Data-driven model reduction via non-intrusive optimization of projection operators and reduced-order dynamics

Alberto Padovan[*], Blaine Vollmer[*], and Daniel J. Bodony[*]

Abstract. Computing reduced-order models using non-intrusive methods is particularly attractive for systems that are simulated using black-box solvers. However, obtaining accurate data-driven models can be challenging, especially if the underlying systems exhibit large-amplitude transient growth. Although these systems may evolve near a low-dimensional subspace that can be easily identified using standard techniques such as Proper Orthogonal Decomposition (POD), computing accurate models often requires projecting the state onto this subspace via a non-orthogonal projection. While appropriate oblique projection operators can be computed using intrusive techniques that leverage the form of the underlying governing equations, purely data-driven methods currently tend to achieve dimensionality reduction via orthogonal projections, and this can lead to models with poor predictive accuracy. In this paper, we address this issue by introducing a non-intrusive framework designed to simultaneously identify oblique projection operators and reduced-order dynamics. In particular, given training trajectories and assuming reduced-order dynamics of polynomial form, we fit a reduced-order model by solving an optimization problem over the product manifold of a Grassmann manifold, a Stiefel manifold, and several linear spaces (as many as the tensors that define the low-order dynamics). Furthermore, we show that the gradient of the cost function with respect to the optimization parameters can be conveniently written in closed form, so that there is no need for automatic differentiation. We compare our formulation with state-of-the-art methods on three examples: a three-dimensional system of ordinary differential equations, the complex Ginzburg-Landau (CGL) equation, and a two-dimensional lid-driven cavity flow at Reynolds number $Re = 8300$.

Key words. Model reduction, Data-driven reduced-order models, Manifold optimization, Operator inference.

AMS subject classifications. 37M05, 37M10, 37N10

1. Introduction.

Computing reduced-order models (ROMs) of high-dimensional systems is often necessary to perform several tasks, including accelerating expensive simulations, developing control strategies and solving design optimization problems. Most model reduction frameworks share the following key ingredients: a possibly nonlinear map from the high-dimensional state space to a low-dimensional space (i.e., an encoder), a possibly nonlinear map from the low-dimensional space to the original high-dimensional space (i.e., a decoder), and reduced-order dynamics to evolve the reduced-order state. Here, we provide a brief review of intrusive and non-intrusive methods where the reduced-order dynamics are continuous in time, and where the encoder and decoder define linear projection operators (i.e., the encoder and decoder are linear maps and the encoder is a left-inverse of the decoder).

Perhaps the most well-known reduced-order models that fall within this category are the so-called linear-projection Petrov-Galerkin models. These are obtained by (obliquely) projecting the full-order dynamics onto a low-dimensional linear subspace. In particular, given a decoder $\boldsymbol{\Phi}\left(\boldsymbol{\Psi}^\intercal\boldsymbol{\Phi}\right)^{-1}$ and an encoder $\boldsymbol{\Psi}^\intercal$, where $\boldsymbol{\Phi}$ and $\boldsymbol{\Psi}$ are tall rectangular matrices that define a projection $\mathbb{P} = \boldsymbol{\Phi}\left(\boldsymbol{\Psi}^\intercal\boldsymbol{\Phi}\right)^{-1}\boldsymbol{\Psi}^\intercal$, the aforementioned linear subspace is given by

[*]Department of Aerospace Engineering, University of Illinois at Urbana-Champaign, Urbana, IL, 61801 (padovan3@illinois.edu)

the span of $\mathbf{\Phi}$, and $\mathbf{\Psi}$ specifies the direction of projection. This is illustrated in figure 1 in [32]. If $\mathbf{\Phi} = \mathbf{\Psi}$, then the projection $\mathbb{P}$ is orthogonal and the model is known as a Galerkin model. In the simplest of cases, a Galerkin model can be obtained by orthogonally projecting the dynamics onto the span of the leading Proper Orthogonal Decomposition (POD) modes of a representative training data set. This procedure is “weakly” intrusive in the sense that it requires access to the governing equations, but not necessarily to the linearization and adjoint of the underlying nonlinear dynamics. In the context of fluids, POD-Galerkin models have been used extensively for both compressible and incompressible flows [31, 23, 2, 32]. However, these models may not perform well in systems that exhibit large-amplitude transient growth. Examples of such systems in fluid mechanics include boundary layers, mixing layers, jets and high-shear flows in general [8]. The difficulty posed by these systems can often be traced back to the non-normal[1] nature of the underlying linear dynamics, which demands the use of carefully chosen oblique projections. In linear systems, or nonlinear systems that evolve near a steady state, this problem can be addressed using methods such as Balanced Truncation [22, 11, 39] or Balanced POD [30], which produce oblique projection operators and corresponding Petrov-Galerkin models by balancing the observability and reachability Gramians associated with the underlying linear dynamics. Extensions and variants of Balanced Truncation and Balanced POD also exist for quadratic-bilinear systems [4] and for systems that evolve in the proximity of time-periodic orbits [38, 20, 27]. Beyond balancing, we find several other approaches from linear systems theory, including $\mathcal{H}_2$ and $\mathcal{H}_\infty$ model reduction, where reduced-order models are obtained by minimizing the $\mathcal{H}_2$ and $\mathcal{H}_\infty$ norms of the error between the full-order and reduced-order transfer functions [37, 12]. As in the case of balancing, extensions of $\mathcal{H}_2$-optimal (and quasi-optimal) model reduction were developed for quadratic-bilinar systems [3, 5]. For highly nonlinear systems that lie outside the region of applicability of linear model reduction methods, one can turn to recently-developed methods such as Trajectory-based Optimization of Oblique Projections (TrOOP) [25] and Covariance Balancing Reduction using Adjoint Snapshots (COBRAS) [26]. TrOOP identifies optimal oblique projections for Petrov-Galerkin modelling by training against trajectories generated by the full-order model, while COBRAS identifies oblique projections for model reduction by balancing the state and gradient covariances associated with the full-order solution map. We shall see that our non-intrusive formulation is closely related to TrOOP, so we will discuss the latter in more detail in section 2.5. All these Petrov-Galerkin methods are intrusive: not only do they require access to the full-order dynamics, but also to their linearization about steady or time-varying base flows and to the adjoint of the linearized dynamics. Thus, they are not easily applicable to systems that are simulated using black-box solvers.

Among existing techniques to obtain data-driven reduced-order models with continuous-time dynamics on linear subspaces, the most well-known is perhaps Operator Inference [28, 19]. Operator Inference fits a model to data by minimizing the difference between (usually polynomial) reduced-order dynamics and the projection of the time-derivative of the full-order state onto a low-dimensional subspace. Usually, this subspace is defined by the span of POD modes,

[1] A non-normal linear operator is one whose right eigenvectors are not mutually orthogonal, and, in the context of fluids, non-normality is due to the presence of the advective transport terms in the Navier-Stokes equation.

and the high-dimensional data are projected orthogonally onto it. While Operator Inference has been shown to work well for systems that evolve in close proximity of an attractor (see, e.g., [29]), it may suffer from the aforementioned drawbacks of orthogonal projections when applied to highly non-normal systems evolving far away from an attractor (e.g., during transients). This will become apparent in the examples sections. In the interest of completeness, it is worth mentioning that the Operator Inference framework is not limited to linear spaces. In fact, Operator Inference reduced-order models were recently computed after orthogonally projecting the data onto quadratic manifolds [14, 6], and extensions of the Operator Inference formulation were developed to preserve the underlying structure or symmetries of the full-order model [35, 15, 18]. We conclude our brief review by acknowledging that there exist several other non-intrusive model reduction frameworks in the literature (e.g., discrete-time formulations such as dynamic mode decomposition (DMD), autoencoders parameterized via neural networks, and many others), and we will mention those that are more closely connected with our formulation as needed throughout the manuscript.

In this paper, we introduce a novel non-intrusive framework to address the problems associated with orthogonal projections. In particular, given training trajectories from the full-order model, we fit an optimal low-order model by simultaneously seeking reduced-order dynamics $\mathbf{f}_r$ and oblique projection operators $\mathbb{P}$ defined by a linear encoder $\mathbf{\Psi}^\intercal$ and a linear decoder $\mathbf{\Phi}\left(\mathbf{\Psi}^\intercal\mathbf{\Phi}\right)^{-1}$. We shall see that the optimization parameters are the subspace $V = \mathrm{Range}(\mathbf{\Phi})$, which lives naturally on the Grassmann manifold, the matrix $\mathbf{\Psi}$, which can be taken to be an element of the orthogonal Stiefel manifold, and the parameters that define the reduced-order dynamics (e.g., reduced-order tensors if the dynamics are taken to be polynomial). Furthermore, if we constrain the reduced-order dynamics $\mathbf{f}_r$ to be of a form that lends itself to straightforward differentiation (e.g., polynomial), we show that the gradient of the cost function with respect to the optimization parameters can be written in closed form. This is quite convenient because it bypasses the need for automatic differentiation and it allows for faster training. We test our formulation on three different examples: a simple system governed by three ordinary differential equations, the complex Ginzburg-Landau (CGL) equation and the two-dimensional incompressible lid-driven cavity flow at Reynolds number $Re = 8300$. On all three examples, we compare our framework with Operator Inference and POD-Galerkin. In the first two examples, we also compare with TrOOP, which has been shown to give very accurate Petrov-Galerkin models in several examples, including highly non-normal and nonlinear jets [25, 26]. On all three examples, our models exhibit better performance than Operator Inference and POD-Galerkin models, and in the first two examples we obtain models with predictive accuracy very close to that of the intrusive TrOOP formulation.

2. Mathematical formulation.

Throughout this section, we consider a general nonlinear system with dynamics defined by

$$\begin{aligned} \frac{\mathrm{d}\mathbf{x}}{\mathrm{d}t} &= \mathbf{f}(\mathbf{x},\mathbf{u}), \quad \mathbf{x}(0) = \mathbf{x}_0 \\ \mathbf{y} &= \mathbf{h}(\mathbf{x}) \end{aligned} \tag{2.1}$$

where $\mathbf{x} \in \mathbb{R}^n$ is the state vector, $\mathbf{x}_0$ is the initial condition, $\mathbf{u} \in \mathbb{R}^m$ is the control input and $\mathbf{y} \in \mathbb{R}^p$ is the measured output. Since our model reduction procedure draws inspiration from the form of Petrov-Galerkin reduced-order models, we begin by providing a brief review of the latter. We then introduce our framework in section 2.2.

2.1. Petrov-Galerkin models.

As discussed in the introduction, Petrov-Galerkin reduced-order models are a class of models obtained by constraining the full-order dynamics in (2.1) to a linear subspace of $\mathbb{R}^n$. While Petrov-Galerkin models can also be obtained via nonlinear projection onto curved manifolds [24], here we constrain our attention to the more common case of linear projections. Given rank-r matrices $\boldsymbol{\Phi} \in \mathbb{R}^{n\times r}$ and $\boldsymbol{\Psi} \in \mathbb{R}^{n\times r}$ that define an *oblique* projection $\mathbb{P} = \boldsymbol{\Phi}\left(\boldsymbol{\Psi}^\intercal\boldsymbol{\Phi}\right)^{-1}\boldsymbol{\Psi}^\intercal$, the corresponding Petrov-Galerkin model for (2.1) is given by

$$\begin{aligned} \frac{\mathrm{d}\hat{\mathbf{x}}}{\mathrm{d}t} &= \mathbb{P}\mathbf{f}\left(\mathbb{P}\hat{\mathbf{x}},\mathbf{u}\right), \quad \hat{\mathbf{x}}(0) = \mathbb{P}\mathbf{x}_0 \\ \hat{\mathbf{y}} &= \mathbf{h}\left(\mathbb{P}\hat{\mathbf{x}}\right), \end{aligned} \tag{2.2}$$

where $\hat{\mathbf{x}}$ lies in the range of $\mathbb{P}$ for all times. In the special case of $\boldsymbol{\Psi} = \boldsymbol{\Phi}$, the projection $\mathbb{P}$ is orthogonal and the model (2.2) is referred to as a Galerkin model. While the state $\hat{\mathbf{x}} \in \mathbb{R}^n$ is an n-dimensional vector (i.e., the same size of the original state $\mathbf{x}$), the dynamics (2.2) can be realized by an equivalent r-dimensional system

$$\begin{aligned} \frac{\mathrm{d}\hat{\mathbf{z}}}{\mathrm{d}t} &= \boldsymbol{\Psi}^\intercal\mathbf{f}\left(\boldsymbol{\Phi}\left(\boldsymbol{\Psi}^\intercal\boldsymbol{\Phi}\right)^{-1}\hat{\mathbf{z}},\mathbf{u}\right), \quad \hat{\mathbf{z}}(0) = \boldsymbol{\Psi}^\intercal\mathbf{x}_0 \\ \hat{\mathbf{y}} &= \mathbf{h}\left(\boldsymbol{\Phi}\left(\boldsymbol{\Psi}^\intercal\boldsymbol{\Phi}\right)^{-1}\hat{\mathbf{z}}\right) \end{aligned} \tag{2.3}$$

where the state vector $\hat{\mathbf{z}} = \boldsymbol{\Psi}^\intercal\hat{\mathbf{x}}$ has dimension r. The primary challenge associated with computing accurate projection-based reduced-order models lies in identifying matrices $\boldsymbol{\Phi}$ and $\boldsymbol{\Psi}$ that define appropriate projections $\mathbb{P}$. While there exist several methods to address this challenge, these are often intrusive in the sense that they require access to the linearization of (2.1) and its adjoint [30, 25, 26]. In the next section, we present a non-intrusive model reduction formulation by allowing for the reduced-order dynamics to be independent of the full-order right-hand side $\mathbf{f}$.

2.2. Non-intrusive optimization of projection operators and reduced-order dynamics.

Here, we consider reduced-order models of the form

$$G(\boldsymbol{\Phi},\boldsymbol{\Psi},\hat{\mathbf{f}}_r) = \begin{cases} \dfrac{\mathrm{d}\hat{\mathbf{z}}}{\mathrm{d}t} = \mathbf{f}_r\left(\hat{\mathbf{z}},\mathbf{u}\right), \quad \hat{\mathbf{z}}(0) = \boldsymbol{\Psi}^\intercal\mathbf{x}_0 \\ \hat{\mathbf{y}} = \mathbf{h}\left(\boldsymbol{\Phi}\left(\boldsymbol{\Psi}^\intercal\boldsymbol{\Phi}\right)^{-1}\hat{\mathbf{z}}\right) \end{cases} \tag{2.4}$$

It is instructive to observe that if $\mathbf{f}_r(\hat{\mathbf{z}},\mathbf{u}) = \boldsymbol{\Psi}^\intercal\mathbf{f}\left(\boldsymbol{\Phi}\left(\boldsymbol{\Psi}^\intercal\boldsymbol{\Phi}\right)^{-1}\hat{\mathbf{z}},\mathbf{u}\right)$ then (2.4) is the exact analog of the Petrov-Galerkin reduced-order model in (2.3). Instead, we let $\mathbf{f}_r$ be a general function of the reduced-order state $\hat{\mathbf{z}}$ and of the input $\mathbf{u}$. So, while Petrov-Galerkin models are fully defined by (the span of) the matrices $\boldsymbol{\Phi}$ and $\boldsymbol{\Psi}$ that define a projection onto a low-dimensional subspace, here we have additional degrees of freedom in the choice of the reduced-order dynamics. We shall see momentarily that this additional freedom allows us to proceed non-intrusively.

Within our framework, we seek reduced-order models of the form of (2.4) by minimizing the error between ground-truth observations $\mathbf{y}$ coming from (2.1) and the predicted observations $\hat{\mathbf{y}}$ given by (2.4). In order to convert this task into an appropriate optimization problem, it is useful to first identify the symmetries and constraints that are present in (2.4). We begin by observing that the system G in (2.4) is invariant with respect to a rotation and scaling of the basis matrix $\mathbf{\Phi}$. In fact, $G(\mathbf{\Phi R}, \mathbf{\Psi}, \mathbf{f}_r) = G(\mathbf{\Phi}, \mathbf{\Psi}, \mathbf{f}_r)$ for any invertible matrix $\mathbf{R}$ of size $r \times r$. It follows that the reduced-order system defined by (2.4) is a function of the r-dimensional subspace $V = \text{Range}(\mathbf{\Phi})$, rather than of the matrix representative $\mathbf{\Phi}$ itself. In the mathematical statement of the problem we will make use of this symmetry and leverage the fact that r-dimensional subspaces of $\mathbb{R}^n$ are elements of the Grassmann manifold $\mathcal{G}_{n,r}$. An analogous type of symmetry does *not* hold for $\mathbf{\Psi}$. In fact, it can be easily verified that there exist invertible matrices $\mathbf{S}$ such that $G(\mathbf{\Phi}, \mathbf{\Psi S}, \mathbf{f}_r) \neq G(\mathbf{\Phi}, \mathbf{\Psi}, \mathbf{f}_r)$. While (2.4) does not enjoy any $\mathbf{\Psi}$-symmetries, we still require $\mathbf{\Psi}$ to have full column rank (otherwise the product $\mathbf{\Psi}^\intercal\mathbf{\Phi}$ would be rank deficient). It is therefore natural to constrain $\mathbf{\Psi}$ to the Stiefel manifold $S_{n,r}$ of orthonormal (and, hence, full-rank) $n \times r$ matrices. Finally, in order to write an optimization problem where the gradient of the cost function with respect to all the parameters can be obtained in closed form, it is convenient to constrain the reduced-order dynamics $\mathbf{f}_r$ to a form that lends itself to straightforward differentiation. Throughout this paper, we will let $\mathbf{f}_r$ be a polynomial function of the reduced-order state $\hat{\mathbf{z}}$ and of the input $\mathbf{u}$ as follows

$$\mathbf{f}_r = \underbrace{\mathbf{A}_r\hat{\mathbf{z}} + \mathbf{B}_r\mathbf{u} + \mathbf{H}_r : \hat{\mathbf{z}}\hat{\mathbf{z}}^\intercal}_{:=\overline{\mathbf{f}}_r} + \mathbf{L}_r : \hat{\mathbf{z}}\mathbf{u}^\intercal + \dots . \tag{2.5}$$

Here, capital letters denote reduced-order tensors that lie naturally on linear manifolds of appropriate dimension (e.g., $\mathbf{A}_r \in \mathbb{R}^{r\times r}$, $\mathbf{B}_r \in \mathbb{R}^{r\times m}$ and $\mathbf{H}_r \in \mathbb{R}^{r\times r\times r}$). In the interest of a more concise description of the mathematical formulation, we take $\mathbf{f}_r = \overline{\mathbf{f}}_r$ (see definition of $\overline{\mathbf{f}}_r$ in the underbrace of equation (2.5)). Higher-order polynomial dynamics can be considered with minimal modification.

We are now ready to state the optimization problem that will give us an optimal reduced-order model of the form of (2.4). Given outputs $\mathbf{y}(t_i)$ sampled at times t_i along a trajectory generated from the full-order system (2.1), we seek a solution to

$$\begin{aligned} \min_{(V,\mathbf{\Psi},\mathbf{A}_r,\mathbf{H}_r,\mathbf{B}_r)\in\mathcal{M}} \quad & J = \sum_{i=0}^{N-1} \|\mathbf{y}(t_i) - \hat{\mathbf{y}}(t_i)\|^2 \\ \text{subject to:} \quad & \frac{\mathrm{d}\hat{\mathbf{z}}}{\mathrm{d}t} = \overline{\mathbf{f}}_r(\hat{\mathbf{z}}, \mathbf{u}), \quad \hat{\mathbf{z}}(t_0) = \mathbf{\Psi}^\intercal\mathbf{x}(t_0) \\ & \hat{\mathbf{y}} = \mathbf{h}\left(\mathbf{\Phi}\left(\mathbf{\Psi}^\intercal\mathbf{\Phi}\right)^{-1}\hat{\mathbf{z}}\right) \\ & V = \text{Range}\left(\mathbf{\Phi}\right) \end{aligned} \tag{2.6}$$

where $\mathcal{M} = \mathcal{G}_{n,r} \times S_{n,r} \times \mathbb{R}^{r\times r} \times \mathbb{R}^{r\times r\times r} \times \mathbb{R}^{r\times m}$ is the product manifold that defines our optimization domain.

2.3. Gradient-based optimization on $\mathcal{M}$.

In order to solve the optimization problem (2.6) using a gradient-based algorithm, it is convenient to view $\mathcal{M}$ as a submanifold of an *ambient-*

space manifold $\overline{\mathcal{M}}$ endowed with a Riemannian metric. We first define $\overline{\mathcal{M}}$ for our specific case, then we discuss the Riemannian metric.

Since $\mathcal{M}$ is a product manifold whose topology is the product topology of its individual components, the ambient-space manifold $\overline{\mathcal{M}}$ can also be defined component-wise. Following [1], we view the Stiefel manifold as an embedded submanifold of the vector space $\mathbb{R}^{n\times r}$, and the Grassmann manifold $\mathcal{G}_{n,r}$ as a quotient manifold of the non-orthogonal Stiefel manifold $\mathbb{R}_*^{n\times r}$ (which is the manifold of rank-r, but non necessarily orthonormal, matrices of size $n \times r$). The manifolds $\mathbb{R}^{r\times r}$, $\mathbb{R}^{r\times r\times r}$ and $\mathbb{R}^{r\times m}$ are vector spaces that do not require any special treatment, so $\overline{\mathcal{M}}$ may finally be defined as

$$\overline{\mathcal{M}} = \mathbb{R}_*^{n\times r} \times \mathbb{R}^{n\times r} \times \mathbb{R}^{r\times r} \times \mathbb{R}^{r\times r\times r} \times \mathbb{R}^{r\times m}. \tag{2.7}$$

In order to define the gradient of the cost function with respect to the parameters, we now endow the ambient-space manifold with a Riemannian metric, which will then be inherited by the optimization manifold $\mathcal{M}$. Formally, a Riemannian metric $g^{\mathcal{M}}$ is a smooth family of inner products $g_p^{\mathcal{M}}$ defined on the tangent spaces of the manifold $\mathcal{M}$,

$$g_p^{\mathcal{M}} : \mathcal{T}_p\mathcal{M} \times \mathcal{T}_p\mathcal{M} \to \mathbb{R}, \tag{2.8}$$

where $\mathcal{T}_p\mathcal{M}$ denotes the tangent space of $\mathcal{M}$ at a point $p \in \mathcal{M}$ [1]. The gradient ξ of the cost function at $p \in \mathcal{M}$ is then defined as the element of the tangent space $\mathcal{T}_p\mathcal{M}$ that satisfies

$$DJ[\eta] = g_p^{\mathcal{M}}\left(\xi, \eta\right), \quad \forall \eta \in \mathcal{T}_p^{\mathcal{M}}, \tag{2.9}$$

where $DJ[\eta]$ is the directional derivative. A metric for a product manifold can be defined as the sum of the component metrics, so we can proceed component-wise as before. The metric for the Stiefel manifold $S_{n,r}$ can be defined as

$$g_{\mathbf{\Psi}}^{S_{n,r}}\left(\xi, \eta\right) = \mathrm{Tr}\left(\xi^\intercal \eta\right), \quad \xi,\, \eta \in \mathcal{T}_{\mathbf{\Psi}} S_{n,r}, \tag{2.10}$$

which is the Euclidean metric inherited from the ambient space $\mathbb{R}^{n\times r}$ [1] and Tr denotes the trace. A metric for the Grassmann manifold can be defined analogously, albeit paying attention to the fact that the Grassmannian is an abstract manifold with non-unique matrix representatives. In particular, given the ambient space metric

$$g_{\mathbf{\Phi}}^{\mathbb{R}_*^{n\times r}}\left(\xi, \eta\right) = \mathrm{Tr}\left(\left(\mathbf{\Phi}^\intercal\mathbf{\Phi}\right)^{-1}\xi^\intercal\eta\right), \quad \xi,\, \eta \in \mathcal{T}_{\mathbf{\Phi}}\mathbb{R}_*^{n\times r} \tag{2.11}$$

we let the metric on $\mathcal{G}_{n,r}$ be defined as

$$g_V^{\mathcal{G}_{n,r}}(\xi, \eta) = g_{\mathbf{\Phi}}^{\mathbb{R}_*^{n\times r}}\left(\overline{\xi}_{\mathbf{\Phi}}, \overline{\eta}_{\mathbf{\Phi}}\right), \quad \xi,\, \eta \in \mathcal{T}_V, \quad \mathrm{Range}\left(\mathbf{\Phi}\right) = V. \tag{2.12}$$

It is worth observing that (2.12) is not yet suited for computation, since there exists an infinite number of elements $\overline{\xi}_{\mathbf{\Phi}}$ and $\overline{\eta}_{\mathbf{\Phi}}$ of $\mathcal{T}_{\mathbf{\Phi}}\mathbb{R}_*^{n\times r}$ that satisfy the equality. The ambiguity is resolved by requiring $\overline{\xi}_{\mathbf{\Phi}}$ and $\overline{\eta}_{\mathbf{\Phi}}$ to lie on the *horizontal space*, which is a subspace of $\mathcal{T}_{\mathbf{\Phi}}\mathbb{R}_*^{n\times r}$ where one may identify unique $\overline{\xi}_{\mathbf{\Phi}}$ and $\overline{\eta}_{\mathbf{\Phi}}$ that satisfy (2.12). This unique vector $\overline{\boldsymbol{\xi}}_{\mathbf{\Phi}}$ is known as the *horizontal lift* of $\boldsymbol{\xi}$ at $\boldsymbol{\Phi}$. A rigorous characterization of the horizontal space is provided in chapter 3 of [1], and the specific case of the Grassmann manifold is considered in example 3.6.4 in the same reference. Finally, for the linear manifolds in the Cartesian product of $\mathcal{M}$, we adopt the Euclidean metric (i.e., the usual tensor dot product).

Now that we have defined the ambient-space manifold $\overline{\mathcal{M}}$ and metrics on $\mathcal{M}$, we can approach the computation of the gradient of the cost function in terms of ambient-space matrix-valued objects, rather than abstract elements of the optimization manifold $\mathcal{M}$. In order to do so, we invoke the "canonical projection" [1]

$$\pi : \widetilde{\mathcal{M}} \to \mathcal{M} : (\boldsymbol{\Phi}, \boldsymbol{\Psi}, \mathbf{A}_r, \mathbf{H}_r, \mathbf{B}_r) \mapsto \left(\mathrm{Range}\left(\boldsymbol{\Phi}\right), \boldsymbol{\Psi}, \mathbf{A}_r, \mathbf{H}_r, \mathbf{B}_r\right), \tag{2.13}$$

where $\widetilde{\mathcal{M}} = \mathbb{R}_*^{n\times r} \times St_{n,r} \times \mathbb{R}^{r\times r} \times \mathbb{R}^{r\times r\times r} \times \mathbb{R}^{r\times m}$. Then, given our cost function $J : \mathcal{M} \to \mathbb{R}$, for any point $(V, \boldsymbol{\Psi}, \mathbf{A}_r, \mathbf{H}_r, \mathbf{B}_r) \in \mathcal{M}$ we have

$$J(V, \boldsymbol{\Psi}, \mathbf{A}_r, \mathbf{H}_r, \mathbf{B}_r) = J\left(\pi\left(\boldsymbol{\Phi}, \boldsymbol{\Psi}, \mathbf{A}_r, \mathbf{H}_r, \mathbf{B}_r\right)\right) = \overline{J}\left(\boldsymbol{\Phi}, \boldsymbol{\Psi}, \mathbf{A}_r, \mathbf{H}_r, \mathbf{B}_r\right), \tag{2.14}$$

where $(\boldsymbol{\Phi}, \boldsymbol{\Psi}, \mathbf{A}_r, \mathbf{H}_r, \mathbf{B}_r) \in \widetilde{\mathcal{M}}$ and $V = \mathrm{Range}(\boldsymbol{\Phi})$. If we view $\overline{J} : \overline{\mathcal{M}} \to \mathbb{R}$ as a function that sends elements of the ambient space to the reals, then equation (2.14) implies that J on $\mathcal{M}$ is equal to the *restriction* of $\overline{J}$ to $\widetilde{\mathcal{M}}$. This restriction ensures that the second argument of $\overline{J}$ is an element of the Stiefel manifold (as opposed to a generic element of $\mathbb{R}^{n\times r}$). We henceforth refer to $\overline{J}$ as the ambient-space cost function. It follows from standard results (see equations (3.37) and (3.39) in [1]) that

$$\left(\overline{\nabla_V J_{\boldsymbol{\Phi}}}, \nabla_{\boldsymbol{\Psi}} J, \nabla_{\mathbf{A}_r} J, \nabla_{\mathbf{H}_r} J, \nabla_{\mathbf{B}_r} J\right) = \left(\nabla_{\boldsymbol{\Phi}} \overline{J}, \mathbb{P}_{\boldsymbol{\Psi}} \nabla_{\overline{\boldsymbol{\Psi}}} \overline{J}, \nabla_{\mathbf{A}_r} \overline{J}, \nabla_{\mathbf{H}_r} \overline{J}, \nabla_{\mathbf{B}_r} \overline{J}\right), \tag{2.15}$$

where the gradient of $\overline{J}$ is evaluated at $(\boldsymbol{\Phi}, \boldsymbol{\Psi}, \mathbf{A}_r, \mathbf{H}_r, \mathbf{B}_r) \in \widetilde{\mathcal{M}}$, and the gradient of J is evaluated at $(V, \boldsymbol{\Psi}, \mathbf{A}_r, \mathbf{H}_r, \mathbf{B}_r) \in \mathcal{M}$ with $V = \mathrm{Range}(\boldsymbol{\Phi})$. Here, $\overline{\nabla_V J_{\boldsymbol{\Phi}}}$ denotes the horizontal lift of $\nabla_V J$ at $\boldsymbol{\Phi}$, $\mathbb{P}_{\boldsymbol{\Psi}}$ denotes the projection onto the tangent space of $St_{n,r}$ at $\boldsymbol{\Psi}$ (see example 3.6.2 in [1]) and we remark that $\nabla_{\overline{\boldsymbol{\Psi}}} \overline{J}$ is an element of the tangent space of $\mathbb{R}^{n\times r}$ at $\boldsymbol{\Psi}$. In summary, the equation above states that the gradient of the cost function with respect to the abstract optimization parameters can be computed in terms of the gradient of the ambient-space cost function. Conveniently, our model reduction formulation allows for the ambient-space gradient to be computed in closed form, and this result is stated in the proposition below. Importantly, we shall see that the computation of the gradient does not require querying the full-order model (2.1). That is, the gradient can be computed non-intrusively. Once the ambient-space gradient is available, the gradient with respect to the optimization parameters is computed using (2.15) by libraries such as `Pymanopt` [36] in Python or `Manopt` [7] in MATLAB.

Proposition 2.1 (Ambient-space gradient). *Let problem* (2.6) *be written as an equivalent unconstrained optimization problem with ambient-space Lagrangian* $\overline{L} : \overline{\mathcal{M}} \to \mathbb{R}$ *defined as*

$$\begin{aligned} \overline{L}(\boldsymbol{\Phi}, \overline{\boldsymbol{\Psi}}, \mathbf{A}_r, \mathbf{H}_r, \mathbf{B}_r) = \sum_{i=0}^{N-1} \Bigg\{ & \overline{J}_i + \int_{t_0}^{t_i} \boldsymbol{\lambda}_i^{\intercal} \left(\frac{\mathrm{d}\hat{\mathbf{z}}}{\mathrm{d}t} - \mathbf{A}_r \hat{\mathbf{z}} - \mathbf{H}_r : \hat{\mathbf{z}}\hat{\mathbf{z}}^{\intercal} - \mathbf{B}_r \mathbf{u} \right) \mathrm{d}t \\ & + \boldsymbol{\lambda}_i(t_0)^{\intercal} \left(\hat{\mathbf{z}}(t_0) - \overline{\boldsymbol{\Psi}}^{\intercal} \mathbf{x}(t_0) \right) \Bigg\}, \end{aligned} \tag{2.16}$$

where $\overline{J}_i = \left\|\mathbf{y}(t_i) - \mathbf{h}\left(\mathbf{\Phi}\left(\overline{\mathbf{\Psi}}^\intercal\mathbf{\Phi}\right)^{-1}\hat{\mathbf{z}}(t_i)\right)\right\|^2$ *and* $\boldsymbol{\lambda}_i(t) \in \mathbb{R}^r$ *with* $t \in [t_0, t_i]$ *is the ith Lagrange multiplier. Defining* $\mathbf{e}(t_i) \coloneqq \mathbf{y}(t_i) - \mathbf{h}\left(\mathbf{\Phi}\left(\overline{\mathbf{\Psi}}^\intercal\mathbf{\Phi}\right)^{-1}\hat{\mathbf{z}}(t_i)\right)$ *and* $\mathbf{C}_{j,k} \coloneqq \partial\mathbf{h}_j/\partial\mathbf{x}_k$, *the gradients of the ambient-space Lagrangian with respect to its parameters are given below,*

$$\nabla_{\mathbf{\Phi}}\overline{L} = \left\{-2\sum_{i=0}^{N-1}\left(\mathbf{I} - \overline{\mathbf{\Psi}}\left(\mathbf{\Phi}^\intercal\overline{\mathbf{\Psi}}\right)^{-1}\mathbf{\Phi}^\intercal\right)\mathbf{C}(t_i)^\intercal\mathbf{e}(t_i)\hat{\mathbf{z}}(t_i)^\intercal\left(\overline{\mathbf{\Psi}}^\intercal\mathbf{\Phi}\right)^{-1}\right\}(\mathbf{\Phi}^\intercal\mathbf{\Phi}) \tag{2.17}$$

$$\nabla_{\overline{\mathbf{\Psi}}}\overline{L} = \sum_{i=0}^{N-1}\left(2\mathbf{\Phi}\left(\overline{\mathbf{\Psi}}^\intercal\mathbf{\Phi}\right)^{-1}\hat{\mathbf{z}}(t_i)\mathbf{e}(t_i)^\intercal\mathbf{C}(t_i)\mathbf{\Phi}\left(\overline{\mathbf{\Psi}}^\intercal\mathbf{\Phi}\right)^{-1} - \mathbf{x}(t_0)\boldsymbol{\lambda}_i(t_0)^\intercal\right) \tag{2.18}$$

$$\nabla_{\mathbf{A}_r}\overline{L} = -\sum_{i=0}^{N-1}\int_{t_0}^{t_i}\boldsymbol{\lambda}_i\hat{\mathbf{z}}^\intercal\,\mathrm{d}t \tag{2.19}$$

$$\nabla_{\mathbf{H}_r}\overline{L} = -\sum_{i=0}^{N-1}\int_{t_0}^{t_i}\boldsymbol{\lambda}_i\otimes\hat{\mathbf{z}}\otimes\hat{\mathbf{z}}\,\mathrm{d}t \tag{2.20}$$

$$\nabla_{\mathbf{B}_r}\overline{L} = -\sum_{i=0}^{N-1}\int_{t_0}^{t_i}\boldsymbol{\lambda}_i\mathbf{u}^\intercal\,\mathrm{d}t, \tag{2.21}$$

where the Lagrange multiplier $\boldsymbol{\lambda}_i(t)$ *satisfies the reduced-order adjoint equation*

$$-\frac{\mathrm{d}\boldsymbol{\lambda}_i}{\mathrm{d}t} = \left[\partial_{\hat{\mathbf{z}}}\overline{\mathbf{f}}_r(\hat{\mathbf{z}})\right]^\intercal\boldsymbol{\lambda}_i, \quad \boldsymbol{\lambda}_i(t_i) = 2\left(\mathbf{\Phi}^\intercal\overline{\mathbf{\Psi}}\right)^{-1}\mathbf{\Phi}^\intercal\mathbf{C}(t_i)^\intercal\mathbf{e}(t_i), \quad t \in [t_0, t_i]. \tag{2.22}$$

Proof. The proof relies on calculus of variations. At a local minimum $p \in \overline{\mathcal{M}}$, the following must hold for every vector $\xi \in \mathcal{T}_p\overline{\mathcal{M}}$,

$$g_p^{\overline{\mathcal{M}}}\left(\nabla_p\overline{L}, \xi\right) = D_p\overline{L}[\xi] = \partial_p\overline{L}[\xi] + \partial_{\hat{\mathbf{z}}}\overline{L}\cdot D_p\hat{\mathbf{z}}[\xi] + \sum_{i=0}^{N-1}\left(\partial_{\boldsymbol{\lambda}_i}\overline{L}\cdot D_p\boldsymbol{\lambda}_i[\xi]\right) = 0, \tag{2.23}$$

where $g_p^{\overline{\mathcal{M}}}$ denotes the ambient-space metric on $\overline{\mathcal{M}}$ at p (which we have defined componentwise earlier in section 2.3). By enforcing $\partial_{\hat{\mathbf{z}}}\overline{L}[\boldsymbol{\eta}] = 0$ for all $\boldsymbol{\eta}$, the equality above reduces to

$$g_p^{\overline{\mathcal{M}}}\left(\nabla_p\overline{L}, \xi\right) = \partial_p\overline{L}[\xi] = 0, \tag{2.24}$$

since $\partial_{\boldsymbol{\lambda}_i}\overline{L} = 0$ for all i by virtue of the fact that $\boldsymbol{\lambda}_i$ is a Lagrange multiplier. We begin by showing that the reduced-order adjoint equation (2.22) enforces $\partial_{\hat{\mathbf{z}}}\overline{L}[\boldsymbol{\eta}] = 0$ for all $\boldsymbol{\eta}$. Given the ambient-space Lagrangian $\overline{L}$, we have

$$\begin{aligned}\partial_{\hat{\mathbf{z}}}\overline{L}[\boldsymbol{\eta}] = \sum_{i=0}^{N-1}\Bigg\{&-2\mathbf{e}(t_i)^\intercal\mathbf{C}(t_i)\mathbf{\Phi}\left(\overline{\mathbf{\Psi}}^\intercal\mathbf{\Phi}\right)^{-1}\boldsymbol{\eta}(t_i) + \boldsymbol{\lambda}_i^\intercal\boldsymbol{\eta}\Big|_{t_0}^{t_i} - \int_{t_0}^{t_i}\left(\frac{\mathrm{d}\boldsymbol{\lambda}_i^\intercal}{\mathrm{d}t} + \boldsymbol{\lambda}_i^\intercal\left[\partial_{\hat{\mathbf{z}}}\overline{\mathbf{f}}_r(\hat{\mathbf{z}})\right]\right)\boldsymbol{\eta}\,\mathrm{d}t\\ &+\boldsymbol{\lambda}_i(t_0)^\intercal\boldsymbol{\eta}(t_0)\Bigg\} = 0,\end{aligned} \tag{2.25}$$

where we have used integration by parts on the time-derivative term. For each $i > 0$, the terms $\boldsymbol{\lambda}_i(t_0)^\intercal\boldsymbol{\eta}(t_0)$ cancel out and the summand vanishes thanks to equation (2.22). Similarly, when $i = 0$, the second and third terms in the sum vanish and the summand is equal to zero for $\boldsymbol{\lambda}_0(t_0) = 2\left(\boldsymbol{\Phi}^\intercal\overline{\boldsymbol{\Psi}}\right)^{-1}\boldsymbol{\Phi}^\intercal\mathbf{C}(t_0)^\intercal\mathbf{e}(t_0)$. We now derive the gradient of $\overline{L}$ with respect to $\boldsymbol{\Phi}$. The partial derivative of $\overline{L}$ with respect to $\boldsymbol{\Phi}$ in the direction of $\boldsymbol{\xi}$ is given by

$$\partial_{\boldsymbol{\Phi}}\overline{L}[\boldsymbol{\xi}] = -2\sum_{i=0}^{N-1}\mathbf{e}(t_i)^\intercal\mathbf{C}(t_i)\left(\boldsymbol{\xi}\left(\overline{\boldsymbol{\Psi}}^\intercal\boldsymbol{\Phi}\right)^{-1} - \boldsymbol{\Phi}\left(\overline{\boldsymbol{\Psi}}^\intercal\boldsymbol{\Phi}\right)^{-1}\overline{\boldsymbol{\Psi}}^\intercal\boldsymbol{\xi}\left(\overline{\boldsymbol{\Psi}}^\intercal\boldsymbol{\Phi}\right)^{-1}\right)\hat{\mathbf{z}}(t_i), \tag{2.26}$$

where we have used the identity $D_{\boldsymbol{\Phi}}\left(\overline{\boldsymbol{\Psi}}^\intercal\boldsymbol{\Phi}\right)^{-1}[\boldsymbol{\xi}] = -\left(\overline{\boldsymbol{\Psi}}^\intercal\boldsymbol{\Phi}\right)^{-1}\overline{\boldsymbol{\Psi}}^\intercal\boldsymbol{\xi}\left(\overline{\boldsymbol{\Psi}}^\intercal\boldsymbol{\Phi}\right)^{-1}$. Using (2.24), and recalling the definition of the ambient-space metric on $\mathbb{R}_*^{n\times r}$ (2.11), we recover the gradient in (2.17). The other gradients can be obtained similarly and the proof is concluded. ■

Another ingredient that is necessary for gradient-based manifold optimization is the concept of a *retraction*. This is a map $R_p : \mathcal{T}_p\mathcal{M} \to \mathcal{M}$ that satisfies $R_p(0) = p$ and $DR_p(0) = I_{\mathcal{T}_p\mathcal{M}}$, where $I_{\mathcal{T}_p\mathcal{M}}$ is the identity map on the tangent space $\mathcal{T}_p\mathcal{M}$ [1]. The use of this map allows us to generalize the concept of moving in the direction of the gradient on a nonlinear manifold: for instance, given a point $p \in \mathcal{M}$ and the gradient $\xi \in \mathcal{T}_p\mathcal{M}$ of a function f defined on $\mathcal{M}$, the next iterate in the direction of the gradient is given by $R_p(p - \alpha\xi) \in \mathcal{M}$, where α is some learning rate. In other words, the retraction allows us to guarantee that all iterates generated by a gradient flow lie on the manifold. Valid retractions for both the Stiefel and Grassmann manifolds are given by the QR decomposition (see examples 4.1.3 and 4.1.5 in [1]), while for linear manifolds the retraction is simply the identity map. Lastly, we point out that second-order gradient-based algorithms (e.g., conjugate gradient) require the concept of *vector transport*. This is described thoroughly in section 8.1 of [1]. Gradient-based algorithms on nonlinear manifolds are well-understood and readily available in libraries such as `Pymanopt` [36] and `Manopt` [7]. Metrics, retractions and vector transports are conveniently handled by these packages, and a user simply needs to provide routines to evaluate the cost function and the ambient-space gradient provided in Proposition 2.1.

2.4. Computational considerations.

In this subsection, we discuss the efficient computation of the ambient-space gradient presented in Proposition 2.1. We then provide an algorithm and and estimate of the computational cost.

In order to efficiently calculate the gradient, it is useful to manipulate the expressions in (2.19)-(2.21) into a form that is more suitable for computation. In particular, since the integrands in (2.19)-(2.21) are linear in $\boldsymbol{\lambda}_i$, we can write, e.g.,

$$\sum_{i=0}^{N-1}\int_{t_0}^{t_i}\boldsymbol{\lambda}_i(t)\hat{\mathbf{z}}(t)^\intercal\,\mathrm{d}t = \sum_{i=1}^{N-1}\int_{t_{i-1}}^{t_i}\boldsymbol{\xi}_i(t)\hat{\mathbf{z}}(t)^\intercal\,\mathrm{d}t, \quad \boldsymbol{\xi}_i(t) = \sum_{j=i}^{N-1}\boldsymbol{\lambda}_j(t), \tag{2.27}$$

where $\boldsymbol{\xi}_i(t)$ may be understood as a cumulative adjoint variable that can be computed by time-stepping the adjoint equation (2.22) backward in time from t_{N-1} to t_0. Then, according to the equation above, the gradients in (2.19)-(2.21) can be conveniently computed as a sum of integrals over short temporal intervals $[t_{i-1}, t_i]$, as opposed to a sum of integrals over temporal intervals of increasing length $[t_0, t_i]$. Having defined $\boldsymbol{\xi}_i(t)$, the term $\sum_{i=0}^{N-1} \mathbf{x}(t_0)\boldsymbol{\lambda}_i(t_0)^\intercal = \mathbf{x}(t_0)\boldsymbol{\xi}_0(t_0)^\intercal$ in equation (2.18) can also be evaluated efficiently. These details are illustrated in Algorithm 2.1.

Algorithm 2.1 Compute ambient-space gradient in Proposition 2.1

```
Input: Training data $\{\mathbf{y}(t_i)\}_{i=0}^{N-1}$, initial condition $\mathbf{x}(0)$ of the full-order model, input $\mathbf{u}(t)$
       and a point $(V, \boldsymbol{\Psi}, \mathbf{A}_r, \mathbf{H}_r, \mathbf{B}_r) \in \mathcal{M}$, with some matrix representative $\boldsymbol{\Phi}$ such that
       Range$(\boldsymbol{\Phi}) = V$.
Output: Ambient-space gradients (2.17)-(2.21) in Proposition 2.1
 1: Initialize arrays to store $\nabla_{\boldsymbol{\Phi}}\overline{L}$, $\nabla_{\overline{\boldsymbol{\Psi}}}\overline{L}$, $\nabla_{\mathbf{A}_r}\overline{L}$, $\nabla_{\mathbf{H}_r}\overline{L}$ and $\nabla_{\mathbf{B}_r}\overline{L}$
 2: Compute the ROM solution $\hat{\mathbf{z}}(t)$ with initial condition $\boldsymbol{\Psi}^\intercal\mathbf{x}(0)$ and external input $\mathbf{u}(t)$
 3: Store values $\hat{\mathbf{z}}(t_i)$ (with $i \in \{0, 1, \ldots, N-1\}$), then compute $\mathbf{e}(t_i)$ and $\mathbf{C}(t_i)$ defined in
    Proposition 2.1
 4: for $i \in \{N-1, N-2, \ldots, 1\}$ do
 5:     Update $\nabla_{\boldsymbol{\Phi}}\overline{L} \leftarrow \nabla_{\boldsymbol{\Phi}}\overline{L} - 2\left(\mathbf{I} - \boldsymbol{\Psi}\left(\boldsymbol{\Phi}^\intercal\boldsymbol{\Psi}\right)^{-1}\boldsymbol{\Phi}^\intercal\right)\mathbf{C}(t_i)^\intercal\mathbf{e}(t_i)\hat{\mathbf{z}}(t_i)^\intercal\left(\boldsymbol{\Psi}^\intercal\boldsymbol{\Phi}\right)^{-1}\left(\boldsymbol{\Phi}^\intercal\boldsymbol{\Phi}\right)$
 6:     Update $\nabla_{\overline{\boldsymbol{\Psi}}}\overline{L} \leftarrow \nabla_{\overline{\boldsymbol{\Psi}}}\overline{L} + 2\boldsymbol{\Phi}\left(\boldsymbol{\Psi}^\intercal\boldsymbol{\Phi}\right)^{-1}\hat{\mathbf{z}}(t_i)\mathbf{e}(t_i)^\intercal\mathbf{C}(t_i)\boldsymbol{\Phi}\left(\boldsymbol{\Psi}^\intercal\boldsymbol{\Phi}\right)^{-1}$
 7:     Compute $\boldsymbol{\xi}_i(t)$ (see (2.27)) for $t \in [t_{i-1}, t_i]$ by integrating the adjoint equation (2.22)
        backward in time with final condition $\boldsymbol{\xi}_i(t_i) = \boldsymbol{\xi}_{i+1}(t_i) + 2\left(\boldsymbol{\Phi}^\intercal\boldsymbol{\Psi}\right)^{-1}\boldsymbol{\Phi}^\intercal\mathbf{C}(t_i)^\intercal\mathbf{e}(t_i)$
 8:     Update $\nabla_{\mathbf{A}_r}\overline{L} \leftarrow \nabla_{\mathbf{A}_r}\overline{L} - \int_{t_{i-1}}^{t_i}\boldsymbol{\xi}_i(t)\hat{\mathbf{z}}(t)^\intercal\,\mathrm{d}t$ using, e.g., Gaussian quadrature
 9:     Update $\nabla_{\mathbf{H}_r}\overline{L} \leftarrow \nabla_{\mathbf{H}_r}\overline{L} - \int_{t_{i-1}}^{t_i}\boldsymbol{\xi}_i(t) \otimes \hat{\mathbf{z}}(t) \otimes \hat{\mathbf{z}}(t)\,\mathrm{d}t$
10:     Update $\nabla_{\mathbf{B}_r}\overline{L} \leftarrow \nabla_{\mathbf{B}_r}\overline{L} - \int_{t_{i-1}}^{t_i}\boldsymbol{\xi}_i(t)\mathbf{u}(t)^\intercal\,\mathrm{d}t$
11: end for
12: Set $\boldsymbol{\xi}_0(t_0) \leftarrow \boldsymbol{\xi}_1(t_0) + 2\left(\boldsymbol{\Phi}^\intercal\boldsymbol{\Psi}\right)^{-1}\boldsymbol{\Phi}^\intercal\mathbf{C}(t_0)^\intercal\mathbf{e}(t_0)$
13: Update $\nabla_{\boldsymbol{\Phi}}\overline{L} \leftarrow \nabla_{\boldsymbol{\Phi}}\overline{L} - 2\left(\mathbf{I} - \boldsymbol{\Psi}\left(\boldsymbol{\Phi}^\intercal\boldsymbol{\Psi}\right)^{-1}\boldsymbol{\Phi}^\intercal\right)\mathbf{C}(t_0)^\intercal\mathbf{e}(t_0)\hat{\mathbf{z}}(t_0)^\intercal\left(\boldsymbol{\Psi}^\intercal\boldsymbol{\Phi}\right)^{-1}$
14: Update $\nabla_{\overline{\boldsymbol{\Psi}}}\overline{L} \leftarrow \nabla_{\overline{\boldsymbol{\Psi}}}\overline{L} + 2\boldsymbol{\Phi}\left(\boldsymbol{\Psi}^\intercal\boldsymbol{\Phi}\right)^{-1}\hat{\mathbf{z}}(t_0)\mathbf{e}(t_0)^\intercal\mathbf{C}(t_0)^\intercal\boldsymbol{\Phi}\left(\boldsymbol{\Psi}^\intercal\boldsymbol{\Phi}\right)^{-1} - \mathbf{x}(t_0)\boldsymbol{\xi}_0(t_0)^\intercal$
```

As far as computational cost is concerned, the algorithm scales with the number of snapshots N along a training trajectory, the ROM dimension r, the polynomial order of the ROM dynamics p, the size of the full-order state n, the number of time steps n_t to integrate the ROM from time t_i to t_{i+1}, and the number of quadrature points n_q used to estimate the temporal integrals. Given the presence of a for loop with $N-1$ iterations (line 4 in the algorithm), the overall cost of is $O(Nc)$, where c is the cost associated with each for loop iteration i. The major contributors to the latter are the presence of matrix-vector products involving $\boldsymbol{\Phi}$ and $\boldsymbol{\Psi}$ (which we recall being matrices of size $n \times r$), the need to integrate the reduced-order adjoint dynamics backward in time (line 7 in the algorithm), and the evaluation of the integrals involving $r-$dimensional tensor products (see, e.g., line 9). The cost of matrix-vector products involving $\boldsymbol{\Phi}$ and $\boldsymbol{\Psi}$ is $O(nr)$, the cost associated with integrating the adjoint equations is $O(n_t r^{p+1})$, where n_t is the number of time steps taken from t_i to t_{i+1}, and the evaluation of the integrals scales as $O(n_q r^{p+1})$, where n_q is the number of quadrature points. Usually, $n_q \ll n_t$ (this is the case if we use high-order Gaussian quadrature), so an estimate of the cost per for-loop iteration is given by $O(nr + n_t r^{p+1})$. In very high-dimensional systems where n is larger than $O(n_t r^p)$, the cost per iteration is dominated by the matrix-vector products involving $\mathbf{\Phi}$ and $\mathbf{\Psi}$, otherwise it is dominated by the ROM time stepper.

2.5. Connection with existing methods.

While our model reduction framework shares similarities with several existing methods, we would like to emphasize a natural connection with the recently-developed Trajectory-based Optimization for Oblique Projections (TrOOP) [25] and the Operator Inference framework introduced in [28].

TrOOP is a model reduction framework whereby a Petrov-Galerkin reduced-order model of the form (2.2) is obtained by optimizing the projection operator $\mathbb{P}$ against trajectories of the full-order model (2.1). More specifically, given r-dimensional subspaces $V = \text{Range}(\mathbf{\Phi})$ and $W = \text{Range}(\mathbf{\Psi})$, TrOOP seeks an optimal $\mathbb{P}$ by solving the following optimization problem

$$\min_{(V,W)\in\mathcal{M}_{\text{TrOOP}}} \quad J_{\text{TrOOP}} = \sum_{i=0}^{N-1} \|\mathbf{y}(t_i) - \hat{\mathbf{y}}(t_i)\|^2 \tag{2.28}$$

subject to (2.2) (or, equivalently, to (2.3)), where $\mathcal{M}_{\text{TrOOP}} = \mathcal{G}_{n,r} \times \mathcal{G}_{n,r}$ is the product of two Grassmann manifolds. While the cost function (2.28) is the same as the one in (2.6), solving the optimization problem (2.28) is intrusive because TrOOP constrains the reduced-order dynamics to be the Petrov-Galerkin projection of the full-order dynamics. Consequently, computing the gradient of the cost function J_{TrOOP} with respect to the parameters requires differentiating through the dynamics $\mathbf{f}$ in (2.1). This can be seen by deriving the gradient in a way analogous to that of Proposition 2.1, or alternatively, following Proposition 4.3 in [25]. As previously discussed, not all black box solvers allow for easy differentiation of the governing equations so, for this reason, solving the TrOOP optimization problem can be infeasible in some applications.

Operator Inference, on the other hand, is a non-intrusive model reduction framework that seeks a reduced-order model by orthogonally projecting the data onto a low-dimensional subspace and then fitting the reduced-order dynamics. This subspace is typically chosen as the span of the leading Proper Orthogonal Decomposition (POD) modes associated with some representative data set generated from (2.1). In particular, given a full-order trajectory $\mathbf{x}(t_i)$ sampled from (2.1) at times t_i, the time-derivative $\mathrm{d}\mathbf{x}(t_i)/\mathrm{d}t$, the input $\mathbf{u}(t_i)$, a r-dimensional subspace spanned by $\mathbf{\Phi} \in \mathbb{R}^{n\times r}$, and some parameterization of the reduced-order dynamics (e.g., $\mathbf{f}_r = \mathbf{A}_r\hat{\mathbf{z}} + \mathbf{H}_r : \hat{\mathbf{z}}\hat{\mathbf{z}}^\intercal + \mathbf{B}_r\mathbf{u}$), Operator Inference solves

$$\min_{(\mathbf{A}_r,\mathbf{H}_r,\mathbf{B}_r)\in\mathcal{M}_{\text{OpInf}}} \quad J_{\text{OpInf}} = \sum_{i=0}^{N-1} \left\| \frac{\mathrm{d}\hat{\mathbf{z}}(t_i)}{\mathrm{d}t} - \mathbf{A}_r\hat{\mathbf{z}}(t_i) - \mathbf{H}_r : \hat{\mathbf{z}}(t_i)\hat{\mathbf{z}}(t_i)^\intercal - \mathbf{B}_r\mathbf{u}(t_i) \right\|^2, \tag{2.29}$$

where $\hat{\mathbf{z}}(t_i) = \mathbf{\Phi}^\intercal\mathbf{x}(t_i)$ and $\mathcal{M}_{\text{OpInf}} = \mathbb{R}^{r\times r} \times \mathbb{R}^{r\times r\times r} \times \mathbb{R}^{r\times m}$. As observed in [28], equation (2.29) can be conveniently written as a linear least-squares problem whose solution is obtained via the Moore-Penrose inverse rather than via iterative gradient-based algorithms. Furthermore, given the least-squares nature of the problem, it is straightforward to add regularization (e.g., to promote stability and/or avoid overfitting) by penalizing the Frobenius norm of the parameters [21, 34]. While Operator Inference offers a convenient non-intrusive model reduction platform, it may suffer from the fact that it maps the high-dimensional data onto a low-dimensional space via orthogonal projection. We shall see that this can lead to inaccurate models if the full-order dynamics exhibit transient growth (e.g., due to non-normal mechanisms).

It is now clear that our model reduction framework merges concepts from both TrOOP and Operator Inference. Specifically: TrOOP seeks optimal projections while constraining the reduced-order dynamics to be of Petrov-Galerkin form, Operator Inference seeks optimal reduced-order dynamics while constraining the projection operator to be orthogonal and onto the span of POD modes, and our formulation simultaneously seeks optimal projections and optimal reduced-order dynamics. Moving forward, we call our formulation "Non-intrusive Trajectory-based optimization of Reduced-Order Models" (NiTROM). In closing this section, it is also worth mentioning that NiTROM solves an optimization problem similar in spirit to the one in "low-rank dynamic mode decomposition" [33], where the encoder and decoder are taken to be elements of the Grassmann manifold, and the reduced-order dynamics are assumed to be linear and discrete in time. Furthermore, by viewing the projection operator as a linear autoencoder, we can find several connections between NiTROM and existing intrusive and non-intrusive model reduction formulations that rely on (usually nonlinear) autoencoders parameterized by neural networks. Recent examples may be found in [13, 10, 24], although, to the best of our understanding, the only autoencoder architecture that defines a nonlinear projection onto a curved manifold is presented in [24].

3. Application to a toy model.

In this section, we apply NiTROM to a three-dimensional toy model, and we compare with the intrusive TrOOP and POD Galerkin formulations and the non-intrusive Operator Inference. The model is governed by the following equations

$$\dot{x}_1 = -x_1 + \nu x_1 x_3 + u \tag{3.1}$$

$$\dot{x}_2 = -2x_2 + \nu x_2 x_3 + u \tag{3.2}$$

$$\dot{x}_3 = -5x_3 + u \tag{3.3}$$

$$y = x_1 + x_2 + x_3, \tag{3.4}$$

where $\dot{x}_1 = \mathrm{d}x_1/\mathrm{d}t$ and ν is a parameter. If ν is small, then these dynamics are effectively linear and governed by a normal (in fact, diagonal) linear operator. Conversely, if ν is large, the dynamics become particularly tedious [25, 26]: not only are they more heavily nonlinear, but the nonlinearity is such that the rapidly-decaying state x_3 has a large impact on the remaining states. Systems where low-energy (or rapidly-decaying) states have a large impact on the remaining states are precisely those where ROMs obtained via orthogonal projection are more likely to give inaccurate predictions. In order to demonstrate this phenomenon, we consider two separate cases, $\nu = 5$ and $\nu = 20$, and we seek two-dimensional ROMs capable of predicting the time history of the measured output y in response to step inputs $u(t) = \gamma H(t)$, where $H(t)$ is the Heaviside step function centered at $t = 0$, and $\gamma \in (0, 1/4)$. Given the quadratic nature of the full-order dynamics, we seek quadratic ROMs of the form

$$\frac{\mathrm{d}\hat{\mathbf{z}}}{\mathrm{d}t} = \mathbf{A}_r \hat{\mathbf{z}} + \mathbf{H}_r : \hat{\mathbf{z}}\hat{\mathbf{z}}^\intercal + \mathbf{\Psi}^\intercal \mathbf{u} \tag{3.5}$$

$$\hat{y} = \mathbf{C}\mathbf{\Phi}\left(\mathbf{\Psi}^\intercal \mathbf{\Phi}\right)^{-1} \hat{\mathbf{z}}, \tag{3.6}$$

where $\mathbf{C} = [1\,1\,1]$ is a row vector and $\mathbf{u} = (u, u, u)$.

For both cases, $\nu = 5$ and $\nu = 20$, we train the models as follows. We collect $y(t)$ from $N_{\text{traj}} = 4$ step responses generated with $\gamma \in \{0.01, 0.1, 0.2, 0.248\}$ and initialized from rest. For each trajectory, we sample y at $N = 20$ equally-spaced times $t_i \in [0, 10]$. The cost function for NiTROM and TrOOP is

$$J = \sum_{j=0}^{N_{\text{traj}}-1} \frac{1}{\alpha_j} \sum_{i=0}^{N-1} \|y^{(j)}(t_i) - \hat{y}^{(j)}(t_i)\|^2, \tag{3.7}$$

with $\alpha_j = N_{\text{traj}} N \|\mathbf{C}\overline{\mathbf{x}}^{(j)}\|^2$, where $\overline{\mathbf{x}}^{(j)}$ is the exact steady state that arises in response to the step input magnitude $\gamma^{(j)}$. (The steady state is computed analytically for simplicity, but it could just as easily have been computed via time-stepping since all steady states considered herein are linearly stable.) For both methods, the optimization was performed using the conjugate gradient algorithm available in `Pymanopt` [36], with the ambient-space gradient defined following Proposition 2.1. Both methods were initialized with $\boldsymbol{\Psi} = \boldsymbol{\Phi}$ given by the leading two POD modes computed from the four training step responses. Additionally, NiTROM was provided with initial reduced-order tensors computed via Galerkin projection of the full-order dynamics onto the POD modes. The cost function for Operator Inference is

$$\begin{aligned} J_{\text{OpInf}} = &\sum_{j=0}^{N_{\text{traj}}-1} \frac{1}{\alpha_j} \sum_{i=0}^{N-1} \left\| \frac{\mathrm{d}\hat{\mathbf{z}}^{(j)}(t_i)}{\mathrm{d}t} - \mathbf{A}_r \hat{\mathbf{z}}^{(j)}(t_i) - \mathbf{H}_r : \hat{\mathbf{z}}^{(j)}(t_i) \hat{\mathbf{z}}^{(j)}(t_i)^\intercal - \boldsymbol{\Phi}^\intercal \mathbf{u}^{(j)}(t_i) \right\|^2 \\ &+ \lambda \|\text{Mat}(\mathbf{H}_r)\|_F^2, \end{aligned} \tag{3.8}$$

where $\boldsymbol{\Phi}$ are the POD modes that we just described, $\hat{\mathbf{z}} = \boldsymbol{\Phi}^\intercal \mathbf{x}$, Mat $(\mathbf{H}_r)$ denotes the matricization of the third-order tensor $\mathbf{H}_r$ and λ is the regularization parameter. In both cases ($\nu = 5$ and $\nu = 20$) $\lambda \approx 10^{-7}$, and the chosen λ is (approximately) the one that yields the best possible Operator Inference model, as measured by the cost function J in (3.7). Also, it is worth mentioning that the time-derivative of the reduced-order state $\mathrm{d}\hat{\mathbf{z}}(t_i)/\mathrm{d}t$ is computed exactly. That is, $\mathrm{d}\hat{\mathbf{z}}(t_i)/\mathrm{d}t = \boldsymbol{\Phi}^\intercal \mathbf{f}(\mathbf{x}(t_i))$, where $\mathbf{f}$ denotes the right-hand side of the full-order dynamics and $\mathbf{x}(t_i)$ is the training full-order snapshot whose POD coefficients are $\hat{\mathbf{z}}(t_i)$.

The models were tested by generating 100 step-response trajectories with γ sampled uniformly at random from the interval $(0, 1/4)$. The results are shown in figure 1a for both values of ν, where the average error over trajectories is defined as

$$e(t) = \frac{1}{N_{\text{traj}}} \sum_{j=0}^{N_{\text{traj}}-1} \frac{1}{\alpha_j} \|y^{(j)}(t) - \hat{y}^{(j)}(t)\|^2, \tag{3.9}$$

with α_j as in (3.7). Figure 1a shows that all models are very accurate when $\nu = 5$. This is expected, since we have seen that for lower values of ν (and for the moderate step input magnitudes we are considering here), the dynamics of the full-order model are effectively linear and (more importantly) governed by a normal operator. Therefore, accurate ROMs can be obtained via orthogonal projection. The accuracy of all the models can also be appreciated in figure 2a, where we see the time history of the output y in response to a sinusoidal input

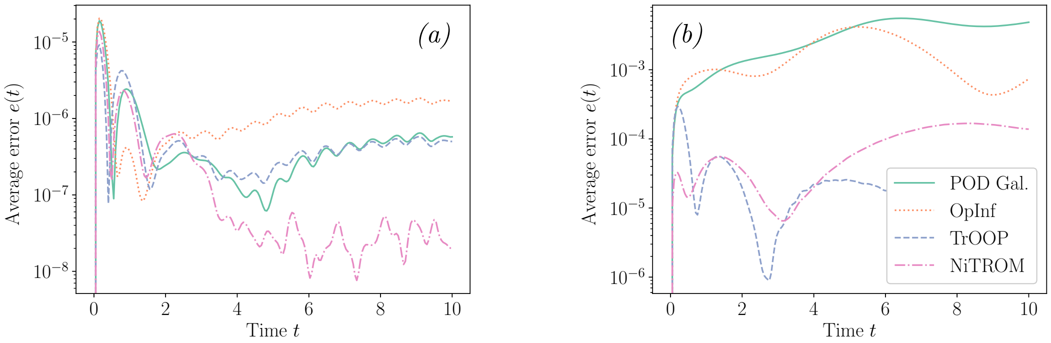

Figure 1: Toy model: (a) average testing error (3.9) for $\nu = 5$ (normal dynamics). (b) Analog for $\nu = 20$ (non-normal dynamics).

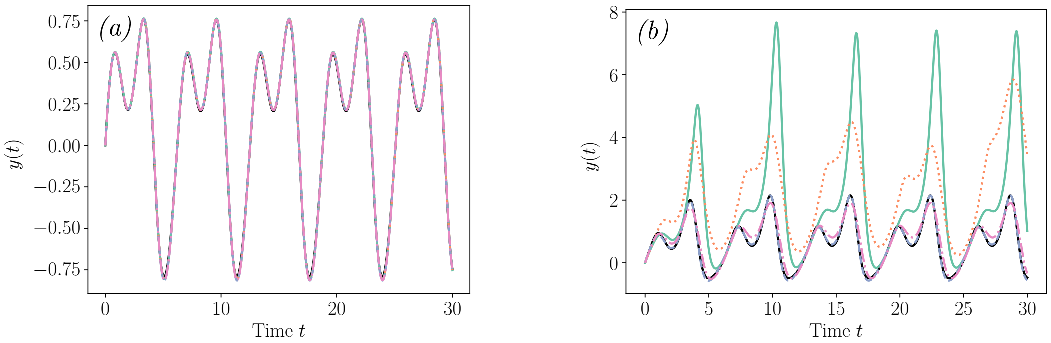

Figure 2: Toy model: time history of the output y in response to a sinusoidal input $u(t) = 0.45\,(\sin(t) + \cos(2t))$ with (a) $\nu = 5$ (normal dynamics) and (b) $\nu = 20$ (non-normal dynamics). The black continuous line is the ground-truth given by the full-order model. The rest of the legend is in figure 1.

(recall that we have not trained on sinusoids). By contrast, as we increase ν to 20, we start to observe some loss in predictive accuracy, particularly from the models (POD Galerkin and Operator Inference) that rely on orthogonal projections. This can be seen in figure 1b, and, even more convincingly, in figure 2b. In the latter, we see that while NiTROM and TrOOP provide a very good estimate of the output y in response to a sinusoidal input, the POD Galerkin and Operator Inference models struggle to do so. This must be attributed to the fact that TrOOP and NiTROM identify ROMs via oblique projection, while the other two methods use orthogonal projections.

For completeness, we also show the decay of the loss function versus conjugate gradient iterations for both TrOOP and NiTROM in figure 3. In particular, we observe that in both cases ($\nu = 5$ and $\nu = 20$), NiTROM attains a lower loss function value than TrOOP. However, in the $\nu = 20$ case, TrOOP reaches the stopping criterion $\|\nabla J\| \leq 10^{-6}$ much faster than NiTROM. Presumably, this is due to the fact that NiTROM's optimization landscape is "less

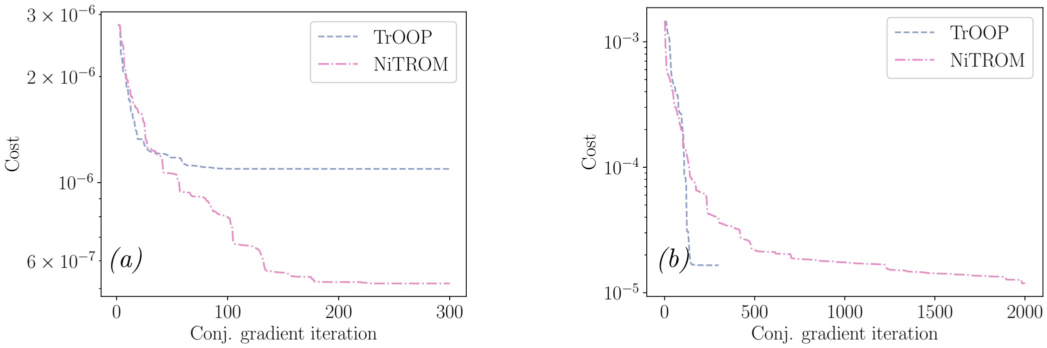

Figure 3: Toy model: cost function value (3.7) versus conjugate gradient iteration for (a) $\nu = 5$ and (b) $\nu = 20$.

friendly" than TrOOP's, as NiTROM admits a larger class of solutions. In fact, while the larger number of parameters in NiTROM allows for a wider class of reduced-order models, it may also make it more difficult for the optimizer to find a "good" local minimum.

4. Application to the complex Ginzburg-Landau (CGL) equation.

In this section we consider the complex Ginzburg-Landau (CGL) equation

$$\frac{\partial q}{\partial t} = \left(-\nu\frac{\partial}{\partial x} + \gamma\frac{\partial^2}{\partial x^2} + \mu\right) q - a|q|^2 q, \quad x \in (-\infty,\infty)\,,\ q(x,t) \in \mathbb{C}, \tag{4.1}$$

with parameters $a = 0.1$, $\gamma = 1-i$, $\nu = 2+0.4i$ and $\mu = \left(\mu_0 - 0.2^2\right) + \mu_2 x^2/2$ with $\mu_2 = -0.01$ and $\mu_0 = 0.38$. Here, $i = \sqrt{-1}$. For this choice of parameters, the origin $q(x,t) = 0$ is linearly stable, but exhibits significant transient growth due to the non-normal nature of the linear dynamics [17]. This type of behavior is common in high-shear flows (e.g., boundary layers, mixing layers and jets), making the CGL a meaningful and widely-used benchmark example. In this section, we are interested in computing ROMs capable of predicting the input-output dynamics of (4.1) in response to spatially-localized inputs. In particular, we wish to predict the time history of complex-valued measurements

$$y = Cq = \exp\left\{-\left(\frac{x+\overline{x}}{s}\right)^2\right\} q \tag{4.2}$$

in response to complex-valued inputs u that enter the dynamics according to

$$Bu = \exp\left\{-\left(\frac{x-\overline{x}}{s}\right)^2\right\} u. \tag{4.3}$$

Here, $s = 1.6$ and $\overline{x} = -\sqrt{-2\left(\mu_0 - 0.2^2\right)/\mu_2}$ is the location of the so-called "branch I" of the disturbance-amplification region (see [17] for additional details). Upon spatial discretization on a grid with n nodes, equation (4.1) can be written as a real-valued dynamical system with cubic dynamics

$$
\begin{aligned}
\frac{\mathrm{d}\mathbf{q}}{\mathrm{d}t} &= \mathbf{A}\mathbf{q} + \mathbf{H} : (\mathbf{q} \otimes \mathbf{q} \otimes \mathbf{q}) + \mathbf{B}\mathbf{u} \\
\mathbf{y} &= \mathbf{C}\mathbf{q},
\end{aligned} \tag{4.4}
$$

where the state $\mathbf{q} \in \mathbb{R}^{2n}$ contains the spatially-discretized real and imaginary components of q, $\mathbf{u} \in \mathbb{R}^2$ contains the real and imaginary components of the input u and $\mathbf{y} \in \mathbb{R}^2$ contains the real and imaginary components of the output y. Thus, given the form of the full-order system, we seek cubic reduced-order models with dynamics expressed as the sum of a linear term, a cubic term and a linear input term.

We train our models by simulating the response of (4.4) to impulses

$$
\mathbf{B}\mathbf{u}(t) = \begin{cases} \beta \mathbf{B}\mathbf{e}_j & \text{if } t = 0 \\ 0 & \text{if } t \neq 0 \end{cases} \tag{4.5}
$$

where $\mathbf{e}_j \in \mathbb{R}^2$ is the unit-norm vector in the standard basis and $\beta \in \{-1.0, 0.01, 0.1, 1.0\}$. We therefore have a total of $N_{\text{traj}} = 8$ training trajectories, and we collect the output $\mathbf{y}$ at $N = 1000$ uniformly-spaced time instances $t_i \in [0, 1000]$. Since the leading five POD modes associated with the training data contain approximately 98% of the variance and are sufficient to reconstruct the time history of the output $\mathbf{y}$ almost perfectly, we seek models of size $r = 5$. The cost functions for NiTROM, TrOOP and Operator Inference are analogous to those considered in section 3, except that the reduced-order dynamics are cubic and the normalization constants α_j in (3.7) are defined as the time-averaged energy of the output $\mathbf{y}$ along the jth trajectory. In Operator Inference, the regularization parameter for the reduced-order fourth-order tensor was chosen as $\lambda = 10^9$ following the same criterion described in the previous section. The NiTROM optimization was initialized with $\mathbf{\Phi} = \mathbf{\Psi}$ given by the first five POD modes of the training data and the reduced-order tensors provided by Operator Inference. The optimization was conducted using *coordinate* descent by successively holding the reduced-order tensors fixed and allowing for the bases $\mathbf{\Phi}$ and $\mathbf{\Psi}$ to vary, and viceversa. On this particular example, we found this procedure to be less prone to getting stuck in "bad" local minima. TrOOP, on the other hand, was initialized with $\mathbf{\Phi}$ and $\mathbf{\Psi}$ given by Balanced Truncation [22, 30] since the initialization with POD modes led to a rather inaccurate local minimum. TrOOP's optimization was carried out using conjugate gradient.

We test the performance of our model by generating 50 trajectories in response to inputs of the form (4.5) with β drawn uniformly at random from $[-1.0, 1.0]$. The average error across all testing trajectories is shown in figure 4a, while a representative impulse response is shown in figure 4b. Overall, we see that both NiTROM and TrOOP achieve very good predictive accuracy and are capable of tracking the output through the heavy oscillatory transients. By contrast, Operator Inference and the POD-Galerkin model exhibit higher errors, and this is most likely due to the highly non-normal nature of the CGL dynamics. In fact, both these methods achieve dimensionality reduction by orthogonally projecting the state onto the span of POD modes, while, as previously discussed, reduced-order models for non-normal systems typically require carefully chosen oblique projections. Finally, we demonstrate the predictive accuracy of NiTROM on unseen sinusoidal inputs of the form $\mathbf{B}\mathbf{u}(t) = 0.05 \sin(k\omega t)\mathbf{B}\mathbf{v}/\|\mathbf{B}\mathbf{v}\|$,

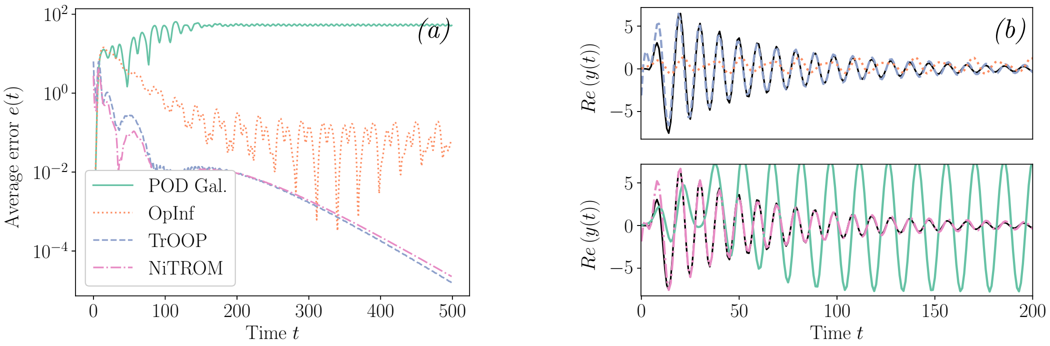

Figure 4: CGL: (a) average testing error (analogous to (3.9)). (b) Real part of the output y from a representative testing impulse response. The black line in panel (b) denotes the ground-truth response.

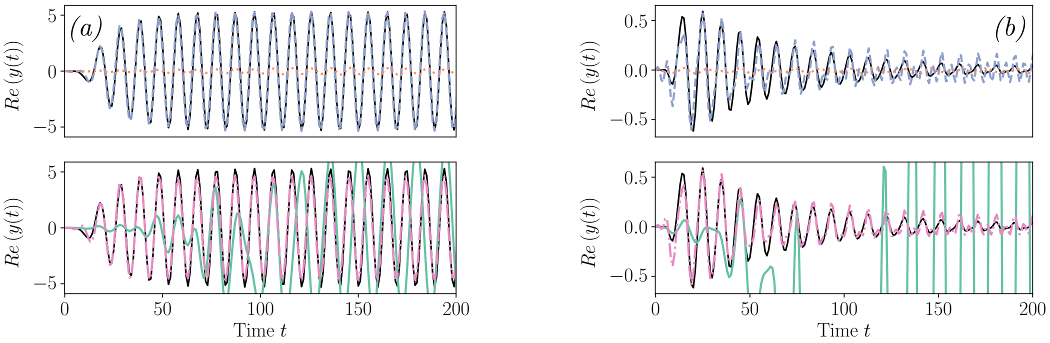

Figure 5: CGL: Real part of the output y in response to a sinusoidal input with frequencies (a) ω and (b) 2ω, where $\omega \approx 0.648$ is the fundamental frequency of the system. The black continuous line indicates the ground truth, and the rest of the legend is in figure 4a.

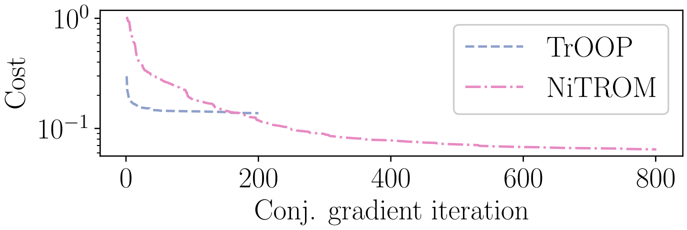

Figure 6: CGL: Cost function value versus conjugate gradient iteration for the CGL equation. TrOOP was initialized using Balanced Truncation, while NiTROM using Operator Inference.

where $\mathbf{v} \in \mathbb{R}^2$ is chosen at random and $\omega \approx 0.648$ is the natural frequency of the system. The results for frequencies ω and 2ω are shown in figure 5, where we see that NiTROM provides an accurate estimate of the response of the system at frequency ω and an acceptable prediction at frequency 2ω. The reason why the prediction at 2ω for both TrOOP and NiTROM is not as clean as the prediction at ω is because the training data exhibited dominant oscillatory dynamics at the natural frequency ω and very little contributions from other frequencies. Nonetheless, the predictions at 2ω are better than those provided by POD-Galerkin and Operator Inference. Before closing this example, we report on the loss function value for both TrOOP and NiTROM in figure 6, but we remark that TrOOP was initialized using Balanced Truncation, while NiTROM was initialized using Operator Inference.

5. Application to the lid-driven cavity flow.

In this section, we apply our model reduction procedure to an incompressible fluid flow inside a lid-driven square cavity. The flow dynamics are governed by the incompressible Navier-Stokes equation and by the continuity equation

$$\frac{\partial \mathbf{v}}{\partial t} + \mathbf{v} \cdot \nabla \mathbf{v} = -\nabla p + Re^{-1}\nabla^2 \mathbf{v} \tag{5.1}$$

$$\nabla \cdot \mathbf{v} = 0, \tag{5.2}$$

where $\mathbf{v}(\mathbf{x}, t) = (u(\mathbf{x}, t), v(\mathbf{x}, t))$ is the two-dimensional velocity vector, $p(\mathbf{x}, t)$ is the pressure and Re is the Reynolds number. Throughout, we consider a two-dimensional spatial domain $D = [0, 1] \times [0, 1]$ with zero-velocity boundary conditions at all walls, except for $u = 1$ at the top wall. The Reynolds number is held at $Re = 8300$, where the flow admits a linearly stable steady state (shown in figure 7a), but exhibits large amplification and significant transient growth due to the non-normal nature of the underlying linear dynamics. The high degree of non-normality and consequent transient growth can be appreciated by looking at figure 7b, where we show the time history of the energy of several impulse responses. In particular, we see that after an initial decay, the energy spikes around $t = 5$ before decaying back to zero. We discretize the governing equations using a second-order finite-volume scheme on a uniform fully-staggered grid of size $N_x \times N_y = 100 \times 100$. With this spatial discretization, no pressure boundary conditions need to be imposed. The temporal integration is carried out using the second-order fractional step (projection) method introduced in [9]. Our solver was validated by reproducing some of the results in [16].

In this example, we are interested in computing data-driven reduced-order models capable of predicting the evolution of the flow in response to spatially-localized inputs that enter the x-momentum equation as

$$B(x, y)w(t) = \exp\left\{ - 5000 \left((x - x_c)^2 + (y - y_c)^2 \right) \right\} w(t), \tag{5.3}$$

with $x_c = y_c = 0.95$. Upon spatial discretization and removal of the pressure via projection onto the space of divergence-free vector fields, the dynamics are governed by

$$\frac{\mathrm{d}}{\mathrm{d}t}\mathbf{q} = \mathbf{A}\mathbf{q} + \mathbf{H} : \mathbf{q}\mathbf{q}^\intercal + \mathbf{B}w, \tag{5.4}$$

where $\mathbf{q} \in \mathbb{R}^N$ is the spatially-discretized divergence-free velocity field (with $N = 2N_xN_y = 2 \times 10^4$), $\mathbf{A}$ governs the linear dynamics, $\mathbf{H}$ is a third-order tensor representative of the

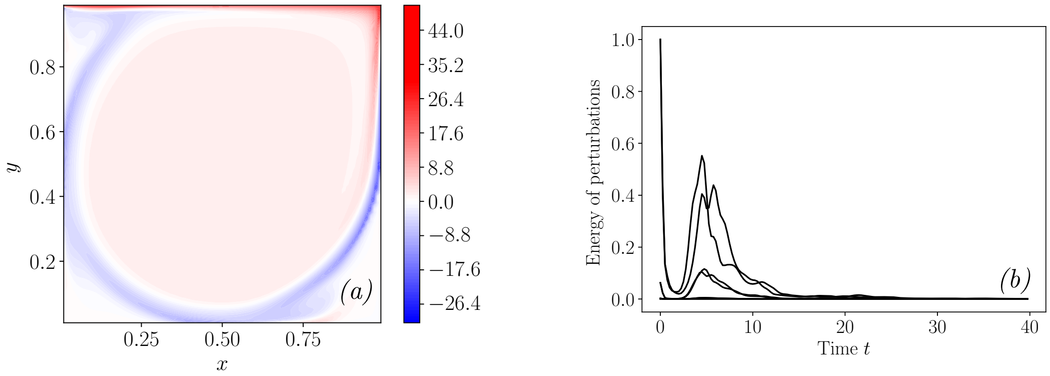

Figure 7: Cavity flow: panel (a) shows the vorticity field from the steady-state solution that exists at $Re = 8300$, and panel (b) shows the energy (i.e., the squared two norm) of the seven training trajectories.

quadratic nonlinearity in the Navier-Stokes equation and $\mathbf{B}$ is the input matrix obtained from (5.3) after enforcing that $\mathbf{B}$ generates a divergence-free vector. (For convenience, we also scale $\mathbf{B}$ to unit norm.) Throughout the remainder of this section, we take $\mathbf{y} = \mathbf{q}$ (i.e., we observe the time evolution of the whole state).

5.1. Training procedure.

We collect seven training trajectories by simulating (5.4) in response to impulses

$$w(t) = \begin{cases} \beta & \text{if } t = 0 \\ 0 & \text{if } t \neq 0, \end{cases} \tag{5.5}$$

with $\beta \in \{-1.0, -0.25, -0.05, 0.01, 0.05, 0.25, 1.0\}$. The time history of the energy of the training trajectories is shown in figure 7b. We save 160 snapshots from each trajectory at equally-distributed temporal instances $t \in [0, 40]$, and then we perform POD. Using the first 50 POD modes, which contain 99.6% of the variance in the training data, we compute an Operator Inference model by minimizing the cost function (2.29). We normalize the trajectories by their time-averaged energy and, as in the previous sections, we also penalize the Frobenius norm of the third-order tensor $\mathbf{H}$ with the regularization parameter taken to be $\lambda = 10^{-3}$.

Given the complexity of the problem and the length of the trajectories, we train NiTROM as follows. First, we pre-project the data onto the span of the first 200 POD modes, which contain $> 99.99\%$ of the variance. This guarantees that the optimal NiTROM bases $\mathbf{\Phi}$ and $\mathbf{\Psi}$ satisfy the divergence-free constraint in (5.2), since the POD modes are computed from divergence-free snapshots. Second, after initializing the search with the Operator Inference model, we train by progressively extending the length of the forecasting horizon. That is, we first optimize a model to make predictions up to $t = 2.5$, then $t = 5$, and so forth all the way up to $t = 40$.

Since, after a first pass, our model exhibited slightly unstable linear dynamics (possibly due to the presence of numerical noise and/or weak decaying oscillations in the tail end of the

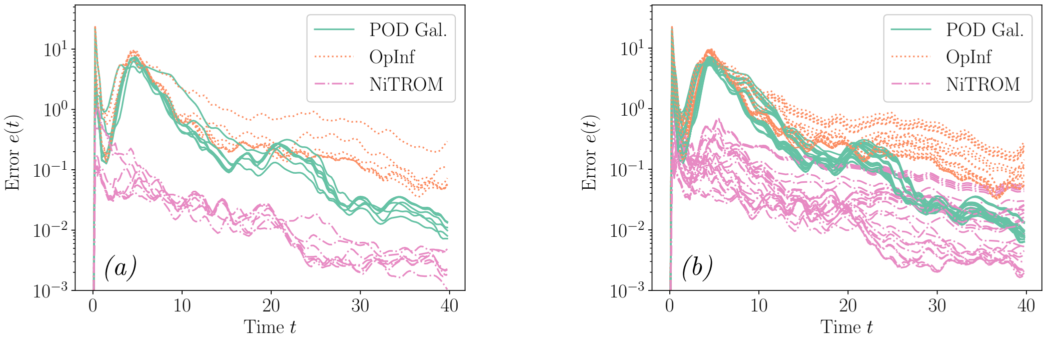

Figure 8: Cavity flow: panel (a) shows the training error from the 7 training impulses responses, and panel (b) shows the testing error computed for 25 unseen impulse responses. The error is defined in equation (5.8).

training data), we added a stability-promoting penalty to our cost function as follows,

$$\widetilde{J} = J_{\text{NiTROM}} + \mu \|\hat{\mathbf{z}}_{\text{lin}}(t_f)\|^2. \tag{5.6}$$

Here, t_f is a sufficiently large time (chosen to be 100 in our case) and $\hat{\mathbf{z}}_{\text{lin}}$ satisfies

$$\frac{\mathrm{d}\hat{\mathbf{z}}_{\text{lin}}}{\mathrm{d}t} = \mathbf{A}_r \hat{\mathbf{z}}_{\text{lin}}, \quad \hat{\mathbf{z}}_{\text{lin}}(0) = \hat{\mathbf{z}}_{\text{lin},0}, \tag{5.7}$$

with $\hat{\mathbf{z}}_{\text{lin},0}$ a unit-norm random vector. Notice that this penalty is truly stability-promoting, as it is analogous to penalizing the Frobenius norm of $e^{\mathbf{A}_r t_f}$, and shrinking the Frobenius norm of the exponential map corresponds to pushing the eigenvalues of $\mathbf{A}_r$ farther into the left-half plane. The gradient of the penalty term with respect to $\mathbf{A}_r$ can be computed straightforwardly following the same logic used in Proposition 2.1. The regularization parameter μ was held at zero for most of the training, until we reached a forecasting horizon $t = 40$ when we set $\mu = 10^{-3}$. The training was conducted using coordinate descent as described in section 4, and we stopped the optimization after approximately 2000 iterations.

5.2. Testing.

In this section we compare NiTROM against Operator Inference and POD Galerkin. We do not compare against TrOOP because of its intrusive need to access the linearized dynamics and the adjoint, and because we are ultimately interested in comparing our formulation against other non-intrusive (or weakly intrusive) model reduction techniques. We test the models by generating 25 impulse responses with the impulse magnitude β drawn uniformly at random from $[-1, 1]$. The training and testing errors for NiTROM, Operator Inference and for the POD-Galerkin model (all with dimension $r = 50$) are shown in figure 8. The error is defined as

$$e(t) = \frac{N}{\sum_{i=0}^{N-1} \|\mathbf{q}(t)\|^2} \|\mathbf{q}(t) - \hat{\mathbf{q}}(t)\|^2, \tag{5.8}$$

where $\mathbf{q}$ is the ground-truth and $\hat{\mathbf{q}}$ is the prediction given by the reduced-order model. From the figure, we see that NiTROM maintains a low error across all trajectories and for all times.

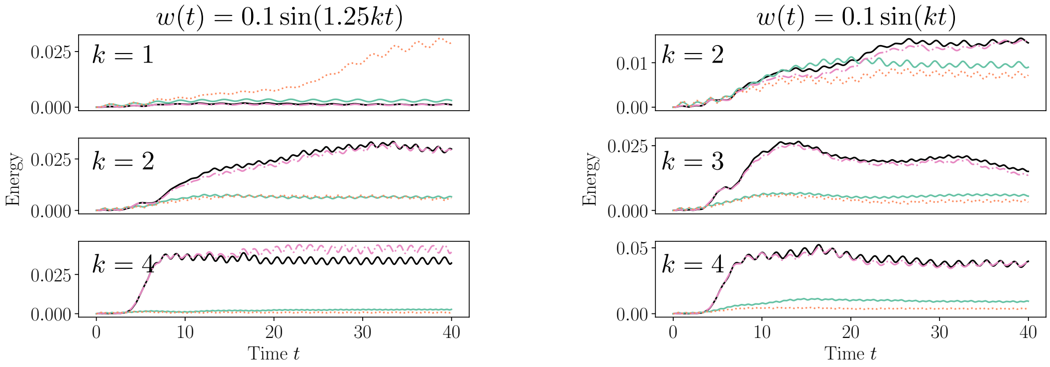

Figure 9: Cavity flow: evolution of the energy of the perturbations in response to sinusoidal inputs $w(t)$. The black line is the full-order model and the rest of the legend is in figure 8.

In particular, we observe that around $t = 5$ (when the fluid exhibits its peak in transient growth, as illustrated in figure 7b) the errors produced by POD Galerkin and Operator Inference can be one to two orders of magnitude larger than those produced by NiTROM.

As in the previous section, we also test the ability of our reduced-order model to predict the response of the fluid to sinusoidal inputs $w(t) = 0.1\sin(k\omega t)$ starting from the stable steady state. The results are shown in figure 9, where we see the response to harmonics of $\omega = 1.25$ and $\omega = 1$, which are frequencies that are naturally excited by the linear dynamics of the flow. In all cases, NiTROM exhibits better predictive accuracy than the other models, and it is capable of tracking the early-stage sharp growth of the perturbations as well as the cavity's long-time oscillatory behavior. Finally, in order to gain further insight into the performance of these models, we show vorticity snapshots at time $t = 35$ from two of the trajectories with frequency. In figure 10, where the forcing frequency was 4.00, Operator Inference and POD Galerkin underestimate the magnitude of the vorticity and they predict the wrong phase of the vortical structures (observe the vorticity field near the bottom wall at $x = 0.5$). In figure 11, where the forcing frequency is 1.25, on the other hand, POD Galerkin provides a reasonable approximation of the vortical structures despite slightly overestimating the vorticity magnitude, while the Operator Inference estimate is overall quite far from the ground truth. By contrast, NiTROM provides an accurate estimate of the vorticity phase and magnitude in both cases.

6. Conclusion.

In this paper, we have introduced a novel non-intrusive data-driven framework to compute accurate reduced-order models of high-dimensional systems that exhibit large-amplitude transient growth. These systems are ubiquitous in fluid mechanics, and they are known to pose challenges to model reduction methods that achieve dimensionality reduction via orthogonal projection onto a low-dimensional subspace (or, more generally, onto a low-dimensional nonlinear manifold). While these challenges can be addressed by intrusive methods that leverage the underlying form of the governing equations to compute an appropriate oblique projection, purely data-driven frameworks tend to achieve dimensionality reduction via orthogonal projection and this can lead to models with poor predictive accuracy.

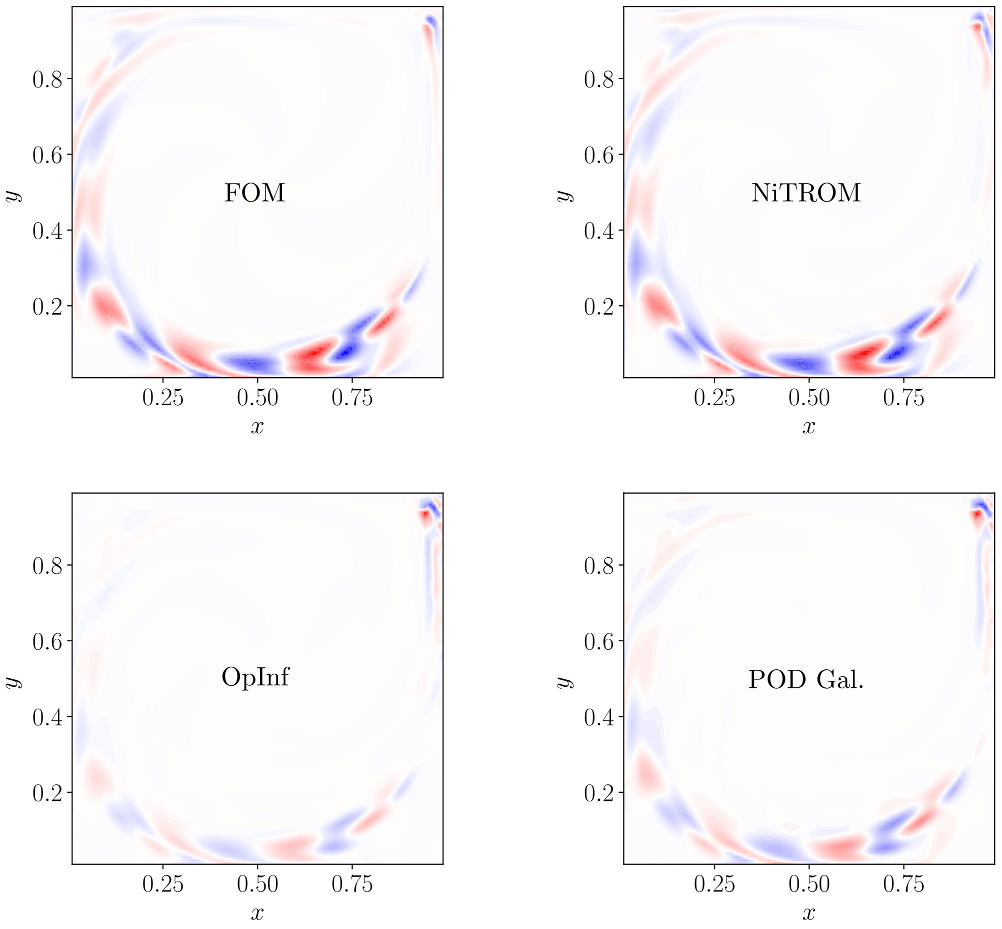

Figure 10: Cavity flow: vorticity field at time $t = 35$ from the trajectory with forcing frequency 4.00 in figure 9. Red indicates positive vorticy with maximum value 0.73, blue indicates negative vorticity with minimum value -0.73 and white is zero vorticity.

Given trajectories from the full-order system, we address this issue by solving an optimization problem to simultaneously find optimal oblique projection operators and reduced-order dynamics on their range. The framework is termed NiTROM—"Non-intrusive Trajectory-based optimization of Reduced-Order Models"—and it is demonstrated on three examples: a simple toy model governed by three ordinary differential equations, the complex Ginzburg-Landau equations and a two-dimensional incompressible lid-driven cavity flow at Reynolds number $Re = 8300$. In all these examples, NiTROM outperforms state-of-the-art non-intrusive and weakly-intrusive methods that rely on orthogonal projections for dimension reduction, and, in the first two examples it exhibits similar performance to optimal (intrusive) Petrov-Galerkin reduced-order models obtained using the recently-introduced TrOOP formulation [25]. Currently, NiTROM is formulated as a linear projection model reduction method, but, in the

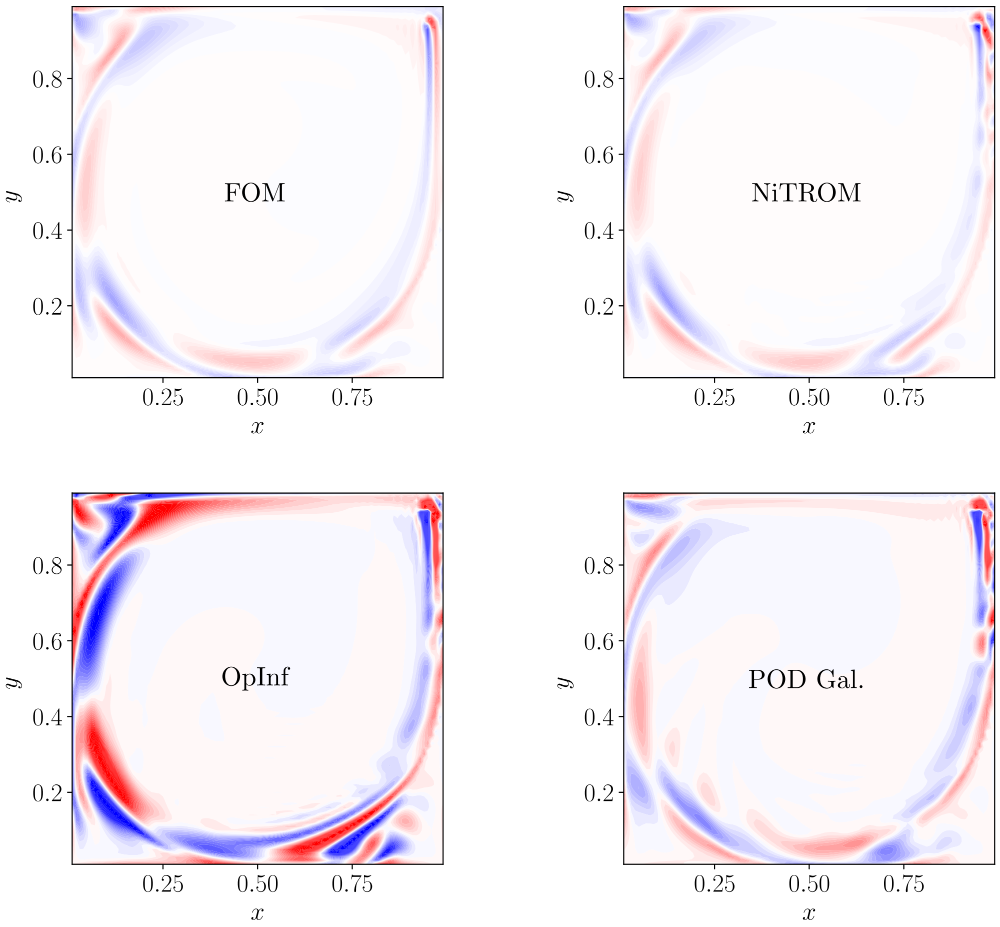

Figure 11: Cavity flow: vorticity field at time $t = 35$ from the trajectory with forcing frequency 1.25 in figure 9. Red indicates positive vorticity with maximum value 0.18, blue indicates negative vorticity with minimum value -0.18 and white is zero vorticity.

future, it would be interesting to explore the possibility of extending it to quadratic (and, more generally, polynomial) manifolds, as done within the Operator Inference formulation in [14].

Acknowledgments. This material is based upon work supported by the National Science Foundation under Grant No. 2139536, issued to the University of Illinois at Urbana-Champaign by the Texas Advanced Computing Center under subaward UTAUS-SUB00000545 with Dr. Daniel Stanzione as the PI.

REFERENCES

[1] P.-A. Absil, R. Mahony, and R. Sepulchre, *Optimization algorithms on matrix manifolds*, Princeton University Press, 2009.

[2] M. F. Barone, I. Kalashnikova, D. J. Segalman, and H. K. Thornquist, *Stable galerkin reduced order models for linearized compressible flow*, Journal of Computational Physics, 228 (2009), pp. 1932–1946.

[3] P. Benner and T. Breiten, *Interpolation-based h_2-model reduction of bilinear control systems*, SIAM Journal on Matrix Analysis and Applications, 33 (2012), pp. 859–885.

[4] P. Benner and P. Goyal, *Balanced truncation model order reduction for quadratic-bilinear control systems*, arXiv preprint arXiv:1705.00160, (2017).

[5] P. Benner, P. Goyal, and S. Gugercin, *$\mathcal{H}_2$-quasi-optimal model order reduction for quadratic-bilinear control systems*, SIAM Journal on Matrix Analysis and Applications, 39 (2018), pp. 983–1032.

[6] P. Benner, P. Goyal, J. Heiland, and I. Pontes Duff, *A quadratic decoder approach to nonintrusive reduced-order modeling of nonlinear dynamical systems*, PAMM, 23 (2023), p. e202200049.

[7] N. Boumal, B. Mishra, P.-A. Absil, and R. Sepulchre, *Manopt, a matlab toolbox for optimization on manifolds*, Journal of Machine Learning Research, 15 (2014), pp. 1455–1459.

[8] J.-M. Chomaz, *Global instabilities in spatially-developing flows: Non-normality and nonlinearity*, Annual Review of Fluid Mechanics, 37 (2005), pp. 357–392.

[9] A. J. Chorin, *Numerical solution of the Navier-Stokes equations*, Mathematics of Computation, 22 (1968), pp. 745–762.

[10] P. Conti, G. Gobat, S. Fresca, A. Manzoni, and A. Frangi, *Reduced order modeling of parametrized systems through autoencoders and sindy approach: continuation of periodic solutions*, Computer Methods in Applied Mechanics and Engineering, 411 (2023), p. 116072.

[11] G. E. Dullerud and F. Paganini, *A Course in Robust Control Theory: a Convex Approach*, Springer New York, 2000.

[12] G. Flagg, C. A. Beattie, and S. Gugercin, *Interpolatory $\mathcal{H}_\infty$ model reduction*, Systems & Control Letters, 62 (2013).

[13] S. Fresca, L. Dede', and A. Manzoni, *A comprehensive deep learning-based approach to reduced order modeling of nonlinear time-dependent parametrized PDEs*, Journal of Scientific Computing, 87 (2021), p. 61.

[14] R. Geelen, S. Wright, and K. Willcox, *Operator inference for non-intrusive model reduction with quadratic manifolds*, Computer Methods in Applied Mechanics and Engineering, 403 (2023), p. 115717.

[15] Y. Geng, J. Singh, L. Ju, B. Kramer, and Z. Wang, *Gradient preserving operator inference: Data-driven reduced-order models for equations with gradient structure*, Computer Methods in Applied Mechanics and Engineering, 427 (2024), p. 117033.

[16] U. Ghia, K. Ghia, and C. Shin, *High-Re solutions for incompressible flow using the Navier-Stokes equations and a multigrid method*, Journal of Computational Physics, 48 (1982), pp. 387–411.

[17] M. Ilak, S. Bagheri, L. Brandt, C. W. Rowley, and D. S. Henningson, *Model reduction of the nonlinear complex Ginzburg–Landau equation*, SIAM Journal on Applied Dynamical Systems, 9 (2010), pp. 1284–1302.

[18] O. Issan and B. Kramer, *Predicting solar wind streams from the inner-heliosphere to earth via shifted operator inference*, Journal of Computational Physics, 473 (2023).

[19] B. Kramer, B. Peherstorfer, and K. E. Willcox, *Learning nonlinear reduced models from data with operator inference*, Annual Review of Fluid Mechanics, 56 (2024).

[20] Z. Ma, C. W. Rowley, and G. Tadmor, *Snapshot-based balanced truncation for linear time-periodic systems*, IEEE Transactions on Automatic Control, 55 (2010), pp. 469–473.

[21] S. A. McQuarrie, C. Huang, and K. E. Willcox, *Data-driven reduced-order models via regularised operator inference for a single-injector combustion process*, Journal of the Royal Society of New Zealand, 51 (2021), pp. 194–211.

[22] B. Moore, *Principal component analysis in linear systems: Controllability, observability, and model reduction*, IEEE Transactions on Automatic Control, 26 (1981), pp. 17–32.

[23] B. R. Noack, K. Afanasiev, M. Morzyński, G. Tadmor, and F. Thiele, *A hierarchy of low-dimensional models for the transient and post-transient cylinder wake*, Journal of Fluid Mechanics, 497 (2003), pp. 335–363.

[24] S. E. Otto, G. R. Macchio, and C. W. Rowley, *Learning nonlinear projections for reduced-order modeling of dynamical systems using constrained autoencoders*, Chaos: An Interdisciplinary Journal of Nonlinear Science, 33 (2023), p. 113130.

[25] S. E. Otto, A. Padovan, and C. W. Rowley, *Optimizing oblique projections for nonlinear systems using trajectories*, SIAM Journal on Scientific Computing, 44 (2022), pp. A1681–A1702.

[26] S. E. Otto, A. Padovan, and C. W. Rowley, *Model reduction for nonlinear systems by balanced truncation of state and gradient covariance*, SIAM Journal on Scientific Computing, 45 (2023), pp. A2325–A2355.

[27] A. Padovan and C. W. Rowley, *Continuous-time balanced truncation for time-periodic fluid flows using frequential Gramians*, Journal of Computational Physics, 496 (2024), p. 112597.

[28] B. Peherstorfer and K. Willcox, *Data-driven operator inference for nonintrusive projection-based model reduction*, Computer Methods in Applied Mechanics and Engineering, 306 (2016), pp. 196–215.

[29] E. Qian, I.-G. Farcaş, and K. Willcox, *Reduced operator inference for nonlinear partial differential equations*, SIAM Journal on Scientific Computing, 44 (2022), pp. A1934–A1959.

[30] C. W. Rowley, *Model reduction for fluids using balanced proper orthogonal decomposition*, Int. J. of Bifurcation and Chaos, 15 (2005), pp. 997–1013.

[31] C. W. Rowley, T. Colonius, and R. M. Murray, *Model reduction for compressible flows using POD and Galerkin projection*, Physica D: Nonlinear Phenomena, 189 (2004), pp. 115–129.

[32] C. W. Rowley and S. T. M. Dawson, *Model reduction for flow analysis and control*, Annual Review of Fluid Mechanics, 49 (2017), pp. 387–417.

[33] P. Sashittal and D. J. Bodony, *Reduced-order control using low-rank dynamic mode decomposition*, Theoretical and Computational Fluid Dynamics, 33 (2019), p. 603–623.

[34] N. Sawant, B. Kramer, and B. Peherstorfer, *Physics-informed regularization and structure preservation for learning stable reduced models from data with operator inference*, Computer Methods in Applied Mechanics and Engineering, 404 (2023), p. 115836.

[35] H. Sharma, Z. Wang, and B. Kramer, *Hamiltonian operator inference: Physics-preserving learning of reduced-order models for canonical hamiltonian systems*, Physica D: Nonlinear Phenomena, 431 (2022), p. 133122.

[36] J. Townsend, N. Koep, and S. Weichwald, *Pymanopt: A Python toolbox for optimization on manifolds using automatic differentiation*, Journal of Machine Learning Research, 17 (2016), p. 1–5.

[37] P. Van Dooren, K. Gallivan, and P.-A. Absil, *$\mathcal{H}_2$-optimal model reduction of mimo systems*, Applied Mathematics Letters, 21 (2008), pp. 1267–1273.

[38] A. Varga, *Balanced truncation model reduction of periodic systems*, in Proceedings of the 39th IEEE Conference on Decision and Control, vol. 3, 2000, pp. 2379–2384.

[39] K. Willcox and J. Peraire, *Balanced model reduction via the proper orthogonal decomposition*, AIAA Journal, 40 (2002), pp. 2323–2330.