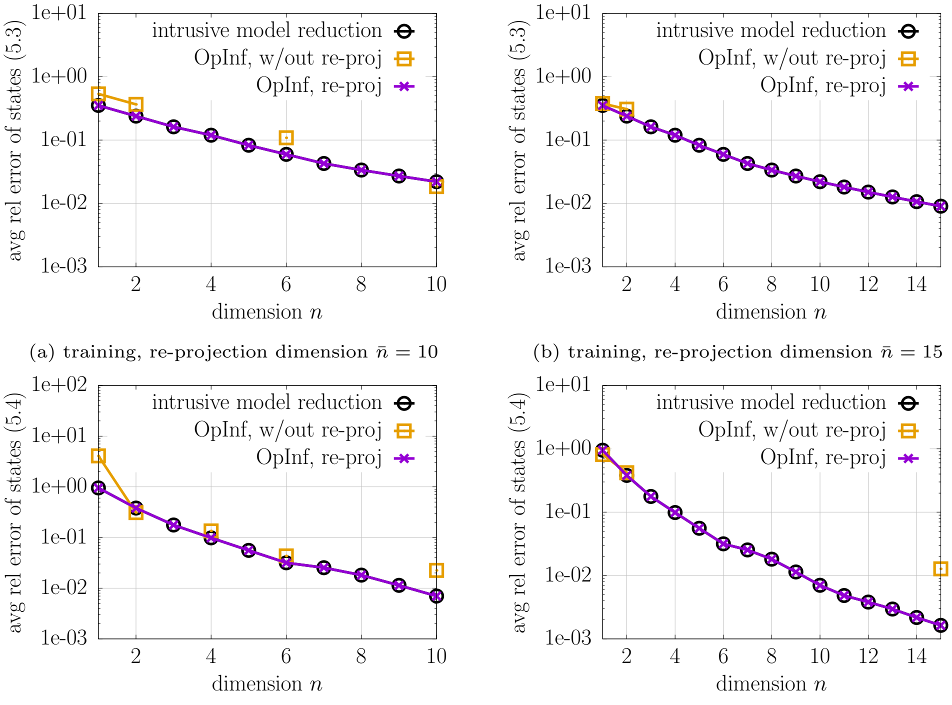
To generate trajectories from system (5.2) for learning low-dimensional models, consider the $m = 10$ equidistant parameters $\mu_1, \dots, \mu_m \in \mathcal{D}$ in the parameter domain $\mathcal{D}$. Set $m' = 5$, and consider the input trajectories $\boldsymbol{U}_l(\mu_j) = [u_{l,0}(\mu_j), \dots, u_{l,K-1}(\mu_j)]$ for $j = 1, \dots, m$ and $l = 1, \dots, m'$, where $u_{l,i}(\mu_j)$ is a realization of the random variable with uniform distribution in $[0, 10]$ for $i = 0, \dots, K-1$. In all of the following numerical results, we excite the high-dimensional system with a realization of a random input signal; however, the proposed approach is applicable with deterministic inputs as well. We generate m trajectories $\boldsymbol{X}_l(\mu_1), \dots, \boldsymbol{X}_l(\mu_m)$ for $l = 1, \dots, m'$ to derive the POD basis matrix $\boldsymbol{V}_{\bar{n}}$ of the POD space $\mathcal{V}_{\bar{n}}$ of dimension $\bar{n} \in \mathbb{N}$ from the snapshot matrix $[\boldsymbol{X}(\mu_1), \dots, \boldsymbol{X}(\mu_m)]$. The trajectories are $\boldsymbol{X}(\mu_i) = [\boldsymbol{X}_1(\mu_i), \dots, \boldsymbol{X}_{m'}(\mu_i)]$ for $i = 1, \dots, m$; cf. section 4.2. The re-projected trajectories $\bar{\boldsymbol{X}}(\mu_1), \dots, \bar{\boldsymbol{X}}(\mu_m)$, and the corresponding trajectories $\bar{\boldsymbol{Y}}(\mu_1), \dots, \bar{\boldsymbol{Y}}(\mu_m)$, are obtained by calling Algorithm 1 for each parameter $\mu_1, \dots, \mu_m$ and for $l = 1, \dots, m'$ and by concatenating the trajectories corresponding to the same parameters as described in section 4.2. We learn models $\hat{\boldsymbol{f}}(\cdot, \cdot, \mu_1), \dots, \hat{\boldsymbol{f}}(\cdot, \cdot, \mu_m)$ by solving the optimization problem (3.6) stated in Corollary 3.2 using the re-projected trajectories. We verified numerically that the data matrices have full rank. Condition (3.4) holds as well, and thus Corollary 3.2 is applicable in this setup, which means that we expect that time stepping the learned model gives a trajectory that matches the corresponding trajectory of the reduced model from intrusive model reduction up to numerical errors. Note that a numerical verification of the rank of the data matrix depends on tolerances. We used the implementation of the `rank` command shipped with MATLAB 2019a and its default tolerance; cf. section 4.2. We construct the reduced models $\tilde{\boldsymbol{f}}(\cdot,\cdot;\mu_1),\dots,\tilde{\boldsymbol{f}}(\cdot,\cdot;\mu_m)$ with intrusive model reduction and learn models $\breve{\boldsymbol{f}}(\cdot,\cdot;\mu_1),\dots,\breve{\boldsymbol{f}}(\cdot,\cdot;\mu_m)$ from the projected trajectories $\breve{\boldsymbol{X}}(\mu_1),\dots,\breve{\boldsymbol{X}}(\mu_m)$ (with*out* re-projection) as described in section 2.3. The projected trajectories $\breve{\boldsymbol{X}}(\mu_1),\dots,\breve{\boldsymbol{X}}(\mu_m)$ are obtained by concatenating the trajectories $\breve{\boldsymbol{X}}_1(\mu_1),\dots,\breve{\boldsymbol{X}}_{m'}(\mu_m)$ accordingly. For a parameter $\mu\in\mathcal{D}\setminus\{\mu_1,\dots,\mu_m\}$, model $\hat{\boldsymbol{f}}(\cdot,\cdot;\mu)$ is derived by componentwise spline interpolation of the operators of the learned models $\hat{\boldsymbol{f}}(\cdot,\cdot;\mu_1),\dots,\hat{\boldsymbol{f}}(\cdot,\cdot;\mu_m)$. The same interpolation approach is used for deriving the reduced model $\tilde{\boldsymbol{f}}(\cdot,\cdot;\mu)$ corresponding to intrusive model reduction and the model $\breve{\boldsymbol{f}}(\cdot,\cdot;\mu)$ learned from trajectories without re-projection for $\mu\in\mathcal{D}\setminus\{\mu_1,\dots,\mu_m\}$. To derive model $\hat{\boldsymbol{f}}(\cdot,\cdot;\mu)$ for dimension $n<\bar{n}$, we truncate the operators of $\hat{\boldsymbol{f}}(\cdot,\cdot;\mu)$ accordingly, which is the same approach as used in [44]. This means that for $n<\bar{n}$, the $n\times n$ submatrix of $\hat{\boldsymbol{A}}_1(\mu)\in\mathbb{R}^{\bar{n}\times\bar{n}}$ of model $\hat{\boldsymbol{f}}(\cdot,\cdot;\mu)$ is extracted, which corresponds to the first n POD modes. A similar process is performed for the input matrix, quadratic terms, and higher-degree nonlinear terms if present. Thus, model $\hat{\boldsymbol{f}}(\cdot,\cdot;\mu)$ is learned once for dimension $\bar{n}$ and then truncated for $n<\bar{n}$. The reduced model $\tilde{\boldsymbol{f}}(\cdot,\cdot;\mu)$ obtained with intrusive model reduction and model $\breve{\boldsymbol{f}}(\cdot,\cdot;\mu)$ are truncated the same way for $n<\bar{n}$.

5.2.2. Results. Figure 4(a) shows the error

$$\frac{1}{m}\sum_{i=1}^{m}\frac{\|\boldsymbol{V}_n\boldsymbol{Z}(\mu_i)-\boldsymbol{X}(\mu_i)\|_F}{\|\boldsymbol{X}(\mu_i)\|_F}\,, \tag{5.3}$$

where $\boldsymbol{Z}(\mu_i)=[\boldsymbol{Z}_1(\mu_i),\dots,\boldsymbol{Z}_{m'}(\mu_i)]$, for $i=1,\dots,m$, is the concatenated trajectory of either the reduced model $\tilde{\boldsymbol{f}}(\cdot,\cdot;\mu_i)$ obtained with intrusive model reduction, the model $\hat{\boldsymbol{f}}(\cdot,\cdot;\mu_i)$ learned from re-projected trajectories, or the model $\breve{\boldsymbol{f}}(\cdot,\cdot;\mu_i)$ learned from trajectories without re-projection for all m' inputs $\boldsymbol{U}_1(\mu_i),\dots,\boldsymbol{U}_{m'}(\mu_i)$. The dimension $\bar{n}$ of the POD space used for re-projection is set to $\bar{n}=10$, and operators are truncated as described in section 5.2.1 to compute error (5.3) corresponding to models with $n<\bar{n}$. The results in Figure 4(a) are reported for the training parameters $\mu_1,\dots,\mu_m$ and the training inputs that are also used in section 5.2.1 to construct the POD basis matrix and to learn the models. The reduced model obtained with intrusive model reduction achieves an error of almost 10^{-2} for $n=10$ dimensions. The model learned from trajectories without re-projection exhibits unstable behavior for most dimensions $n=1,\dots,10$ in the sense that the state during time stepping numerically diverges to NaNs (not a number). Missing values in Figure 4(a) mean that the states diverged to NaNs. In contrast, the model learned from trajectories with re-projection achieves an error (5.3) that closely follows the error of the reduced model from intrusive model reduction. To test the learned models on parameters that are different from the parameters used for learning the models, we select $m_{\text{test}}=7$ test parameters $\mu_1^{\text{test}},\dots,\mu_7^{\text{test}}$ equidistantly in $\mathcal{D}$ and set the input constant to 1. The corresponding error

$$\frac{1}{m_{\text{test}}}\sum_{i=1}^{m_{\text{test}}}\frac{\|\boldsymbol{V}_n\boldsymbol{Z}(\mu_i^{\text{test}})-\boldsymbol{X}(\mu_i^{\text{test}})\|_F}{\|\boldsymbol{X}(\mu_i^{\text{test}})\|_F} \tag{5.4}$$



(a) training, re-projection dimension $\bar{n} = 10$

(b) training, re-projection dimension $\bar{n} = 15$

(c) test, re-projection dimension $\bar{n} = 10$

(d) test, re-projection dimension $\bar{n} = 15$

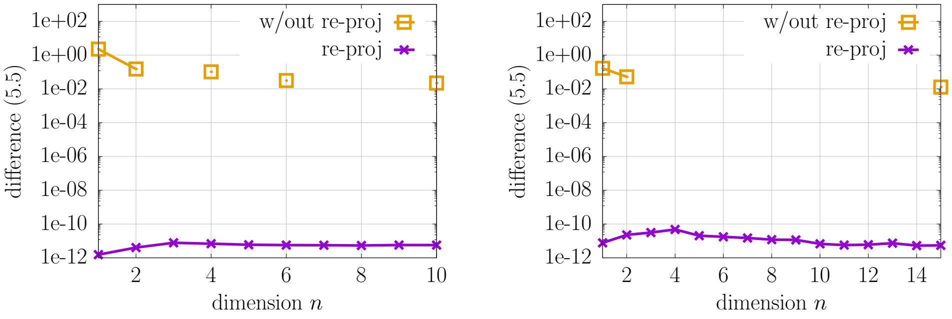
FIG. 4. *Burgers' equation: The results in plots* (a)–(d) *demonstrate Corollary* 3.2*, which states that models fitted with operator inference to re-projected trajectories are the reduced models obtained with traditional model reduction. In contrast, models fitted to projected trajectories (without re-projection) perform significantly worse and even show unstable behavior (missing values).*

is plotted in Figure 4(c). The models learned from re-projected trajectories achieve errors similar to those of the reduced models obtained with intrusive model reduction, in contrast to models learned from trajectories with*out* re-projection, which lead to significantly higher errors. Similar observations can be made for $\bar{n} = 15$, as shown in Figure 4(b) for training parameters and training inputs and in Figure 4(d) for test parameters and test inputs.

Now consider the difference

$$\frac{1}{m_{\text{test}}} \sum_{i=1}^{m_{\text{test}}} \frac{\|\boldsymbol{Z}(\mu_i^{\text{test}}) - \tilde{\boldsymbol{X}}(\mu_i^{\text{test}})\|_F}{\|\tilde{\boldsymbol{X}}(\mu_i^{\text{test}})\|_F} \tag{5.5}$$

between the trajectories of the reduced models from intrusive model reduction and the trajectories computed with the learned models. Thus, $\boldsymbol{Z}(\mu_i^{\text{test}})$ in (5.5) is either the trajectory obtained with $\hat{\boldsymbol{f}}(\cdot, \cdot; \mu_i^{\text{test}})$ or with $\breve{\boldsymbol{f}}(\cdot, \cdot; \mu_i^{\text{test}})$ for $i = 1, \dots, m_{\text{test}}$. The difference (5.5) is plotted in Figure 5. The models learned from re-projected trajectories achieve a difference to the reduced model from intrusive model reduction of less than 10^{-10}, whereas the models learned from trajectories without re-projection are up to eight orders of magnitude worse in terms of difference (5.5) and diverge in most cases (missing values in the plots).



(a) difference, re-projection dimension $\bar{n} = 10$

(b) difference, re-projection dimension $\bar{n} = 15$

FIG. 5. *Burgers' equation: The plots show that time stepping models learned from re-projected trajectories give the same trajectories, up to numerical errors, as reduced models obtained with intrusive model reduction. In contrast, models learned from trajectories without re-projection lead to behavior significantly different from that of the corresponding reduced models from intrusive model reduction. The results shown are for the test parameters $\mu'_1, \dots, \mu'_7$. Missing values mean that the states of the corresponding model diverged to NaNs during time stepping.*

5.3. Chafee–Infante equation.

A similar setup as in [8] is used in this section.

5.3.1. Setup. Set the spatial domain to $\Omega = (0, 1) \subset \mathbb{R}$ and the end time to $T = 4$. We consider the Chafee–Infante equation given by

$$\frac{\partial}{\partial t} x(\xi, t) - \frac{\partial^2}{\partial \xi^2} x(\xi, t) + x^3(\xi, t) - x(\xi, t) = 0 \,, \qquad \xi \in \Omega \,, \tag{5.6}$$

with the spatial coordinate $\xi \in \Omega$ and time $t \in [0, T]$. Note that we consider a parameter-independent version of the Chafee–Infante equation. The boundary conditions are

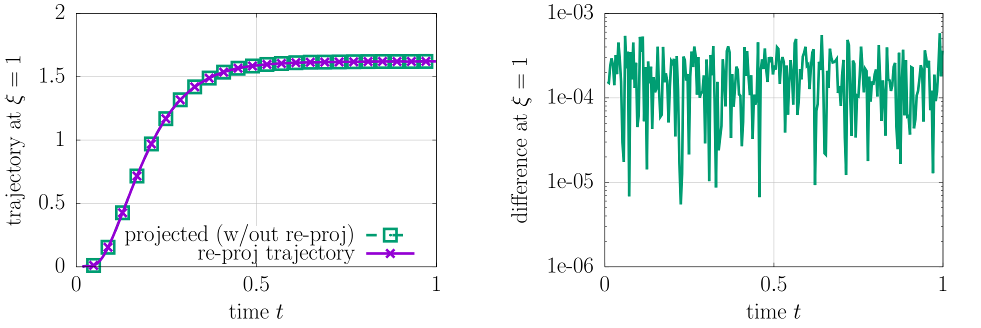
$$\frac{\partial}{\partial \xi} x(1, t) = 0 \,, \quad x(0, t) = u \,, \quad t \in [0, T] \,,$$

with the input $u \in \mathbb{R}$. The initial condition is $x(\xi, t) = 0$ for $\xi \in \Omega \cup \{0, 1\}$. The spatial domain Ω is discretized on an equidistant grid with mesh width 2^{-7} and finite differences. Time is discretized with the forward Euler method and time-step size $\delta t = 10^{-5}$ to obtain the time-discrete dynamical system with polynomial nonlinear terms up to degree $\ell = 3$:

$$\boldsymbol{x}_{k+1} = \boldsymbol{A}_1 \boldsymbol{x}_k + \boldsymbol{A}_2 \boldsymbol{x}_k^2 + \boldsymbol{A}_3 \boldsymbol{x}_k^3 + \boldsymbol{B} u_k \,, \qquad k = 0, \dots, K - 1, \tag{5.7}$$

for $K = 4 \times 10^5$ and $N = 128$ and where the input matrix $\boldsymbol{B}$ corresponds to the discretization of the boundary conditions and the input u_k now varies in time.

Consider the $m' = 25$ input trajectories $\boldsymbol{U}_1, \dots, \boldsymbol{U}_{m'}$ with components sampled randomly from a uniform distribution in $[0, 10]$, and let $\boldsymbol{X}_1, \dots, \boldsymbol{X}_{m'}$ be the corresponding trajectories of system (5.7). The same steps as in section 5.2.1 are performed to concatenate the trajectories $\boldsymbol{X}_1, \dots, \boldsymbol{X}_{m'}$, to derive a POD space of dimension $\bar{n} \in \mathbb{N}$ and the corresponding re-projected trajectories $\bar{\boldsymbol{X}}_1, \dots, \bar{\boldsymbol{X}}_{m'}$ and the concatenated re-projected trajectory $\bar{\boldsymbol{X}}$, and to learn the model $\hat{\boldsymbol{f}}(\cdot, \cdot)$ from the re-projected trajectory $\bar{\boldsymbol{X}}$. Additionally, as described in section 5.2.1, the reduced model $\tilde{\boldsymbol{f}}(\cdot, \cdot)$ from intrusive model reduction and the model $\breve{\boldsymbol{f}}(\cdot, \cdot)$ learned from the trajectories without re-projection are constructed. The test input is $u(t) = 25(\sin(\pi t) + 1)$, which is also used in [8]. The data matrices have numerically full rank; cf. section 5.2.1.



(a) trajectories

(b) difference re-projected and projected trajectory

FIG. 6. *Chafee–Infante: Even though the projected trajectory (without re-projection) and the re-projected trajectory are similar in this example, as shown in plots* (a) *and* (b)*, the corresponding closure error* (2.11) *has a significant polluting effect on operators learned from trajectories without re-projection, as shown in Figure* 7.

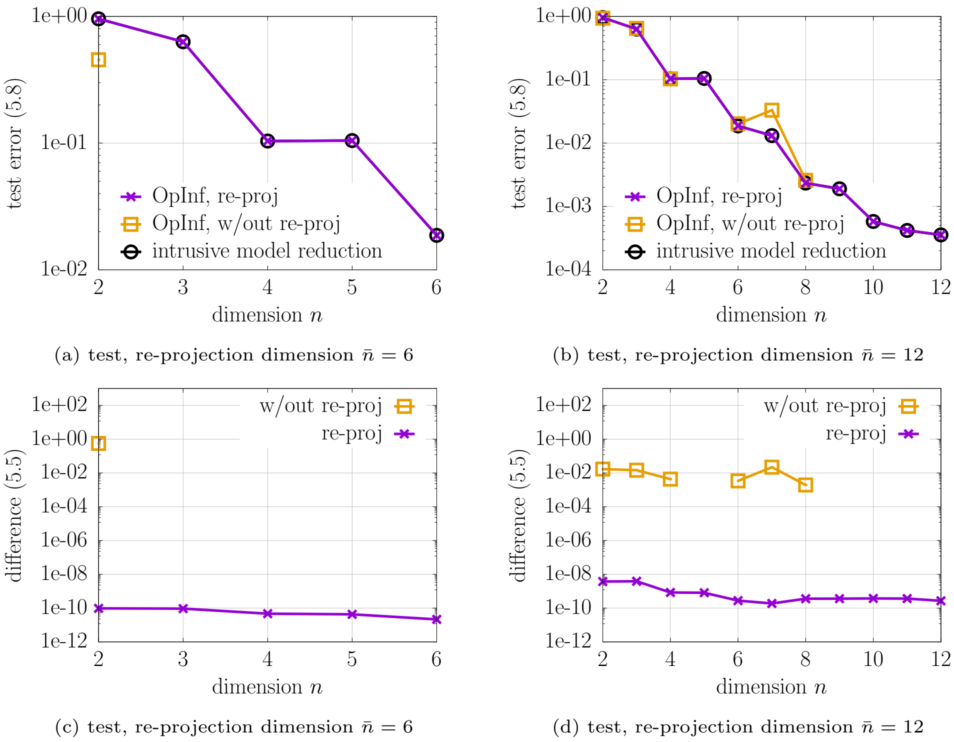
5.3.2. Results. Consider the re-projected trajectory $\bar{\boldsymbol{X}}_1$ and the projected trajectory $\breve{\boldsymbol{X}}_1 = \boldsymbol{V}_{\bar{n}}^T \boldsymbol{X}_1$ for $\bar{n} = 10$. Let $\boldsymbol{v}_{\cdot,N} \in \mathbb{R}^{1 \times \bar{n}}$ be the last row of $\boldsymbol{V}_{\bar{n}}$ so that $\boldsymbol{v}_{\cdot,N}\breve{\boldsymbol{X}}_1$ and $\boldsymbol{v}_{\cdot,N}\bar{\boldsymbol{X}}_1$ is the approximation of the state at spatial coordinate $\xi = 1$ given by the projected and the re-projected trajectories, respectively. Figure 6(a) plots $\boldsymbol{v}_{\cdot,N}\breve{\boldsymbol{X}}_1$ and $\boldsymbol{v}_{\cdot,N}\bar{\boldsymbol{X}}_1$ restricted to time $t \in [0, 1]$. Both trajectories overlap, which indicates that the projected and the re-projected trajectories are similar in this example. The absolute value of the difference $\boldsymbol{v}_{\cdot,N}\breve{\boldsymbol{X}}_1 - \boldsymbol{v}_{\cdot,N}\bar{\boldsymbol{X}}_1$ against the time step is shown in Figure 6(b) and indicates again that the projected and the re-projected trajectories are close relative to the absolute value of the trajectories in Figure 6(a); however, even this small difference has a polluting effect on operator inference that can lead to poor models. Consider Figure 7, which shows the test error

$$\frac{\|\boldsymbol{V}_n \boldsymbol{Z}_{\text{test}} - \boldsymbol{X}_{\text{test}}\|_F}{\|\boldsymbol{X}_{\text{test}}\|_F} \tag{5.8}$$

for $n \leq \bar{n}$ and where $\boldsymbol{Z}_{\text{test}}$ is computed with the test input with either model $\boldsymbol{f}(\cdot,\cdot)$, $\hat{\boldsymbol{f}}(\cdot,\cdot)$, $\tilde{\boldsymbol{f}}(\cdot,\cdot)$, or $\breve{\boldsymbol{f}}(\cdot,\cdot)$. Even though the difference between the projected and the re-projected trajectories is small in this example, the results in Figure 7 demonstrate that fitting a model to trajectories without re-projection leads to poor approximations of the reduced models obtained with intrusive model reduction.

5.3.3. Results with two-step Adams–Bashforth time stepping. We consider the same setup as in section 5.3.1, except that (5.6) is discretized with the two-step explicit Adams–Bashforth scheme. Algorithm 1 is called to generate the re-projected trajectory with the two-step explicit Adams–Bashforth scheme. The first time step in the two-step Adams–Bashforth scheme is performed with the explicit Euler method, which means that the system has the form (5.7) for the first time step and thus the re-projection scheme is applicable. Once the operators $\hat{\boldsymbol{A}}_1, \hat{\boldsymbol{A}}_2, \hat{\boldsymbol{A}}_3, \hat{\boldsymbol{B}}$ are inferred, a reduced model with two-step Adams–Bashforth time stepping is obtained with

$$\begin{aligned}\hat{\boldsymbol{x}}_{k+2} = \hat{\boldsymbol{x}}_{k+1} + \frac{3}{2}\left((\hat{\boldsymbol{A}}_1 - \boldsymbol{I})\hat{\boldsymbol{x}}_{k+1} + \hat{\boldsymbol{A}}_2\hat{\boldsymbol{x}}_{k+1}^2 + \hat{\boldsymbol{A}}_3\hat{\boldsymbol{x}}_{k+1}^3 + \hat{\boldsymbol{B}}u_{k+1}\right)\\ - \frac{1}{2}\left((\hat{\boldsymbol{A}}_1 - \boldsymbol{I})\hat{\boldsymbol{x}}_k + \hat{\boldsymbol{A}}_2\hat{\boldsymbol{x}}_k^2 + \hat{\boldsymbol{A}}_3\hat{\boldsymbol{x}}_k^3 + \hat{\boldsymbol{B}}u_k\right),\end{aligned} \tag{5.9}$$



(a) test, re-projection dimension $\bar{n} = 6$

(b) test, re-projection dimension $\bar{n} = 12$

(c) test, re-projection dimension $\bar{n} = 6$

(d) test, re-projection dimension $\bar{n} = 12$

FIG. 7. *Chafee–Infante equation: Models learned from re-projected trajectories achieve performance in terms of error* (5.8) *similar to that of reduced models from intrusive model reduction in this example. Even though the difference between re-projected trajectories and projected trajectories is small in this example (cf. Figure* 6*), models learned from trajectories without re-projection perform significantly worse than models learned from re-projected trajectories. Missing values correspond to models that numerically diverged during time stepping.*

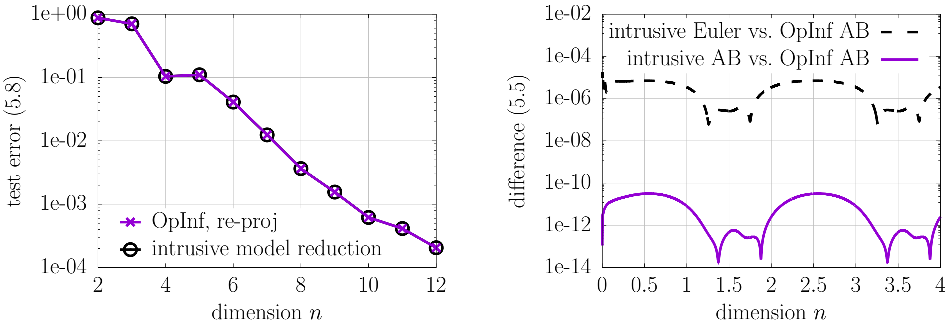
where $\boldsymbol{I} \in \mathbb{R}^{n \times n}$ is the $n \times n$ identity matrix. Note that the identity matrix is subtracted from $\hat{\boldsymbol{A}}_1$ because $\hat{\boldsymbol{A}}_1$ corresponds to a time discretization with the explicit Euler method. A similar process leads to reduced models with Adams–Bashforth schemes with more than two steps. Figure 8(a) shows that first inferring $\hat{\boldsymbol{A}}_1, \hat{\boldsymbol{A}}_2, \hat{\boldsymbol{A}}_3, \hat{\boldsymbol{B}}$ and then assembling the model (5.9) gives the same trajectory up to numerical errors as intrusive model reduction in terms of the error (5.8). Figure 8(b) shows the difference (5.5) of the trajectories obtained with a model from intrusive model reduction with Adams–Bashforth time stepping and the model obtained with operator inference and re-projection with Adams–Bashforth time stepping. The difference between the two trajectories is small and comparable to what is obtained with explicit Euler time stepping; cf. Figure 7. Figure 8(b) also shows that there is a more significant difference between the trajectory obtained with the model from operator inference and Adams–Bashforth and the trajectory obtained from intrusive model reduction and explicit Euler time stepping.

5.4. Diffusion-reaction equation.

The setup of the diffusion-reaction equation in this section follows the example in [42].

5.4.1. Setup.

Let $\Omega = (0, 1)^2 \subset \mathbb{R}^2$ be the spatial domain with boundary $\partial\Omega$ and closed set $\bar{\Omega} = \Omega \cup \partial\Omega$. Further, let $\mu \in \mathcal{D} = [1, 1.5]$ be the parameter domain.



(a) Adams–Bashforth time stepping (b) time-stepping with Euler vs. Adams–Bashforth (AB)

FIG. 8. *Chafee–Infante equation: Plot* (a) *shows that reduced models can be recovered even if the high-dimensional system is discretized with a two-step Adams–Bashforth scheme. It is important to note that it is required that the first time step of the high-dimensional system is performed with an explicit Euler step. Plot* (b) *compares the learned model with a two-step Adams–Bashforth scheme to models obtained with intrusive model reduction with either explicit Euler or two-step Adams–Bashforth schemes.*

Consider the PDE

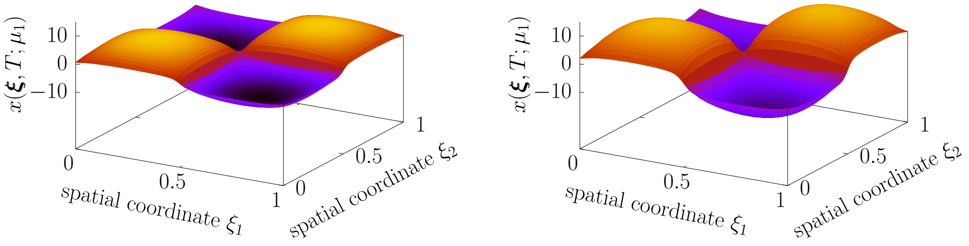
$$\frac{\partial}{\partial t} x(\boldsymbol{\xi}, t; \mu) = -\Delta x(\boldsymbol{\xi}, t; \mu) + s(\boldsymbol{\xi}) u(t) + g(x(\boldsymbol{\xi}, t; \mu)) \,, \qquad \boldsymbol{\xi} \in \Omega \,, \tag{5.10}$$

where the spatial coordinate is $\boldsymbol{\xi} = [\xi_1, \xi_2]^T$, the source term $s : \bar{\Omega} \to \mathbb{R}$ is $s(\boldsymbol{\xi}) = 10^{-1} \sin(2\pi\xi_1) \sin(2\pi\xi_2)$, and the nonlinear term $g : \bar{\Omega} \to \mathbb{R}$ is the second-order Taylor approximation of $x \mapsto -(a \sin(\mu) + 2) \exp(-(\mu^2) b) \exp(\mu x c)$ about 0 and with $a = 0.1$, $b = 2.7$, and $c = 1.8$, which is the same nonlinear term as used in [42]. The initial condition is 0. We impose homogeneous Neumann boundary conditions and discretize with finite differences on a grid with 64 equidistant grid points in each dimension. Time is discretized with the forward Euler method and time step size $\delta t = 10^{-2}$ and end time $T = 100$. The corresponding time-discrete dynamical system is

$$\boldsymbol{x}_{k+1}(\mu) = \boldsymbol{A}_1(\mu) \boldsymbol{x}_k(\mu) + \boldsymbol{A}_2(\mu) \boldsymbol{x}_k^2(\mu) + \boldsymbol{A}_3(\mu) \boldsymbol{x}_k^3(\mu) + \boldsymbol{B} u_k \,, \qquad k = 0, \dots, K-1 \,,$$

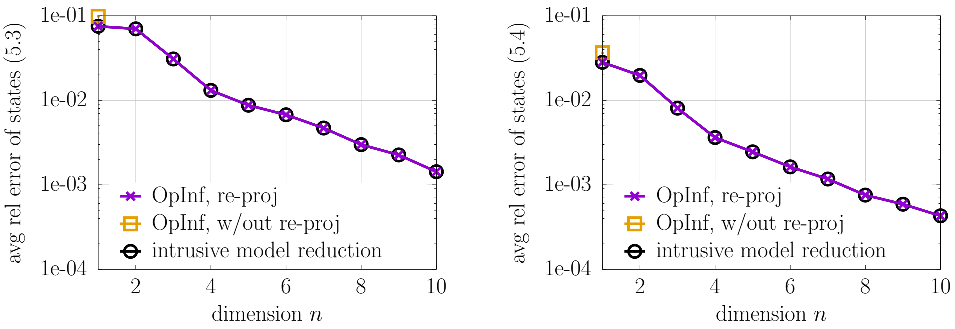
for $K = 10^4$. The dimension N of the state $\boldsymbol{x}_k$ at time step k is $N = 64^2 = 4096$. Plots of $\boldsymbol{x}_K(\mu)$ for $\mu = 1.0625$ and $\mu = 1.4375$ are given in Figure 9.

In order to construct a reduced space, we select $m = 10$ equidistant parameters $\mu_1, \dots, \mu_m \in \mathcal{D}$ and set the inputs to be realizations of the random variables uniformly distributed in $[1, 1000]$. From these trajectories, the basis matrix $\boldsymbol{V}_{\bar{n}}$ with $\bar{n} = 10$ columns is computed with POD. Then re-projected trajectories are sampled up to time $t = 5$ (instead of end time $T = 100$). For each μ_i, 10 re-projected trajectories with different random inputs are derived and concatenated together as described in section 4.2. The concatenation of trajectories ensures that the data matrix $\bar{\boldsymbol{D}}$ has full rank in this example, which has been numerically verified with the same procedure as in section 5.2.1. Models are learned with operator inference from the re-projected trajectories to obtain $\hat{\boldsymbol{f}}(\cdot, \cdot; \mu_1), \dots, \hat{\boldsymbol{f}}(\cdot, \cdot; \mu_m)$. The same process is repeated for the trajectories without re-projection to obtain the models $\breve{\boldsymbol{f}}(\cdot, \cdot; \mu_1), \dots, \breve{\boldsymbol{f}}(\cdot, \cdot; \mu_m)$. The rest of the setup is the same as in section 5.2. Test parameters are selected by taking the nine equidistant parameters in $\mathcal{D}$ and ignoring the two parameters at


$x(\boldsymbol{\xi}, T; \mu_1)$
10
0
−10
0
0.5
1
spatial coordinate ξ_1
0
0.5
1
spatial coordinate ξ_2

(a) high-dimensional system, $\mu = 1.0625$ (b) high-dimensional system, $\mu = 1.4375$

FIG. 9. *Diffusion-reaction: Plots show the numerical approximation of the solution of* (5.10) *for parameters* $\mu = 1.0625$ *and* $\mu = 1.4375$*, respectively.*



(a) training (b) test

FIG. 10. *Diffusion-reaction: Models learned from trajectories without re-projection show unstable behavior (missing values) for all dimensions* $n \geq 2$*. In contrast, models learned with operator inference from re-projected trajectories achieve the same errors as the reduced models from intrusive model reduction, which is guaranteed by Corollary* 3.2 *in this example.*

the boundaries, which are training parameters. Thus, there is a total of seven test parameters. Components of the test inputs are random with uniform distribution in $[1, 1000]$.

5.4.2. Results. Figure 10(a) shows the error (5.3) for the training parameters and training inputs. The model fitted to trajectories without re-projection numerically diverges to NaNs during time stepping for all dimensions $n > 2$. The model fitted to re-projected trajectories matches the behavior of the reduced model from intrusive model reduction as expected from the analysis presented in Corollary 3.2. The same observations can be made for the error (5.4) with the test parameters and test inputs.

6. Conclusions.

The presented approach exactly recovers reduced models from data under certain conditions. This result holds preasymptotically in the number of data points and the dimension of the reduced space as long as the corresponding data matrix is full rank. The optimization problem underlying operator inference with re-projected trajectories is convex and can be solved with standard numerical linear algebra packages. Numerical experiments demonstrate that reduced models are learned up to numerical errors in practice for a wide class of systems with polynomial nonlinear terms. Extension to more general systems remains future work: (1) An extension to systems with polynomial nonlinear terms in the inputs requires reformu-

Algorithm 3. MATLAB code for generating matrix $\mathcal{A}$ used in the example in section 2.4.

```
N = 10;
rng(1); % for reproducibility
D = diag(-logspace(-1, -2, N));
[W, ~] = qr(randn(N, N), 0);
A = W'*D*W; % matrix of time-continuous system
```

lating the least-squares problem underlying operator inference; however, we expect the re-projection scheme to apply to polynomial nonlinear terms in the inputs as well. (2) Systems with nonpolynomial nonlinear terms in the state and the inputs can be first lifted into systems with polynomial terms only, to which then operator inference and re-projection can be applied. Such liftings and transformations are investigated in, e.g., [24, 8, 47]. (3) An especially important class is given by systems with bilinear terms that mix state and inputs, which are obtained, e.g., via Carleman linearization of nonlinear systems; see [2, 22, 8]. It seems that operator inference with re-projection can be applied to bilinear systems because bilinear systems can be written as polynomials in the states and inputs and the corresponding operator-inference least-squares problems are linear in the reduced operators. However, more detailed investigations are necessary to determine what conditions are required so that the recovery of the reduced models obtained with intrusive model reduction methods can be guaranteed.

Appendix. System matrix of toy example. Algorithm 3 shows MATLAB code for generating matrix $\mathcal{A}$ of the example introduced in section 2.4. Code to reproduce the results presented in sections 2.4 and 5.1 is available in the git repository https://github.com/pehersto/reproj.

Acknowledgments. The author would like to thank Elizabeth Qian, Nihar Sawant, and Karen Willcox for many helpful discussions. The author would also like to thank Silke Glas and Shane McQuarrie for comments on an earlier version of this manuscript. The numerical experiments were computed with support by the NYU IT High Performance Computing resources, services, and staff expertise.

REFERENCES

[1] D. Amsallem, J. Cortial, K. Carlberg, and C. Farhat, *A method for interpolating on manifolds structural dynamics reduced-order models*, Internat. J. Numer. Methods Engrg., 80 (2009), pp. 1241–1258.

[2] A. C. Antoulas, *Approximation of Large-Scale Dynamical Systems*, SIAM, Philadelphia, 2005, https://doi.org/10.1137/1.9780898718713.

[3] A. Antoulas and B. D. Q. Anderson, *On the scalar rational interpolation problem*, IMA J. Math. Control Inform., 3 (1986), pp. 61–88.

[4] A. Antoulas, C. Beattie, and S. Gugercin, *Interpolatory model reduction of large-scale dynamical systems*, in Efficient Modeling and Control of Large-Scale Systems, J. Mohammadpour and K. Grigoriadis, eds., Springer-Verlag, New York, 2010.

[5] A. C. Antoulas, I. V. Gosea, and A. C. Ionita, *Model reduction of bilinear systems in the Loewner framework*, SIAM J. Sci. Comput., 38 (2016), pp. B889–B916, https://doi.org/10.1137/15M1041432.

[6] C. Audouze, F. De Vuyst, and P. B. Nair, *Reduced-order modeling of parameterized PDEs using time–space-parameter principal component analysis*, Internat. J. Numer. Methods Engrg., 80 (2009), pp. 1025–1057.

[7] C. Beattie and S. Gugercin, *Realization-independent $\mathcal{H}_2$-approximation*, in Proceedings of the IEEE Conference on Decision and Control, Maui, HI, 2012, pp. 4953–4958.

[8] P. Benner, P. Goyal, and S. Gugercin, *$\mathcal{H}_2$-quasi-optimal model order reduction for quadratic-bilinear control systems*, SIAM J. Matrix Anal. Appl., 39 (2018), pp. 983–1032, https://doi.org/10.1137/16M1098280.

[9] P. Benner, S. Gugercin, and K. Willcox, *A survey of projection-based model reduction methods for parametric dynamical systems*, SIAM Rev., 57 (2015), pp. 483–531, https://doi.org/10.1137/130932715.

[10] S. L. Brunton, B. W. Brunton, J. L. Proctor, and J. N. Kutz, *Koopman invariant subspaces and finite linear representations of nonlinear dynamical systems for control*, PLoS ONE, 11 (2016), pp. 1–19.

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

[12] K. Carlberg, M. Barone, and H. Antil, *Galerkin v. least-squares Petrov–Galerkin projection in nonlinear model reduction*, J. Comput. Phys., 330 (2017), pp. 693–734.

[13] K. Carlberg, C. Bou-Mosleh, and C. Farhat, *Efficient non-linear model reduction via a least-squares Petrov–Galerkin projection and compressive tensor approximations*, Internat. J. Numer. Methods Engrg., 86 (2011), pp. 155–181.

[14] K. P. Champion, S. L. Brunton, and J. N. Kutz, *Discovery of nonlinear multiscale systems: Sampling strategies and embeddings*, SIAM J. Appl. Dyn. Syst., 18 (2019), pp. 312–333, https://doi.org/10.1137/18M1188227.

[15] A. Chorin, O. H. Hald, and R. Kupferman, *Optimal prediction with memory*, Phys. D, 166 (2002), pp. 239–257.

[16] A. Chorin and P. Stinis, *Problem reduction, renormalization, and memory*, Commun. Appl. Math. Comput. Sci., 1 (2006), pp. 1–27.

[17] J. Degroote, J. Vierendeels, and K. Willcox, *Interpolation among reduced-order matrices to obtain parameterized models for design, optimization and probabilistic analysis*, Internat. J. Numer. Methods Fluids, 63 (2010), pp. 207–230.

[18] L. Fick, Y. Maday, A. T. Patera, and T. Taddei, *A stabilized POD model for turbulent flows over a range of Reynolds numbers: Optimal parameter sampling and constrained projection*, J. Comput. Phys., 371 (2018), pp. 214–243.

[19] D. Givon, R. Kupferman, and A. Stuart, *Extracting macroscopic dynamics: Model problems and algorithms*, Nonlinearity, 17 (2004), pp. R55–R127.

[20] I. V. Gosea and A. Antoulas, *Data-driven model order reduction of quadratic-bilinear systems*, Numer. Linear Algebra Appl., 25 (2018), e2200.

[21] A. Gouasmi, E. J. Parish, and K. Duraisamy, *A priori estimation of memory effects in reduced-order models of nonlinear systems using the Mori-Zwanzig formalism*, Proc. A., 473 (2017), 20170385.

[22] P. Goyal, M. I. Ahmad, and P. Benner, *Model reduction of quadratic-bilinear descriptor systems via Carleman bilinearization*, in Proceedings of the European Control Conference (ECC), 2015, pp. 1177–1182.

[23] M. A. Grepl and A. T. Patera, *A posteriori error bounds for reduced-basis approximations of parametrized parabolic partial differential equations*, M2AN Math. Model. Numer. Anal., 39 (2005), pp. 157–181.

[24] C. Gu, *QLMOR: A projection-based nonlinear model order reduction approach using quadratic-linear representation of nonlinear systems*, IEEE Trans. Comput.-Aided Design Integr. Circuits Syst., 30 (2011), pp. 1307–1320.

[25] S. Gugercin, A. C. Antoulas, and C. Beattie, *$\mathcal{H}_2$ model reduction for large-scale linear dynamical systems*, SIAM J. Matrix Anal. Appl., 30 (2008), pp. 609–638, https://doi.org/10.1137/060666123.

[26] M. Guo and J. S. Hesthaven, *Data-driven reduced order modeling for time-dependent problems*, Comput. Methods Appl. Mech. Engrg., 345 (2019), pp. 75–99.

[27] B. Haasdonk and M. Ohlberger, *Efficient reduced models and a posteriori error estimation for parametrized dynamical systems by offline/online decomposition*, Math. Comput. Model. Dyn. Syst., 17 (2011), pp. 145–161.

[28] E. Hairer, G. Wanner, and S. P. Nørsett, *Solving Ordinary Differential Equations* I, Springer, Berlin, 1993.

[29] J. S. Hesthaven and S. Ubbiali, *Non-intrusive reduced order modeling of nonlinear problems using neural networks*, J. Comput. Phys., 363 (2018), pp. 55–78.

[30] A. C. Ionita and A. C. Antoulas, *Data-driven parametrized model reduction in the Loewner framework*, SIAM J. Sci. Comput., 36 (2014), pp. A984–A1007, https://doi.org/10.1137/

130914619.

[31] B. Kramer and K. Willcox, *Nonlinear model order reduction via lifting transformations and proper orthogonal decomposition*, AIAA J., 57 (2019), pp. 2297–2307.

[32] J. N. Kutz, S. L. Brunton, B. W. Brunton, and J. L. Proctor, *Dynamic Mode Decomposition: Data-Driven Modeling of Complex Systems*, SIAM, Philadelphia, 2016, https://doi.org/10.1137/1.9781611974508.

[33] S. Le Clainche and J. M. Vega, *Higher order dynamic mode decomposition*, SIAM J. Appl. Dyn. Syst., 16 (2017), pp. 882–925, https://doi.org/10.1137/15M1054924.

[34] S. Lefteriu and A. Antoulas, *A new approach to modeling multiport systems from frequency-domain data*, IEEE Trans. Comput.-Aided Design Integr. Circuits Syst., 29 (2010), pp. 14–27.

[35] L. Ljung, *System Identification*, Prentice-Hall, Englewood Cliffs, NJ, 1987.

[36] H. Ly and H. Tran, *Modeling and control of physical processes using proper orthogonal decomposition*, Math. Comput. Model., 33 (2001), pp. 223–236.

[37] A. Mayo and A. Antoulas, *A framework for the solution of the generalized realization problem*, Linear Algebra Appl., 425 (2007), pp. 634–662.

[38] I. Mezić, *Spectral properties of dynamical systems, model reduction and decompositions*, Nonlinear Dynam., 41 (2005), pp. 309–325.

[39] S. Pan and K. Duraisamy, *Data-driven discovery of closure models*, SIAM J. Appl. Dyn. Syst., 17 (2018), pp. 2381–2413, https://doi.org/10.1137/18M1177263.

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

[41] B. Peherstorfer, S. Gugercin, and K. Willcox, *Data-driven reduced model construction with time-domain Loewner models*, SIAM J. Sci. Comput., 39 (2017), pp. A2152–A2178, https://doi.org/10.1137/16M1094750.

[42] B. Peherstorfer and Y. Marzouk, *A transport-based multifidelity preconditioner for Markov chain Monte Carlo*, Adv. Comput. Math., 45 (2019), pp. 2321–2348.

[43] B. Peherstorfer and K. Willcox, *Dynamic data-driven reduced-order models*, Comput. Methods Appl. Mech. Engrg., 291 (2015), pp. 21–41.

[44] B. Peherstorfer and K. Willcox, *Data-driven operator inference for nonintrusive projection-based model reduction*, Comput. Methods Appl. Mech. Engrg., 306 (2016), pp. 196–215.

[45] B. Peherstorfer, K. Willcox, and M. Gunzburger, *Survey of multifidelity methods in uncertainty propagation, inference, and optimization*, SIAM Rev., 60 (2018), pp. 550–591, https://doi.org/10.1137/16M1082469.

[46] C. Prud'homme, Y. Maday, A. T. Patera, G. Turinici, D. V. Rovas, K. Veroy, and L. Machiels, *Reliable real-time solution of parametrized partial differential equations: Reduced-basis output bound methods*, J. Fluids Eng., 124 (2001), pp. 70–80.

[47] E. Qian, B. Kramer, A. N. Marques, and K. Willcox, *Transform & learn: A data-driven approach to nonlinear model reduction*, in AIAA Aviation Forum, 2019.

[48] C. Rowley, I. Mezić, S. Bagheri, P. Schlatter, and D. Henningson, *Spectral analysis of nonlinear flows*, J. Fluid Mech., 641 (2009), pp. 115–127.

[49] G. Rozza, D. Huynh, and A. T. Patera, *Reduced basis approximation and a posteriori error estimation for affinely parametrized elliptic coercive partial differential equations*, Arch. Comput. Methods Eng., 15 (2008), pp. 229–275.

[50] S. H. Rudy, S. L. Brunton, J. L. Proctor, and J. N. Kutz, *Data-driven discovery of partial differential equations*, Science Advances, 3 (2017), e1602614.

[51] H. Schaeffer, R. Caflisch, C. D. Hauck, and S. Osher, *Sparse dynamics for partial differential equations*, Proc. Natl. Acad. Sci. USA, 110 (2013), pp. 6634–6639.

[52] H. Schaeffer, G. Tran, and R. Ward, *Extracting sparse high-dimensional dynamics from limited data*, SIAM J. Appl. Math., 78 (2018), pp. 3279–3295, https://doi.org/10.1137/18M116798X.

[53] P. Schmid, *Dynamic mode decomposition of numerical and experimental data*, J. Fluid Mech., 656 (2010), pp. 5–28.

[54] P. Schmid and J. Sesterhenn, *Dynamic mode decomposition of numerical and experimental data*, in 61st Annual Meeting of the APS Division of Fluid Dynamics, 2008.

[55] A. Schmidt and B. Haasdonk, *Reduced basis approximation of large scale parametric algebraic Riccati equations*, ESAIM Control Optim. Calc. Var., 24 (2018), pp. 129–151.

[56] L. Sirovich, *Turbulence and the dynamics of coherent structures*, Quart. Appl. Math., 45 (1987), pp. 561–571.

[57] P. Stinis, *Renormalized Mori-Zwanzig-reduced models for systems without scale separation*, Proc. A., 471 (2015), 20140446.

[58] R. Swischuk, B. Kramer, C. Huang, and K. Willcox, *Learning physics-based reduced-order models for a single-injector combustion process*, AIAA J., to appear.

[59] R. Swischuk, L. Mainini, B. Peherstorfer, and K. Willcox, *Projection-based model reduction: Formulations for physics-based machine learning*, Comput. & Fluids, 179 (2019), pp. 704–717.

[60] E. H. Thiede, D. Giannakis, A. R. Dinner, and J. Weare, *Galerkin approximation of dynamical quantities using trajectory data*, J. Chem. Phys., 150 (2019), 244111.

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

[62] K. Veroy and A. T. Patera, *Certified real-time solution of the parametrized steady incompressible Navier-Stokes equations: Rigorous reduced-basis a posteriori error bounds*, Internat. J. Numer. Methods Fluids, 47 (2005), pp. 773–788.

[63] K. Veroy, C. Prud'homme, D. Rovas, and A. T. Patera, *A posteriori error bounds for reduced-basis approximation of parametrized noncoercive and nonlinear elliptic partial differential equations*, in 16th AIAA Computational Fluid Dynamics Conference, American Institute of Aeronautics and Astronautics, Reston, VA, 2003.

[64] Z. Wang, I. Akhtar, J. Borggaard, and T. Iliescu, *Proper orthogonal decomposition closure models for turbulent flows: A numerical comparison*, Comput. Methods Appl. Mech. Engrg., 237/240 (2012), pp. 10–26.

[65] M. O. Williams, I. G. Kevrekidis, and C. W. Rowley, *A data-driven approximation of the Koopman operator: Extending dynamic mode decomposition*, J. Nonlinear Sci., 25 (2015), pp. 1307–1346.

[66] D. Wirtz, N. Karajan, and B. Haasdonk, *Surrogate modeling of multiscale models using kernel methods*, Internat. J. Numer. Methods Engrg., 101 (2014), pp. 1–28.

[67] Z. Zhang, M. Guo, and J. S. Hesthaven, *Model order reduction for large-scale structures with local nonlinearities*, Comput. Methods Appl. Mech. Engrg., 353 (2019), pp. 491–515.

[68] Y. Zhu, J. M. Dominy, and D. Venturi, *On the estimation of the Mori-Zwanzig memory integral*, J. Math. Phys., 59 (2018), 103501.