Journal of Computational Physics 506 (2024) 112910

Contents lists available at ScienceDirect

Journal of Computational Physics

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

Nonlinear dimensionality reduction then and now: AIMs for dissipative PDEs in the ML era

Eleni D. Koronaki [a], Nikolaos Evangelou [b], Cristina P. Martin-Linares [c], Edriss S. Titi [d,e], Ioannis G. Kevrekidis [b,*]

[a] Faculté des Sciences, de la Technologie et de la Communication, Université de Luxembourg, Maison du Nombre, Avenue de la Fonte 6, L-4364 Esch-sur-Alzette, Luxembourg
[b] Department of Chemical and Biomolecular Engineering and Department of Applied Mathematics and Statistics, Whiting School of Engineering, Johns Hopkins University, 3400 North Charles Street, Baltimore, MD 21218, USA
[c] Department of Mechanical Engineering, Whiting School of Engineering, Johns Hopkins University, 3400 North Charles Street, Baltimore, USA
[d] Department of Mathematics, Texas A & M University, College Station, TX 77843, USA
[e] Department of Applied Mathematics and Theoretical Physics, University of Cambridge, Cambridge CB3 0WA, UK

ARTICLE INFO

Keywords:
Reduced order models
Data-driven post processing Galerkin
Diffusion maps
Autoencoders
Reduced dynamics
Data-driven latent spaces

ABSTRACT

This study presents a collection of purely data-driven workflows for constructing reduced-order models (ROMs) for distributed dynamical systems. The ROMs we focus on, are data-assisted models inspired by, and templated upon, the theory of Approximate Inertial Manifolds (AIMs); the particular motivation is the so-called post-processing Galerkin method of Garcia-Archilla, Novo and Titi. Its applicability can be extended: the need for accurate truncated Galerkin projections and for deriving closed-formed corrections can be circumvented using machine learning tools. When the right *latent variables* are not *a priori* known, we illustrate how autoencoders as well as Diffusion Maps (a manifold learning scheme) can be used to discover good sets of latent variables and test their explainability. The proposed methodology can express the ROMs in terms of (a) theoretical (Fourier coefficients), (b) linear data-driven (POD modes) and/or (c) nonlinear data-driven (Diffusion Maps) coordinates. Both Black-Box and (theoretically-informed and data-corrected) Gray-Box models are described; the necessity for the latter arises when truncated Galerkin projections are so inaccurate as to not be amenable to post-processing. We use the Chafee-Infante reaction-diffusion and the Kuramoto-Sivashinsky dissipative partial differential equations to illustrate and successfully test the overall framework.

1. Introduction

Separation of time-scales in dynamical systems is crucial toward the development of Reduced Order Models (ROMs). For a certain class of dissipative evolution equations, the long term dynamics are attracted exponentially fast to smooth invariant objects known as inertial manifolds (IMs), facilitating the construction of ROMs on those. The dynamics on the IM can be described by the Inertial Form (a finite ODE system), which accurately captures the long-term behavior of the original infinite-dimensional system [54,29,61,3]. The purpose of this paper is to (somewhat systematically) outline (and demonstrate) links between "traditional" AIM technology

* Corresponding author.
E-mail address: yannisk@jhu.edu (I.G. Kevrekidis).

https://doi.org/10.1016/j.jcp.2024.112910
Received 8 October 2023; Received in revised form 21 February 2024; Accepted 2 March 2024

Available online 8 March 2024
0021-9991/

and contemporary data-driven reduction tools, giving rise to "mathematics-assisted" algorithmic ROM workflows. Such connections had initially been experimentally attempted in the 1990s (e.g. [35,60]); they are currently experiencing a strong revival due to the explosion in machine-learning-assisted modeling [39,5,37,6,7,67].

IMs have been proven to exist for only a few systems, and even then, they have not been constructed explicitly [29]. It is, nevertheless, still possible to find approximations of either the global attractor or the IM itself, i.e. Approximate Inertial Manifolds (AIMs), or the dynamics on it, i.e. the Approximate Inertial Form (AIF), and then track the dynamics in this reduced space. The key ansatz is that the attracting dynamics in the complement space to the AIM, are quickly slaved to, and embodied in, the AIF. Along these lines, the Galerkin projection, as well as nonlinear Galerkin projections on approximate inertial manifolds are also popular choices for reduced order modeling [44,30,53,7]. AIM-based ROMs have been proposed for reaction-diffusion systems [19,2], the Kuramoto-Sivashinsky equation [19,20,29], the two-dimensional Navier-Stokes equations [59,58,27], and for the three-dimensional Navier-Stokes equations [26].

It must also be noted, that, for a dissipative PDE, in the case where a clear time-scale separation exists, we converge to the inertial manifold exponentially fast. In the absence of a large separation we still converge, exponentially fast, to a thin layer around some manifold, also called an *approximate inertial manifold*. The dynamics in this case are exponentially fast attracted to this layer (that contains the attractor) but, once the trajectory enters the layer it may *"take forever" to arrive to the attractor*.

In the late 90s the post-processing Galerkin method was proposed [22,21], initially in the context of dissipative equations. Post-processing Galerkin takes into account the observation that the error between the result of integrating a truncated Galerkin on the one hand, and the projection of the true solution in the finite-dimensional Galerkin space, on the other, is significantly smaller than the error between the truncated Galerkin and the full solution (superconvergence [23,63]). We will return to this below and illustrate it in Sec. 4 and Fig. 6. Given this observation, one uses the dynamics expressed only in terms of the leading low modes (a truncated version of the equations) to integrate. Once the time integration is finished one can *post-process* the obtained solution by approximating the high modes as a function of the solution in the leading modes. Since, in the post-processing Galerkin framework, the correction is computed only at the end of time integration, this makes it much cheaper to implement computationally than true nonlinear Galerkin [22,21]. Moreover, truncation analysis derivation of the spectral method for dissipative evolution equations, such as the Navies-Stokes equations, gives rise to the *post-processing* Galerkin as the *leading order numerical scheme*, and not the Galerkin scheme itself, as commonly believed [43].

Model identification assisted by machine-learning has emerged in the 90s and is now experiencing a rebirth as a tool to discover minimal parametrizations of an IM, which can subsequently be used to evolve the dynamical system in a reduced space [42,11,65,66,15,41]. Some efforts implemented linear methods like POD [35,31,60], to identify a suitable subspace that contains the majority of the variance of the system, and parametrizes the long term dynamics. More recently, operator inference with quadratic manifolds has been proposed for model reduction [25,64,49,47]. Nonlinear dimensionality reduction methods, such as autoencoders [34] or Diffusion Maps (DMAPs) [14] have also been used to discover latent variables of data that originally live in a high-dimensional space. Learning a dynamical system in the latent space of an autoencoder (even as a collection of local charts), or in Diffusion Maps space, also provides a systematic approach to ROM construction (e.g. [51,57,17,40,37,6]). Needless to say, nonlinear system identification assisted by machine learning remains a very active current research endeavor, encompassing a plethora of directions from symbolic methods e.g. [8], to physics-informed methods, e.g. [50], to numerics-informed methods [6].

In our view, the "1980s" IM and AIM efforts towards useful reduced order models of dissipative PDEs can be succinctly summarized as follows: Given the functional form of the PDE for which we know (or believe) an IM exists, and having an estimate of the dimensionality of said manifold:

(A) start by finding the (leading) eigenmodes, say k of them, of the (dissipative part of the) operator that "determines" (parametrizes) the IM. In that sense, the components of the solution in the remaining "higher order" eigenmodes can be expressed as functions of the components in the lower, determining, ones;

(B) guided by separation of time scales ideas, construct the AIM approximating this function, by writing the components of the higher eigenmodes as (approximate) functions of the components of the lower, determining ones. Several implementable such approximations have been proposed and analyzed: e.g. the "steady" manifold, the "Euler-Galerkin", and the Foias-Manley-Temam (FMT) manifold among others. We already have a practical result: if somebody provides *as observations* the lower mode amplitudes, we can meaningfully *and analytically* improve the full spatiotemporal solution, complementing it with the recovered higher mode components. We will return to this theme when discussing post-processing Galerkin. Let it be noted here that even though the original motivation of AIM was to find an approximation to the IM whenever the latter exists, however, this idea was generalized later and implemented by finding a manifold which approximates the global attractor as a set; observing that global attractor always exists for genuine dissipative dynamical system;

(C) beyond just correcting such observations, these functions can be used to correct approximations of the dynamics through their low-order Galerkin truncation: From an accurate, high-order Galerkin truncation, we keep only the low, "determining" Galerkin ODEs; instead of omitting the higher order terms as negligible, we now *substitute* the AIM function in the low terms. We now have the "steady", or "Euler-Galerkin" or FMT inertial forms.

This original program is complemented by the "post-processing Galerkin" protocol: Here we actually keep the low order Galerkin truncation, ignoring the contribution of the higher order, slaved modes to it, expecting/believing that, in its low-dimensional space, these few ODEs are accurate enough to approximate the projection of the exact solution on the Galerkin space. The authors of [22,21,23] took into account the observation that the total error of the solutions predicted by the truncated low-order Galerkin

is appreciably larger than the error after adding to them (in a sense, "reinjecting") the AIM-approximated higher order solution components. This reinjection is performed *after* the truncated low-order Galerkin equations have been integrated until each time instance of interest (we remind the reader that this is revisited in Sec. 4 and Fig. 6).

They named the approach "post-processing Galerkin" since it takes place after the truncated low-order Galerkin has been obtained and integrated: it is these concrete available *solutions* of the model that are being improved - not the model itself.

Explicit AIMs have been obtained in the context of spectral Galerkin approximation by writing approximations of the evolution equation of the high modes in terms of the low modes, a closure relation. In the context of spectral Galerkin approximation based on Fourier modes or eigenfunctions of the Stokes operator, one can naturally decouple the phase space into low Fourier (eigenfunctions) modes and their complement high Fourier (eigenfunctions) modes. Therefore, the above-described strategy of obtaining AIM is possible to be executed explicitly, and leads to an analytical closure. For the examples we present in this work, the spectral Galerkin approximation could indeed provide a desirable closure.

However, we would like to note that in the context of the Finite Element Galerkin method, the above decomposition to coarse spatial scales and their complement is not a straightforward task. Therefore the above strategy can not be followed to obtain an explicit (paper and pencil) closure form, that expresses the fine spatial scales of the solution in terms of the coarse finite elements spatial scales. For this case, a more general framework for implementing the Post-processing Galerkin can be used [21]. In this more general case, an explicit form of an AIM in order to implement post-processing Galerkin is not required. We briefly present this more general scheme in Sec A.1.

The innovation of this work, as it progresses beyond symbolic model (AIM) or solution (post-processing Galerkin) improvement, is the introduction of data-driven methods that allow us (given accurate simulation data or observations), to efficiently address the following issues:

(a) Estimate the AIM dimensionality in a data-driven way (either through autoencoders or through manifold learning).
(b) Learn good reduced AIFs (the "correct", nonlinear Galerkin, right hand-side of the reduced, low order, components of the PDE) in a data-driven way.
(c) Learn the AIM functions (high order mode components as a function of low order model components) in a data driven way.
(d) Given the learned AIM in (c), correct the solutions of a low-order Galerkin truncation (a "data driven" post-processing Galerkin).

Beyond the steps (b-d) above, that more or less correspond to the traditional (A-C) analytical steps, a couple of very useful data-driven "twists" are introduced.

(e) Circumvent the assumption of accuracy of the low-order (linear) Galerkin truncation; the low order AIF is learned from observations of the low-order components of accurate full PDE dynamics; and now the "post-processing" that follows can be done (1) with the same "old" analytical AIMs, or, interestingly (2) with data-driven learned AIMs from the same accurate full PDE dynamics.
(f) Gray-Box (in some sense "physics-assisted") learning: instead of a fully black-box learning of the AIF using PDE observations, we now learn *the correction* of the not-so-quantitative low-order linear Galerkin truncation. This correction can be learned as an additive (residual) term, or even as a *functional* correction - hoping for easier training, since what is learned is a perturbation of the identity [46].
(g) (*This is not so much a step in our list, as a branching towards new capabilities*). Up to now, everything *but the eigenfunctions* parametrizing the manifold was data-driven; the eigenfunctions themselves were still analytical. If we allow ourselves to find the parametrization of the manifold in a data-driven way, two individually significant new options arise:

  (i) Use linear data-driven eigenfunctions: the leading Principal Components (PODs) of the full accurate PDE simulations. Now the low-order PODs parametrize the manifold, and the higher order POD components embody the AIM. POD-Galerkin takes the place of traditional Galerkin.
  (ii) Use a *nonlinear*, data-driven AIM parametrization and either:
    - use the latent variables of an autoencoder to parametrize the AIM, learn the corresponding accurate AIF, *and* post-process it to more accurate spatiotemporal PDE solution reconstruction;
    - use the leading POD components to parametrize the AIM, learn the accurate corresponding AIF, *and* post-process it for a more accurate spatiotemporal PDE solution reconstruction. At the risk of making this list ridiculously long, we also add -and illustrate below- the possibility
    - of using spectral (Diffusion Map) data mining to parametrize the AIM along with the associated *Geometric Harmonics* for the post-processing.

A schematic overview of the different options proposed in each case, are presented in Fig. 1, with references to the subsequent sections where they are discussed in detail.

The remainder of the paper is organized as follows: After listing the illustrative examples used in this study (Sec 2), we proceed with describing the methodology (Sec. 3.) We start with briefly reviewing the "traditional" approximations of IMs and IFs (and AIMs and AIFs) (Sec. 3.1). We then discuss neural network-based alternatives to approximating IMs and IFs (Sec. 3.1.2), followed by nonlinear manifold learning methods for determining the dimensionality and parametrization of the latent space (Sec. 3.2.1 and 3.2.2). After presenting our results we conclude by pointing out that the technology can be easily "transferred" to POD parametrizations of the IM (Sec. 3.2.3).

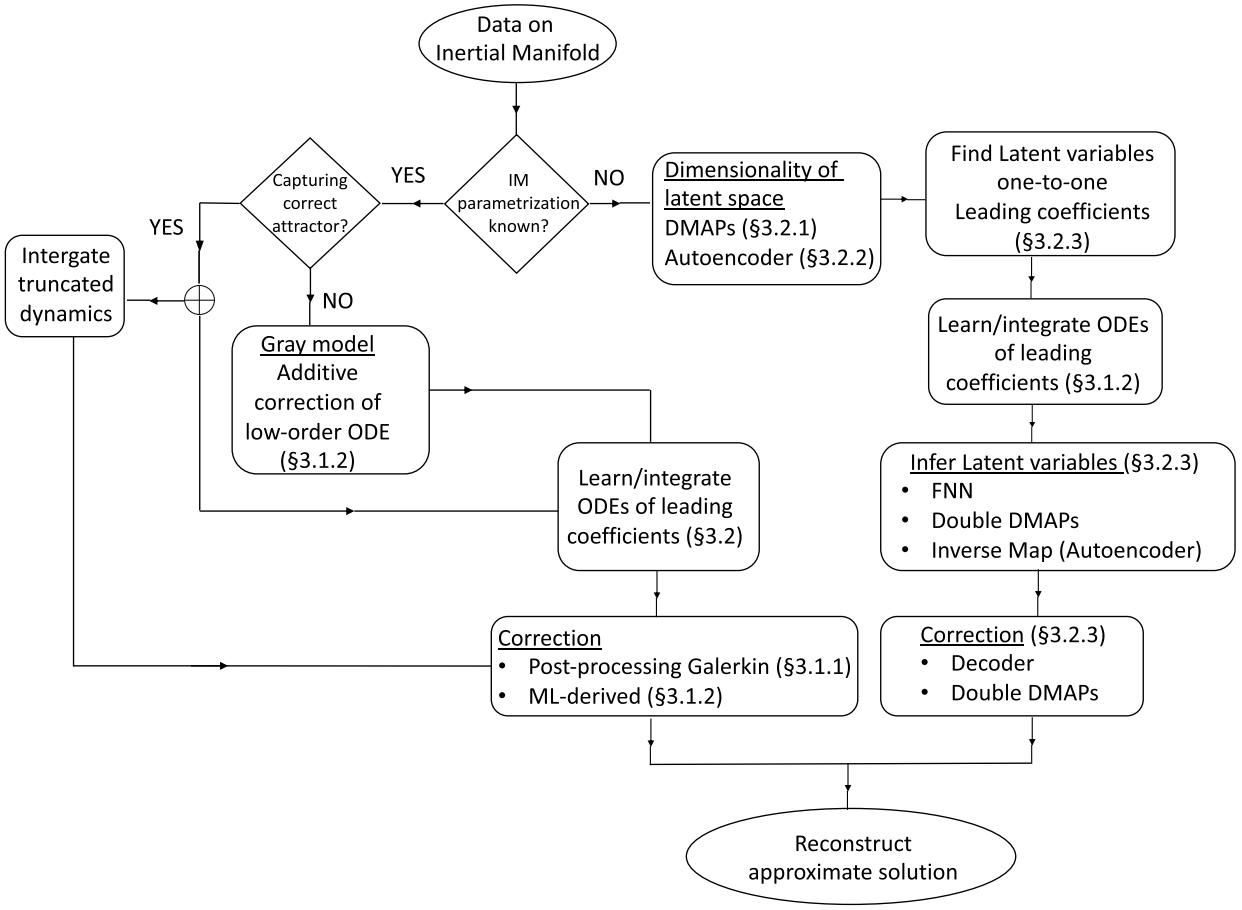

Fig. 1. Flowchart of the proposed workflow.

2. Illustrative examples: the Chafee-Infante and the Kuramoto-Sivashinsky equations

Our first example is the reaction-diffusion Chafee-Infante partial differential equation (PDE), for which the dimensionality of the Inertial Manifold (IM), for the parameter range of interest, is known; it reads:

$$u_t = u - u^3 + \nu u_{xx}. \tag{1}$$

The parameter ν was chosen as $\nu = 0.16$ and Dirichlet boundary conditions, $u(0,t) = u(\pi,t) = 0$, were used. The Chafee-Infante PDE, for $\nu = 0.16$, has been shown to have a two-dimensional inertial manifold [56,24,28,17]. To simulate the dynamics on/near this two-dimensional manifold, the Galerkin projection

$$u(x,t) \approx \sum_{k=1}^{3} \alpha_k(t) sin(kx) \tag{2}$$

was used [24,28,17]. The first two leading sine coefficients $\alpha_1(t), \alpha_2(t)$ are sufficient to parameterize this two-dimensional manifold, and Galerkin equations based only on the first two modes provide a qualitatively correct approximation of the dynamics in these two modes. We will consider the solution of the Chafee-Infante equation with three modes as the ground truth (cf Fig. 7a). For the post-processing process, the truncated equations with the first two sine coefficients $\tilde{\boldsymbol{\alpha}} = \{\alpha_1, \alpha_2\}$ are the ones used for integration up to time $t = T$. Then, their solution is post-processed to recover $\hat{\alpha} = \alpha_3$ and reconstruct the full solution $u(x,T)$. For this first example, the truncated dynamics governed by the first two sine coefficients are (considered to be) qualitatively, but not quantitatively accurate; the post-processing step aims to correct the obtained solution from these truncated dynamics.

To demonstrate the potential of the proposed methodology in a case with more complex dynamics, we select the Kuramoto-Sivashinsky (KS) PDE,

$$u_t = -\nu(uu_x + u_{xx}) - 4u_{xxxx}; \ \text{for} \ x \in [0, 2\pi]. \tag{3}$$

The KS (Equation (3)) is a prototypical equation with dynamics that include chaos, derived in the context of a diverse range of physical systems such as, but not limited to, thin film flow on inclined planes and instabilities in a laminar flame front [36,55,4,9,10,29,32]. The parameter ν in our case is set to $\nu = 33$ and periodic boundary conditions are used $u(0,t) = u(2\pi,t)$. In this example, Fourier series expansion with 8 terms is used to approximate the ground truth $u(x,t)$:

$$u(x,t) \approx \sum_{k=1}^{8} \alpha_k(t) sin(kx) + \beta_k(t) cos(kx);\ x \in [0, 2\pi], \tag{4}$$

which results in 8 ODEs for the sine coefficients ($\{\alpha_k\}_{k=1}^{8}$) and 8 for the cosine coefficients ($\{\beta_k\}_{k=1}^{8}$).

Restriction to the space of odd functions leads to retaining only the sine terms, resulting in a system of 8 ODEs for the sine coefficients which is considered, in this work, as the exact solution of the KS. We use the truncation to the leading three sine coefficients $\tilde{\boldsymbol{\alpha}} = \{\alpha_1, \alpha_2, \alpha_3\}$ to study the dynamics for $\nu = 33$; however even though it has been shown that a 3D manifold exists, the truncated equations based on the leading coefficients do not provide an accurate approximation of the dynamics of these coefficients. In this case the *traditional* post-processing Galerkin methodology, does not apply (we do not have a good base solution to correct). We circumvent this issue by constructing Gray-Box models, as we show below in Sec. 4.3.1.

3. Methodology

3.1. Approximating the IM and the IF (known latent space)

3.1.1. Euler-Galerkin

As a preamble to *traditional* post-processing Galerkin, here we discuss *nonlinear* Galerkin schemes, in particular the "Euler-Galerkin" algorithm, that provides a closed-form approximation of inertial manifolds [18]. Consider the evolution equation

$$\frac{du}{dt} + Au + F(u) = 0, \qquad u \in H \tag{5}$$

where H is an appropriate Hilbert space, A is a self-adjoint positive-definite linear operator with compact inverse, and let F be a nonlinear operator such that equation (5) is globally well-posed in time for all initial data in H. By denoting a projection onto the span of the first n eigenvectors of A by P and $Q = I - P$ we can split Equation (5) into

$$\frac{dp}{dt} + Ap + PF(p+q) = 0 \tag{6}$$

$$\frac{dq}{dt} + Aq + QF(p+q) = 0 \tag{7}$$

where $p = Pu$, $q = Qu$ and $q + p = u$. Assuming that the long-term dynamics of Equation (5) live in a $n-$dimensional inertial manifold described as the graph of a function $\Phi : PH \to QH$ we can write the projection of the inertial manifold onto PH as

$$\frac{dp}{dt} + Ap + PF(p + \Phi(p)) = 0. \tag{8}$$

An approximation of Φ is achieved through a Galerkin truncation of m modes in Equation (7), where $m > n$. The projection to the space of the higher modes $n+1, \dots, m$ defines Q_m. Since the higher modes are attracted exponentially fast to the IM and become functions of the lower modes, we perform an implicit Euler step to approximate the solution $\hat{q}$ with a step size τ. By assuming an initial condition $q_0 = 0$ we get

$$\hat{q} + \tau A\hat{q} + \tau Q_m F(p + \hat{q}) = 0. \tag{9}$$

Instead of completely solving equation (9) we perform a single fixed-point iteration using an initial $\hat{q} = 0$ and holding the lower modes $1, \dots, n$ (the components of p) constant. This gives the approximation:

$$\hat{\Phi}_m(p) = -\tau(I + \tau A)^{-1} Q_m F(p) \tag{10}$$

an algebraic expression that estimates the higher modes $\{n+1, \dots, m\}$ as a function of the lower n modes and thus an approximation of the IM itself.

Substituting $\hat{\Phi}_m(p)$ for the $m - n$ higher modes gives

$$\frac{dp}{dt} + Ap + PF(p + \hat{\Phi}_m(p)) = 0 \tag{11}$$

and more precisely an Euler-Galerkin approximation consisting of n differential equations

$$\frac{dp}{dt} + Ap + PF(p - \tau(I + \tau A)^{-1} Q_m F(p)) = 0. \tag{12}$$

In this work, the (nonlinear) Euler-Galerkin algorithm was applied to the Chafee-Infante partial differential equation, as detailed in Sec. A.2.

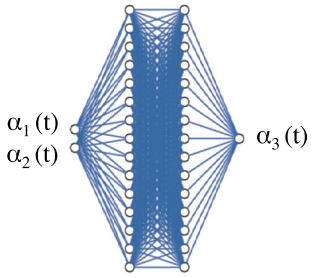

Fig. 2. Illustrative example of a feed-forward neural network for prediction of higher coefficients. In this example the lower harmonics, $\alpha_1(t)$ and $\alpha_2(t)$ are used as inputs to the network that predicts $\alpha_3(t)$.

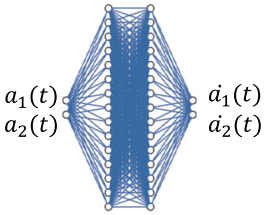

Fig. 3. An example of a feed-forward neural network architecture for the approximation of the right-hand-side of an evolution ODE.

3.1.2. Neural network derived AIM and AIF

The higher sine modes' coefficients $\hat{\boldsymbol{\alpha}}$, which are necessary for accurate reconstruction of the solution in physical space, can be obtained in a data-driven manner. Specifically, here we use deep neural networks, schematically shown in Fig. 2, to learn the coefficients $\hat{\boldsymbol{\alpha}}$, given the values of leading (lower) sine modes' coefficients at a specific point in time, $t = T$, $\tilde{\boldsymbol{\alpha}}(T)$.

$$\hat{\boldsymbol{\alpha}}(t) = f_{NN}(\tilde{\boldsymbol{\alpha}}(t)),$$

where $\tilde{\boldsymbol{\alpha}}$ stands for leading sine coefficients (low modes) and $\hat{\boldsymbol{\alpha}}$ stands for the higher sine modes coefficients. The leading coefficients $\hat{\boldsymbol{\alpha}}(T)$ have been obtained as a result of the time integration of the truncated dynamics.

Alternatively, when the result of time-integration of the two truncated lower sine coefficients equations, is inaccurate, we can correct it by learning a data-driven truncated ODE in the lower sine coefficients, with general form:

$$\frac{d\boldsymbol{\alpha}}{dt} = \boldsymbol{f}(\boldsymbol{\alpha}) \tag{13}$$

where $\boldsymbol{\alpha} \in \mathbb{R}^m$, here $m = 2$, are the variables in which we observe the evolution of the dynamics. Observe that since: m = 2 here the Poincaré-Bendixon theorem applies. Hence the dynamics of the low modes is either goes to a limit-cycle or to a steady state. This data-driven AIF was first explicitly described and implemented in Theodoropoulos et al. [60] (see also in [35]).

The function $\boldsymbol{f}$ is approximated by a fully connected neural network, schematically represented in Fig. 3. The goal is to predict the time derivatives of the lower sine coefficients from their values. Once this is done, the right-hand-side of the ODEs in Eq. (13) can be used in conjunction with any method of integration in time, such as the Runge-Kutta, to accurately approximate $\tilde{\alpha}(T)$ and then proceed as above to post-process $\tilde{\alpha}(T)$.

For parameter values of the KS equation for which the long-term truncated dynamics may not be accurate, an appealing alternative to the Black-Box approach discussed above arises.

One can remedy the situation by *first correcting* the reduced dynamics, before deriving the missing terms for reconstruction. This can be achieved by constructing a "Gray-Box" data-driven dynamic model. This Gray-Box model describes the evolution of a reduced system, by adding to the truncated dynamics a *learned* correction term, which can be thought of as a closure. This correction is approximated by a neural network that takes as inputs the lower order sine coefficients and delivers the difference between their true time-derivatives and the truncated Galerkin time-derivatives:

$$\frac{d\tilde{\boldsymbol{\alpha}}^{\mathrm{p}}}{dt} - \frac{d\tilde{\boldsymbol{\alpha}}^{\mathrm{t}}}{dt} = g_{NN}(\tilde{\boldsymbol{\alpha}}(t))$$

where $\frac{d\tilde{\boldsymbol{\alpha}}^{\mathrm{p}}}{dt}$ is the true vector field projected in the leading sine coefficients $\tilde{\boldsymbol{\alpha}}$ and $\frac{d\tilde{\boldsymbol{\alpha}}^{\mathrm{t}}}{dt}$ is the vector field of the corresponding truncated Galerkin projection.

Here, g_{NN} is approximated using a neural network implemented in tensorflow [1] with 6 hidden layers, 95 neurons each, and a *tanh* activation function. The loss function used is the mean squared error (MSE), and the Adam optimizer is employed.

Finally, it is worth noting that the proposed workflow works equally well, when considering evolution equation of the leading POD mode coefficients, as parametrizing the IM. An illustrative example, based on the Chafee-Infante POD-based equations can be found in A.2.

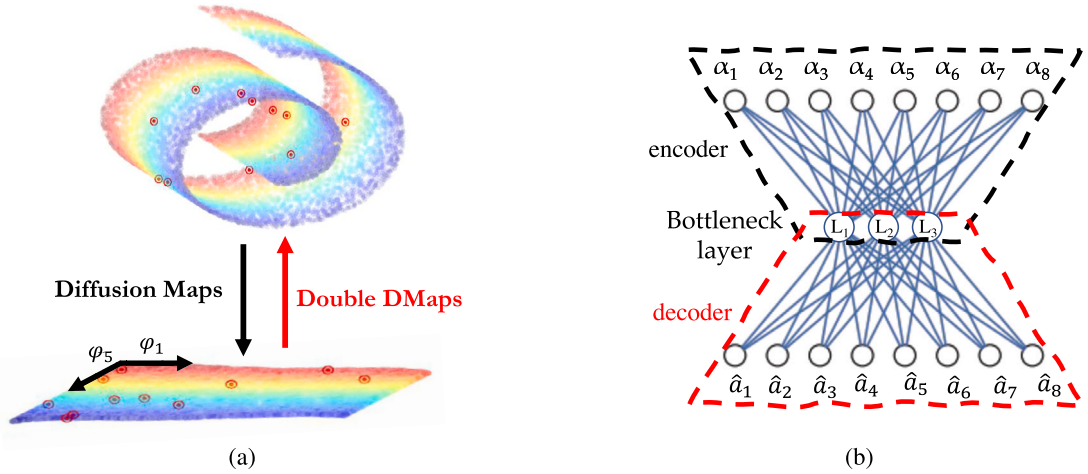

Fig. 4. Learning a low dimensional embedding of data: (a) Manifold learning with Diffusion Maps and inverse transformation with Double Diffusion Maps [17]; (b) Representative autoencoder structure, including encoder/decoder and the bottleneck layer.

3.2. Learning the dimensionality of the latent space

In most cases, a minimal parametrization of the IM of a dynamical system is not known *a priori*. It is possible to discover it, using different data mining approaches, such as Diffusion Maps and autoencoders. Both methods are discussed in the following paragraphs and summarized in Fig. 4

3.2.1. Diffusion maps

Diffusion maps [13,48,14] is a manifold learning framework that can (based upon diffusion processes) facilitate discovering low-dimensional intrinsic geometric descriptions of data sets, even when the data is high-dimensional, nonlinear and/or corrupted by (relatively small) noise. It is used here to discover the dimensionality of the IM and provide a data-driven parametrization of it.

The parametrization of the manifold is obtained through a few eigenvectors, ϕ_i, of a scaled affinity matrix, which contains the Euclidean distances between all the pairs of available data points. A detailed description of the Diffusion Maps algorithm is provided in the Sec. A.5 of the Appendix and for the Double Diffusion Maps in Sec. A.6.

3.2.2. Autoencoders

Autoencoders [34] are neural networks that are trained (a) to encode high-dimensional data into a low-dimensional representation (b) to reconstruct the original high-dimensional from this lower-dimensional representation (cf. Fig. 4b). In this context, the input layer is the same as the output, and the low-dimensional encoding is parametrized by the weights of the bottleneck layer. The loss function

$$L_\theta = ||\boldsymbol{\alpha}^{(k)} - \tilde{\boldsymbol{\alpha}}^{(k)}||^2 \tag{14}$$

is commonly used to train an autoencoder where $\boldsymbol{\alpha}^{(k)}$ represent a data point in the ambient space and $\tilde{\boldsymbol{\alpha}}^{(k)}$ the reconstructed data point k from the autoencoder.

In this work, we use autoencoders for an additional second use case, which relies on the observation that the discovered autoencoder latent coordinates are one-to-one with the leading sine coefficients $\tilde{\boldsymbol{\alpha}}$, as discussed in detail in the following paragraph.

3.2.3. Theoretical and data-driven latent variables: transformations and AIMs

The *local* one-to-one relation between the autoencoder's latent variables ($\boldsymbol{L}$) and the leading sine coefficients ($\tilde{\boldsymbol{\alpha}}$) is tested by computing the Inverse Function Theorem across the training data. The Inverse Function Theorem guarantees local invertibility in a neighborhood of any point $\boldsymbol{L}_i \in \boldsymbol{L}$ if the determinant of the Jacobian ($\det(\mathbf{J}_f(\mathbf{L}))$ is bounded away from zero. We provide a more detailed description of the Inverse Function Theorem in Sec. A.7 of the Appendix. The Jacobian computation in our case is performed by using automatic differentiation with Tensorflow.

In this case, a more global one-to-one correspondence (on the vicinity of our data) also exists. This is tested by constructing regression schemes between the leading sine coefficients and the latent coordinates of the autoencoder. As shown in Fig. 5 upon training of the autoencoder, learning to infer the latent variables $\boldsymbol{L}$ from the leading sine coefficients, can be done either with a feedforward neural network or Double DMAPs. The fact that the latent variables are found to be one-to-one with the leading sine coefficients does not mean that we need to map back and forth between the latent variables and the leading sine coefficients during integration. It allows us to integrate in either set of variables (here we use the sine coefficients) and then uniquely map each point in one set to the other.

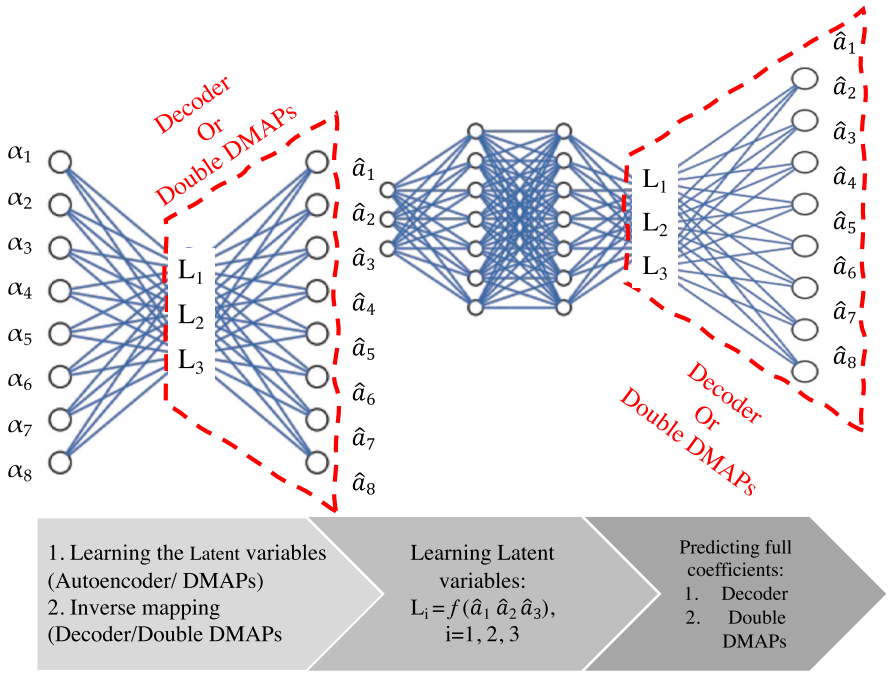

Fig. 5. Schematic representation of computational workflow: First step includes learning the minimum representation of the Approximate Inertial Manifold either with DMAPs or an autoencoder, as well as the inverse transformation, i.e. from the latent variables to the sine coefficients. Secondly, the latent variables are learned as a function of the leading three sines or POD coefficients, and finally, the full coefficients are predicted either with the decoder or Double DMAPS.

Alternatively, and this does not require training an additional regression scheme, the decoder part of the autoencoder can be used to compute an *inverse-map*. This inverse map utilizes the leading Fourier modes, $\tilde{\boldsymbol{\alpha}}$, in which the dynamics have evolved, and the trained decoder, to find the latent autoencoder variables that minimize the algebraic optimization constraint

$$\underset{\boldsymbol{L}}{\operatorname{argmin}} ||\tilde{\boldsymbol{\alpha}} - \text{decoder}(\boldsymbol{L})||_F. \tag{15}$$

For the optimization in Equation (15) only the variables $\tilde{\alpha}_1, \tilde{\alpha}_2, \tilde{\alpha}_3$ computed by the decoder are used; the latent autoencoder variables are denoted as $\boldsymbol{L}$, the leading Fourier modes as $\tilde{\boldsymbol{\alpha}}$. After solving the optimization problem in Equation (15) the decoder can be used to recover all the Fourier modes given $\boldsymbol{L}$.

This second use case of the autoencoder allows us to map from the leading Fourier modes to the latent space and back to the full Fourier models without the need of constructing an additional regression scheme. Once the latent variables are predicted, the decoder of the autoencoder or the inverse transformation from the DMAP to the ambient coordinates, is used to approximate the full set of reconstructed coefficients.

4. Results

Before presenting our results we remind the reader, through the illustration in Fig. 6 of the basic premise and the various errors associated with the post-processing Galerkin concept. The main *premise* is that the distance Δ_1, between the projection of the true solution (point 5) and the truncated Galerkin solution (point 3) is much smaller than the distance Δ_3 between point 3 and the true solution (point 1), the total error of truncated Galerkin approximation [23]. This motivates the need for post-processing, which establishes that the distance Δ_4 between point 1 and the post-processed Galerkin (point 2) is also much smaller than the total error (and comparable to Δ_1) as shown in Fig. 6.

4.1. Euler-Galerkin vs. neural-network AIMs: Chafee-Infante

A collection of time-instances of the Chafee-Infante equation, along trajectories for various initial conditions, that are sufficiently close to the global attractor are used for the purposes of the presented results. We make sure the density of the data is uniform by subsampling the collected data by using the *datafold* Python library [38]. We start by providing a comparison between the solution obtained with the three sine coefficients, here considered as the ground truth, (cf Fig. 7a) and the truncated equations with the first two modes. The different post-processing schemes are applied to the solution of the truncated equations at the end of the desired integration. The comparison between the two is shown in Fig. 7 where the reconstructed solution is shown with a blue dashed line and the ground truth simulation with a red line. The relative error along each step of the time integration until time $T = 5$, is shown

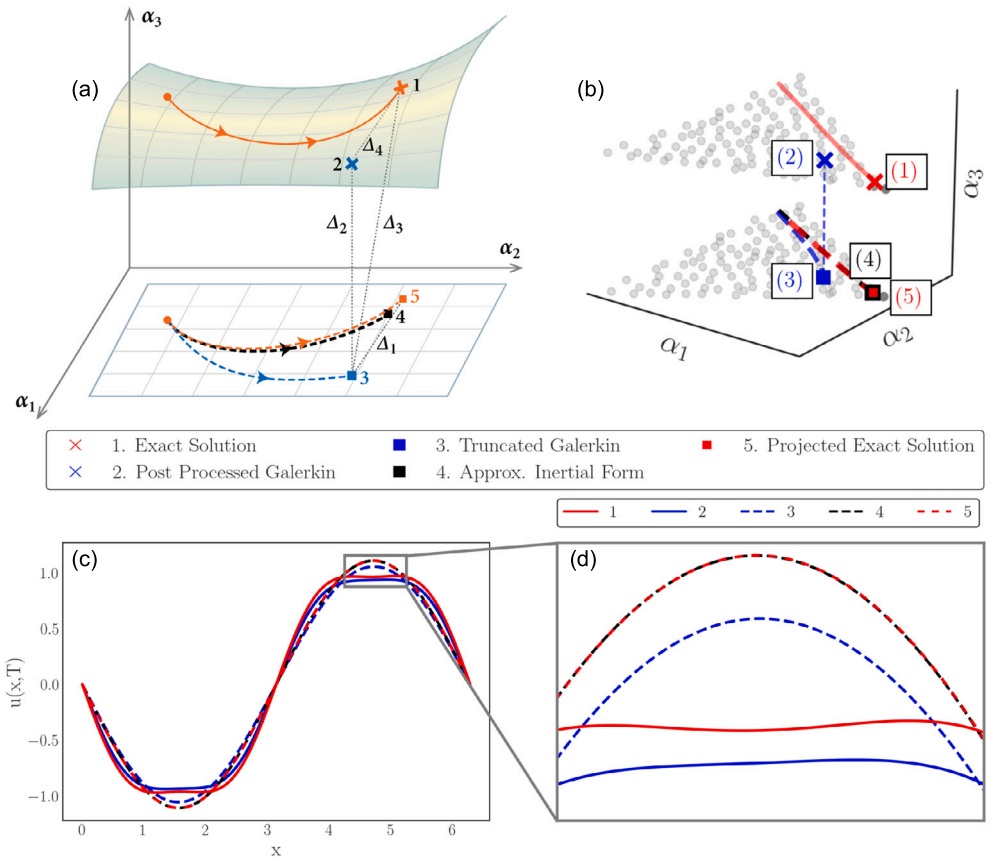

Fig. 6. (a) A schematic illustrating the benefits of the post-processing Galerkin methodology. A trajectory of the exact solution is shown *on the manifold* in a_1, a_2, a_3 as a red solid trajectory, its final state denoted with an x marker and (1). The projection of the exact solution in a_1, a_2 is shown with a red dashed line, its final state denoted as (3), and a blue square. The trajectory integrated by using the approximate inertial form is shown with a black dashed line, and its final state is shown with a black square and denoted as (4). The trajectory integrated by using the truncated Galerkin is shown with a blue dashed line, its final state denoted as (3) and a blue square, and its post-processing (mapping) on the manifold, denoted as (2), and indicated by a blue x marker. The dotted line Δ_1 shows the distance between (1) and (5), the dotted line Δ_2 shows the distance between (2) and (3), the dotted line Δ_3 shows the distance between (1) and (4). The main *premise* of post-processing Galerkin is that Δ_1 and Δ_4 are much smaller than Δ_2. (b) The same components used in (a) are shown for the Chafee-Infante PDE. (c) The reconstructed solution in $u(x,T)$ for all possible options. (d) A blow-up of the reconstruction in $u(x,T)$. (For interpretation of the colors in the figure(s), the reader is referred to the web version of this article.)

in Fig. 7b. It is worth pointing out that, because the training data set is uniformly sampled on (or very close to) the global attractor (inertial manifold), all the trajectories remain on the global attractor, regardless of the integration time.

The solution of the 2D truncated dynamics is then corrected, using the value of $\alpha_3(T)$ as predicted by a neural network (described in Sec. 3.1.1), using as inputs, the values of $\alpha_1(T)$ and $\alpha_2(T)$ at the final time-step, $t = T$. The results are shown in Fig. 7c with a dashed blue line; included in the same figure, with a solid blue line, is the solution corrected with the theoretically (Euler-Galerkin AIM) derived value of $\alpha_3(T)$. The relative error along the integration time till time $T = 5$ for the ML-derived α_3 is shown in Fig. 7d. Both, the ML-derived and the theoretical corrections help recover the accuracy and both lead to a mean absolute percentage error (MAPE) of less than 1%. The mean absolute percentage error (MAPE) is also computed at the same time instance ($T = 5$) but for 100 randomly selected initial conditions, for the 2D and the ML-corrected 2D model. This is shown in Fig. 7e, where the favorable effect of the correction on the mean absolute percentage error is clearly visualized. In these and in subsequent results, the MAPE refers to point-wise average of the absolute percentage error in each sample.

As an alternative, it is also possible to correct the *learned* ODE in two dimensions, derived as described in 3.1.2. The accuracy achieved is similar to the accuracy of the *true* truncated 2D model.

4.2. Kuramoto-Sivashinsky: data-driven latent spaces and their AIMs

The KS equation is selected in order to explore the application of the proposed methodology in cases where the minimum dimension of the Approximate Inertial Form (AIF) is not known a priori, although it has been argued to be three-dimensional [29]. Nevertheless, the truncated dynamics *are not always quantitatively close to the actual behavior*, which will be addressed with "Gray-Box" modeling. The identification of the minimum dimension of the AIF is an important challenge in the implementation of post-processing Galerkin methods and will be addressed here with two different approaches: nonlinear manifold learning and in particular Diffusion Maps discussed in 3.2.1, and autoencoders discussed in 3.2.2.

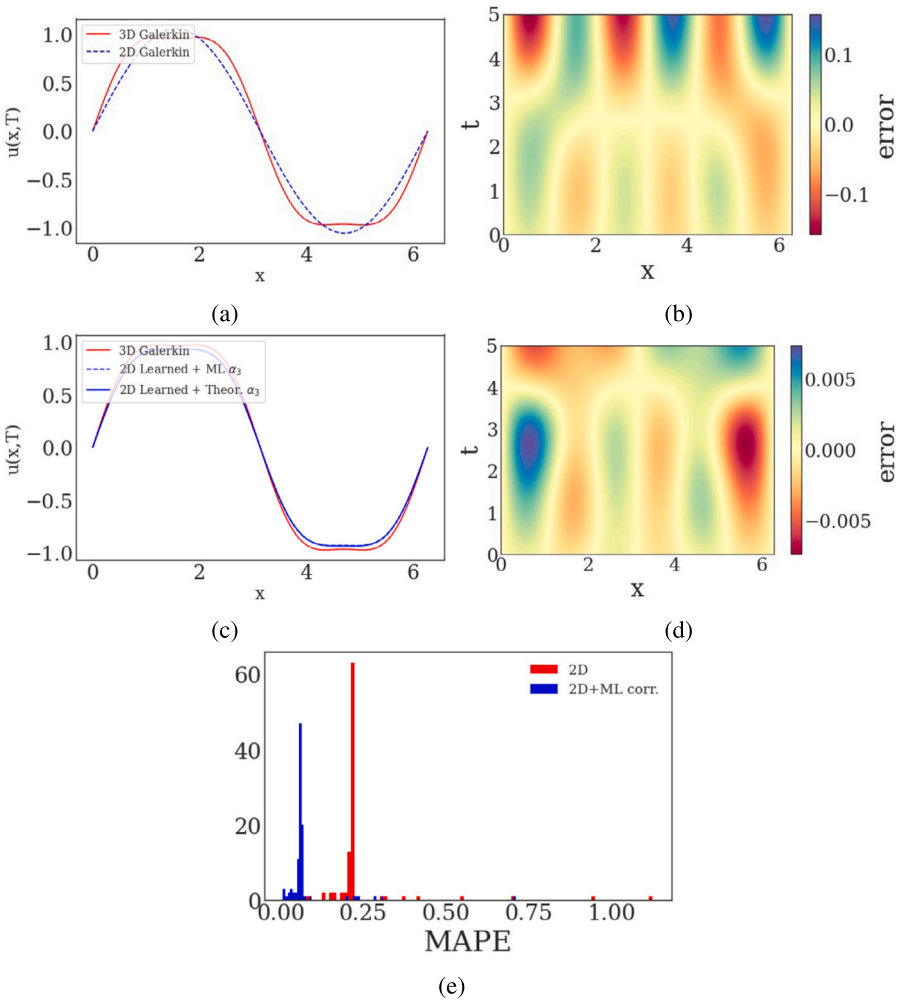

Fig. 7. Solution of Chafee Infante reconstructed in physical space; (a) The result of the 3D and the 2D Galerkin are shown in red and dashed black line respectively; (b) Relative error of reconstructed 2D Galerkin solution at each time-step; (c) Comparison of 3D Galerkin (red line) to the 2D Galerkin corrected with the neural network-derived term (dashed blue line); and the 2D Galerkin corrected with the theoretically derived α_3; (d) Relative error of reconstructed solution of the 2D ODE, corrected with the ML-derived α_3, at each time-step; (e) Histogram of the mean absolute percentage error of the 2D and ML-corrected 2D model at time $T = 5$, for 100 randomly selected initial conditions.

4.2.1. Learning the dimensionality of the latent space

Autoencoders

A collection of data is sampled for the KS parameter value $\nu = 33$, in various time instances of time-integration sufficiently close or on the global attractor. The data are used as inputs to an autoencoder and are reduced by the encoder into a low dimensional bottleneck layer which parametrizes an approximation of the inertial manifold. It is then possible to map to the approximation of the high dimensional variables with the decoder. The encoder/decoder components of the network can be used independently as it will be demonstrated in a subsequent section to improve the accuracy of the reduced order model.

The three latent variables of the bottleneck layer are one-to-one functions of the first three sine coefficients, α_1, α_2 and α_3. This is shown in Fig. 8, where the three bottleneck variables are plotted and colored according to the three sine coefficients. The smooth color variation suggests a one-to-one correlation between the latent and the ambient variables. It so happens that each one of the sine coefficients is one-to-one with each of the Diffusion Maps coordinates (the comparison is shown in A.8).

In Fig. 9, the histogram of the Jacobian determinant's values, $\det(\mathbf{J}_f(\mathbf{L}))$, along the training and test data shows that this quantity is always positive and thus the mapping $f : \boldsymbol{L} \to \tilde{\boldsymbol{\alpha}}$ is locally invertible.

The one-to-one relationship between the leading sine coefficients and the autoencoder's latent variables $\boldsymbol{L}$ facilitates the second use case of the autoencoder we discussed earlier. This second use case utilizes the decoder to solve an *inverse-problem* and map the leading sine coefficients $\tilde{\boldsymbol{\alpha}}$ to the autoencoder's latent space. Since, we showed that $f : \tilde{\boldsymbol{L}} \to \tilde{\boldsymbol{\alpha}}$ is a locally invertible map we can use the trained decoder and estimate $\boldsymbol{L}$ given $\tilde{\boldsymbol{\alpha}}$ by solving the optimization problem described in Equation (15). As initial conditions to solve the optimization problem randomly sampled points from the training set were used. After optimization, the decoder can be used

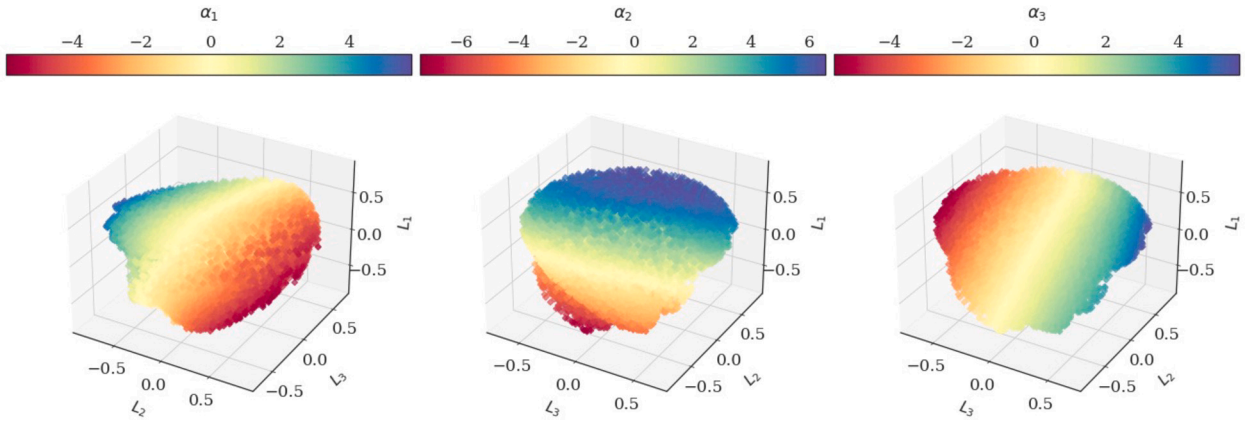

Fig. 8. Latent variables of the autoencoder bottleneck layer; The three latent variables colored by the value of the first three sine coefficients, α_1 (left), α_2 (center) and α_3 (right). Smoothness in color gradation suggests a one-to-one relation.

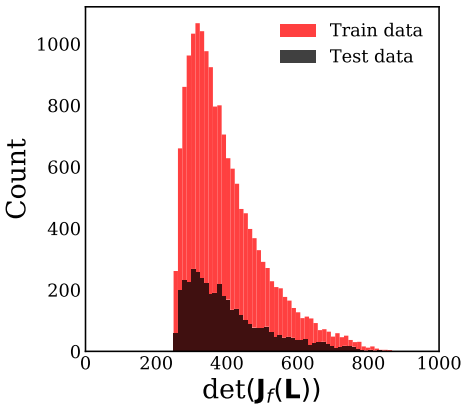

Fig. 9. The histogram of the determinant of the Jacobian $\det(\mathbf{J}_f(\mathbf{L}))$ computed along the training and test sets with automatic differentiation of the decoder.

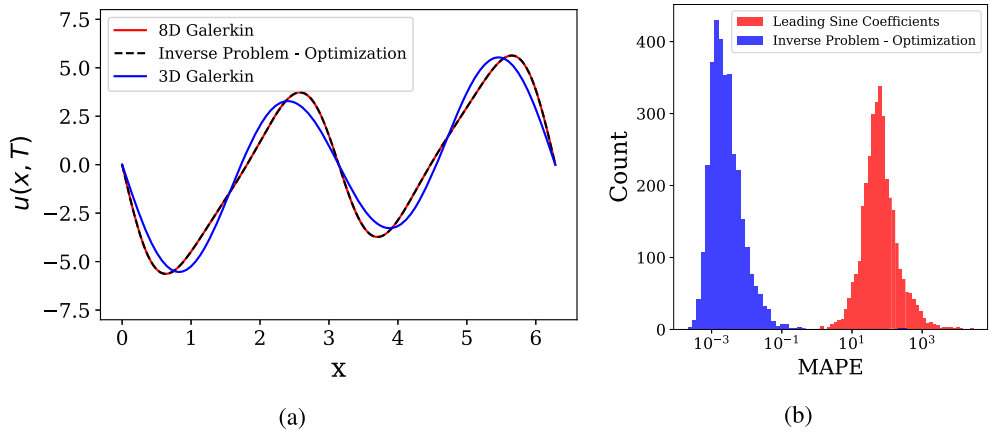

Fig. 10. (a) A visual comparison for one data point between the true solution, the solution obtained after solving the inverse problem, and the truncated solution obtained by using the first three leading sine coefficients. (b) The mean absolute percentage error (MAPE) across all the test points between the true solution and (i) the solution based on the leading sine coefficients (red histogram) (ii) the solution obtained after solving the inverse problem with optimization (blue histogram).

to reconstruct the remaining sine coefficients and from those the solution in $u(x,t)$ space, from the obtained values in autoencoder's latent space.

In Fig. 10a, we contrast, for one reconstructed trajectory (i) the true solution $u(x,T)$ obtained from the full equations, (ii) the reconstructed solution based on the first three learned sine coefficients, and (iii) the reconstructed solution obtained by solving the inverse map and using the decoder to reconstruct the full solution. In Fig. 10b the histograms show a comparison of the MAPE in

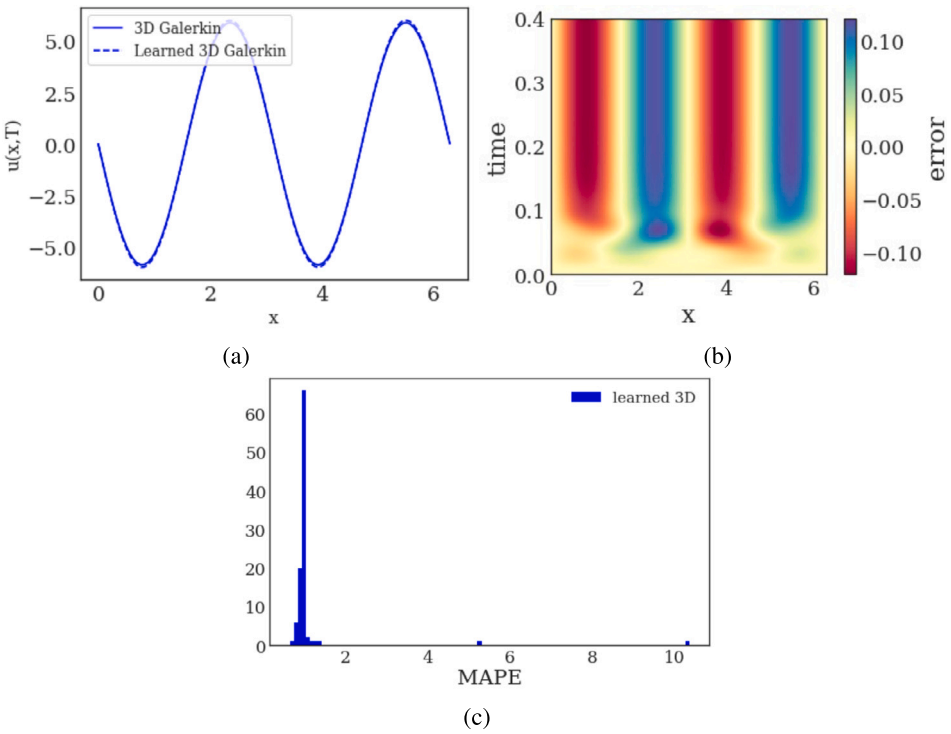

Fig. 11. (a) Reconstructed solution of the *learned* 3D equation (broken line) and the actual 3D dynamics (solid line), showing almost perfect agreement. (b) Relative error of the learned and true 3D solutions along the integration time. (c) Histogram of the mean absolute percentage error of the learned 3D model at time $T = 0.06$, for 100 randomly selected initial conditions.

$u(x,t)$ space between the true solution and the solutions based on (i) the three leading sine coefficients and (ii) the solution obtained after implementing the optimization step.

Diffusion Maps and their data-driven AIMs

Diffusion Maps is implemented to encode the high dimensional data to a low dimensional manifold parametrized by three Diffusion Maps coordinates shown in Fig. 20 of the Appendix. The Diffusion Maps coordinates ϕ_1, ϕ_2 and ϕ_3 are one-to-one with the coefficients of the first three sine terms. This is shown in Fig. 20, of the Appendix, by the smooth color transition in the diffusion maps plot when colored by α_1, α_2 and α_3.

The sine coefficients, α_i, are reconstructed with the help of Double DMAPS and by the decoder of the autoencoder as discussed in Secs. 3.2.1 and 3.2.2 respectively. The MSE for the Double DMAPs approach is 0.00492, whereas for the autoencoder it is 0.0155. The precision of the autoencoder decreases for higher harmonics, which leads to the overall drop in accuracy of the reconstruction (this comparison is shown in the Sec. A.8, Fig. 21). Double DMAPs predicts accurately *all* the coefficients (this comparison is shown in the Sec. A.7).

4.3. Data-driven post-processing Galerkin

Having established that the first three sine coefficients are one-to-one with the data-driven latent variables, the next step is to learn a data-driven ODE of the time evolution of the first three reconstructed sine coefficients as described in Sec. 3.1.2.

The feedforward neural network is trained using as input the values of $\hat{\alpha}_1$, $\hat{\alpha}_2$ and $\hat{\alpha}_3$, that are reconstructed by the latent space learning methods, i.e. autoencoders and DMAPs (the results of latent space identification and reconstruction of the hight-dimensional variables are presented in the SI). The predicted time-derivatives, the right-hand-side of the learned ODE, for each one of the sine coefficients are pictured in Fig. 22 (c.f. A.8) versus the actual values of that components time derivative, $\dot{\alpha}$. The top row shows the predicted right-hand-side from the three sine coefficients resulting from the autoencoder, with $MSE = 9.5$. The respective predictions from the Double DMAPs reconstruction are shown on the bottom row, with $MSE = 2.2$.

The neural network-derived approximation is then used in conjunction with an ODE solver, such as the Runge-Kutta, in order to integrate in time. The outcome of integration is reconstructed in physical space and compared to the outcome of the ground truth integration (in 8D) and also the reconstructed solution using only the first three modes of the ground truth. This is shown in Fig. 11, alongside the error between the learned 3D ODE and the actual 3 modes of the Galerkin approximation, which demonstrates that the learned ODE predicts accurately the *low dimensional time evolution of the first three modes*.

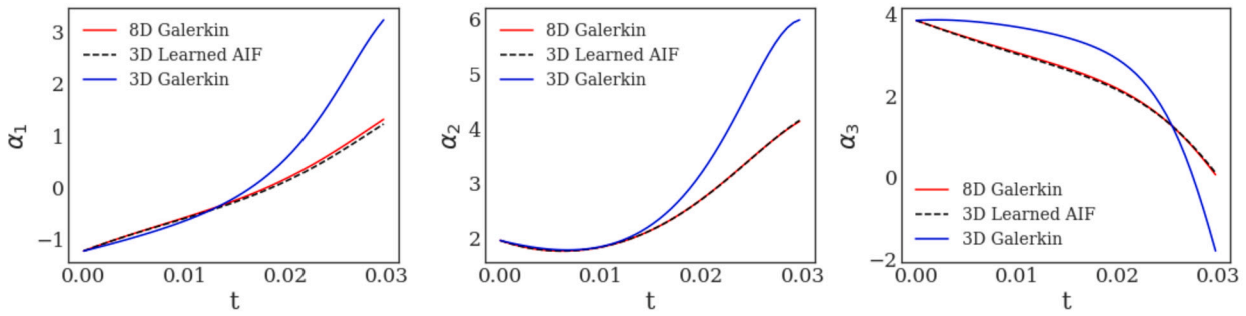

Fig. 12. Left: Comparison of time evolution of the first three sine coefficients of the Galerkin discretization; 8-dimensional (red), learned 3-dimensional (black) the 3-dimensional (blue) Galerkin discretization.

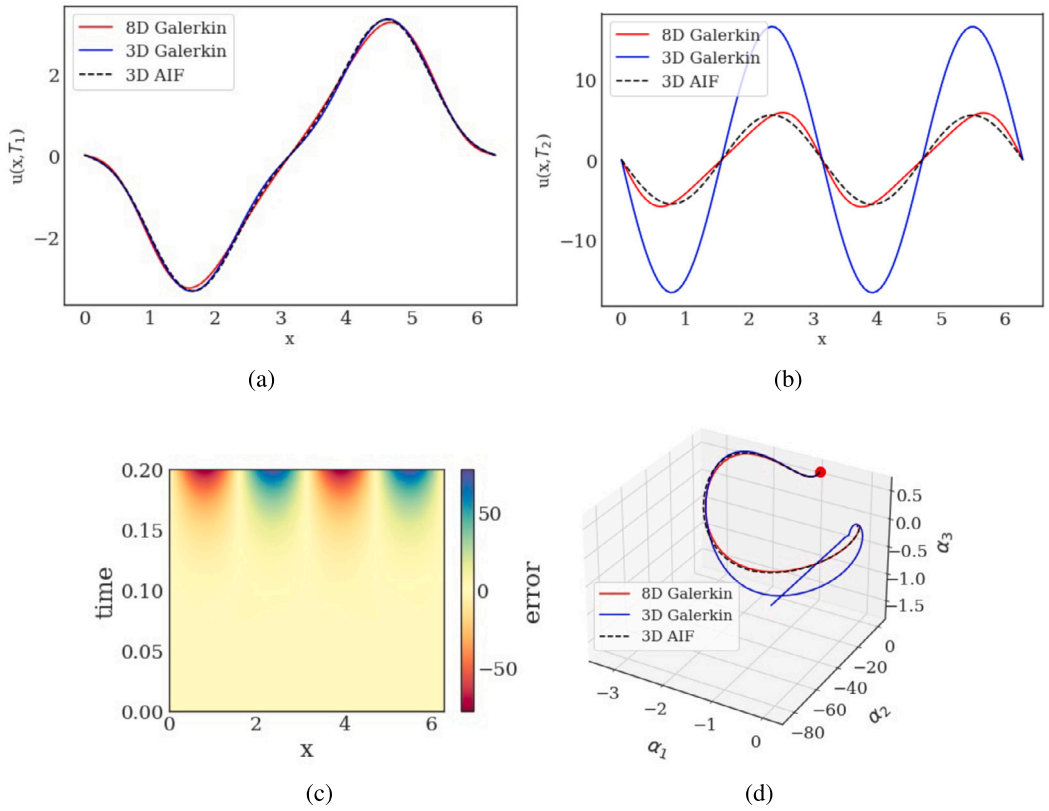

Fig. 13. Comparison of the solution, in physical space, in two time instances (a) T_1, where the 3D and 8D dynamics are sufficiently close and (b) T_2, when they are quite far apart; (c) Relative error between the truncated and the learned 3D AIF dynamics; (d) Comparison of time evolution of the first three sine coefficients of the Galerkin discretization: 8-dimensional (solid line), learned 3-dimensional (broken line) the 3-dimensional AIF (dotted line) Galerkin discretization.

4.3.1. When post-processing Galerkin works, when it does not work, and how to fix it

It is worth looking into the time evolution of the first three modes, α_1, α_2 and α_3 that result from the truncated 3D dynamics and compare it to the evolution of the first three terms of the full 8D dynamics and those of the learned AIF ODE. This is shown in Fig. 12, where it becomes evident that the first three terms of the learned 3D ODE are close to the trajectory of the first three terms of the 8D Galerkin. In contrast, the truncated 3D dynamics deviate significantly, past a certain point in time, from the ground truth dynamics. This observation suggests that using a post-processing scheme *directly on the truncated equations* won't be able to correct the dynamics. This motivates us to use an ML correction so-called "Gray-Box" model discussed further in the next section.

In essence, the post-processing Galerkin method relies on the premise that the solution of the truncated problem is reasonably close to the projection of the ground truth solution. Here it is demonstrated that even though the AIM is indeed three-dimensional, in the range of physical parameters examined, the truncated long term dynamics in three dimensional space are not accurate. This is demonstrated further in Fig. 13, where the solution in physical space is reconstructed from the 8D Galerkin, the 3D Galerkin and the learned 3D AIF ODEs, in different instances along the same trajectory. At initial stages of the trajectory, the solutions of the three methods are reasonably close, as shown in Fig. 13a. Later in time, the truncated 3D solution is growing quantitatively further apart

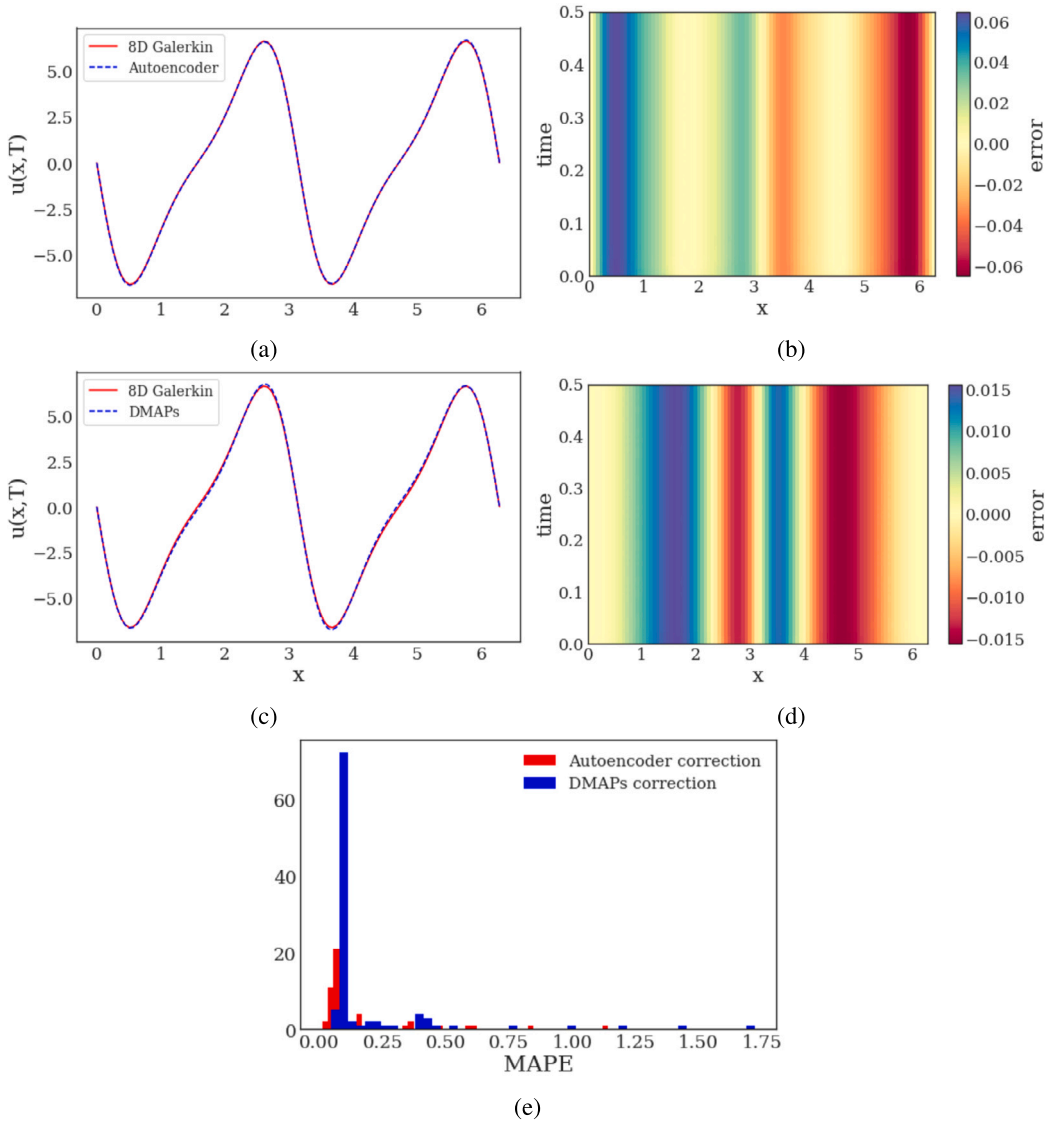

Fig. 14. (a) Comparison, at $\alpha = 33$ and $T = 0.5$ of the reconstructed Kuramoto-Sivashinsky solution, between the 8-dimensional Galerkin (solid red line) and the 3-dimensional learned AIF ODE corrected with the decoder-derived higher harmonics terms (broken blue line); (b) Relative error over the time integration interval. (c) Comparison, at $\alpha = 33$ and $T = 0.5$ of the reconstructed Kuramoto-Sivashinsky solution, between the 8-dimensional Galerkin (solid red line) and the 3-dimensional learned AIF ODE corrected with the DMAPs-derived higher harmonics terms (broken blue line); (d) Relative error along the time-span of integration. (e) Histograms of MAPE of the autoencoder-corrected and DMAPs-corrected solution at time $T = 0.5$, for 100 randomly selected initial conditions.

from the ground truth, whereas the learned 3D AIF ODE follows closely the 8D dynamics (Fig. 13b). This is made clear in Fig. 13c, where the relative error between the learned and the truncated 3D solution is plotted along the trajectory. This can also be observed in phase space shown in Fig. 13d, on the right. The red point corresponds to the initial condition. The values of sine coefficients initially evolve in a similar manner but eventually, the truncated 3D dynamics deviate.

Data-driven post-processing Galerkin

To recover the values of all the α_is, necessary for accurate reconstruction, the first step involves predicting the latent variables, either the bottleneck variables from the autoencoder or alternatively the DMAPs latent coordinates. One way to achieve this, is with a feedforward neural network with three inputs (the first three sine coefficients) and three outputs (the latent variables). In this implementation, the neural network consists of 5 hidden layers with 80 neurons each and a *tanh* activation function, implemented in tensorflow [1]. The mean squared error is used as the loss function along with the Adam optimizer.

It is then possible to employ either Double DMAPs, in the case of DMAPs, or the decoder of the autoencoder, and predict the corresponding α_is with $MSE = 0.09$ for both cases. The reconstructed solution in physical space is compared to the ground truth in Fig. 14.

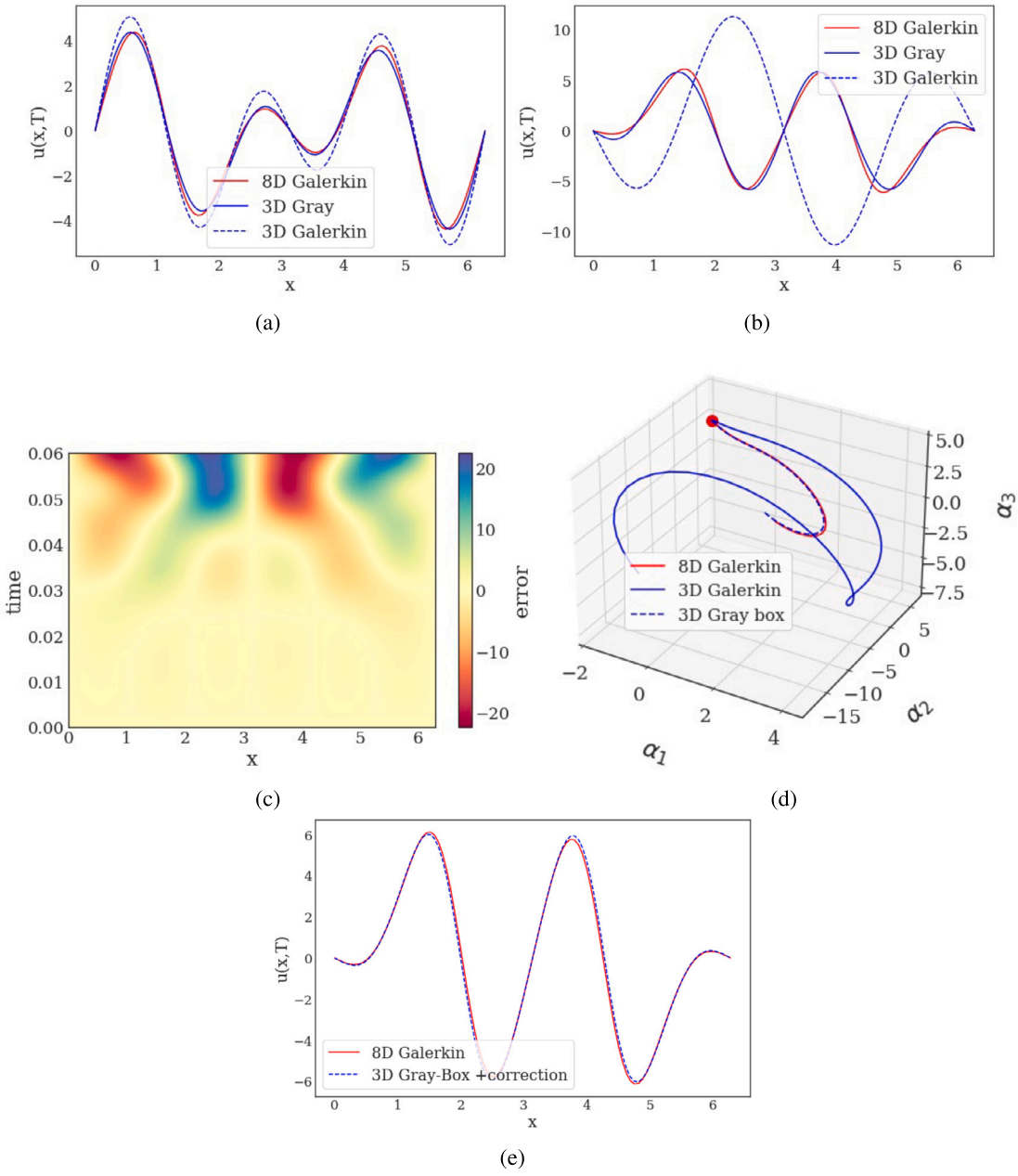

Fig. 15. Gray-Box correction of 3D Galerkin dynamics: Comparison of the solution, in physical space, in two time instances (a) $T_1 = 0.02$, where the 3D and the 8D Galerkin dynamics are sufficiently close and (b) $T_2 = 0.05$, when they are quite far apart; (c) Relative error between the truncated and the Gray-Box 3D dynamics; (d) Comparison of time evolution of the first three sine coefficients of the Galerkin discretization: 8-dimensional (solid red line), Gray-Box 3-dimensional (broken blue line) truncated 3-dimensional (blue solid line) Galerkin discretization (e) Comparison of the solution, in physical space, of the corrected Gray-Box with the 8D Galerkin at $T_2 = 0.05$.

When the PPG assumptions fail

If we have a accurate low-dimensional observation we can correct, in principle theoretically with Euler-Galerkin, or in practice with machine learning approaches as described above. If this is not available, then we proceed to improve the AIF itself though the Gray-Box approach.

The method's performance is demonstrated in Fig. 15 for two cases. In the first case (cf Fig. 15a), the 3D, 8D, and corrected Gray-Box dynamics are shown for the reconstructed physical space solution, at $T_1 = 0.02$, when the truncated 3D dynamics are close to the ground truth. At a later time-step, at $T_2 = 0.05$ (cf Fig. 15b), the truncated dynamics have deviated far from the truth. The Gray-Box model corrects the deviation in both cases and accurately captures the ground truth with the addition of post-processing terms, as seen in Fig. 15e.

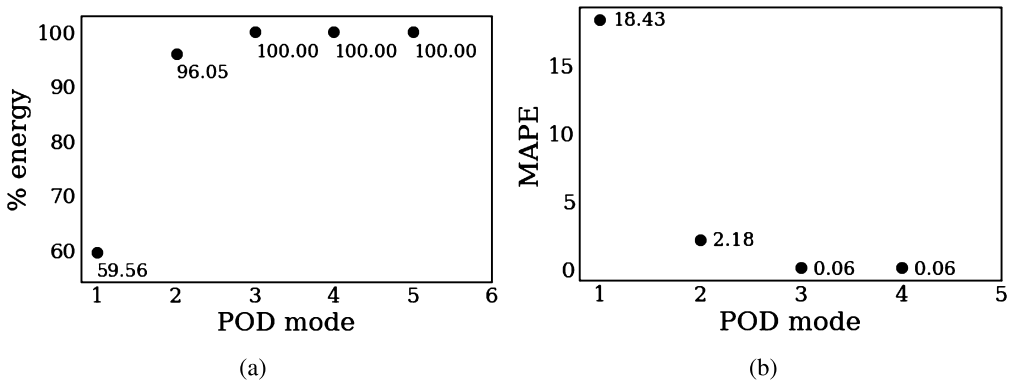

Fig. 16. (a) % energy of the data contained by progressively increasing POD basis size. 3 POD modes represent 99.99 % of the variance (b) MAPE of the dataset projected on progressively increasing POD basis, with respect to the original. When considering 3 POD modes, the MAPE drops to 0.06%.

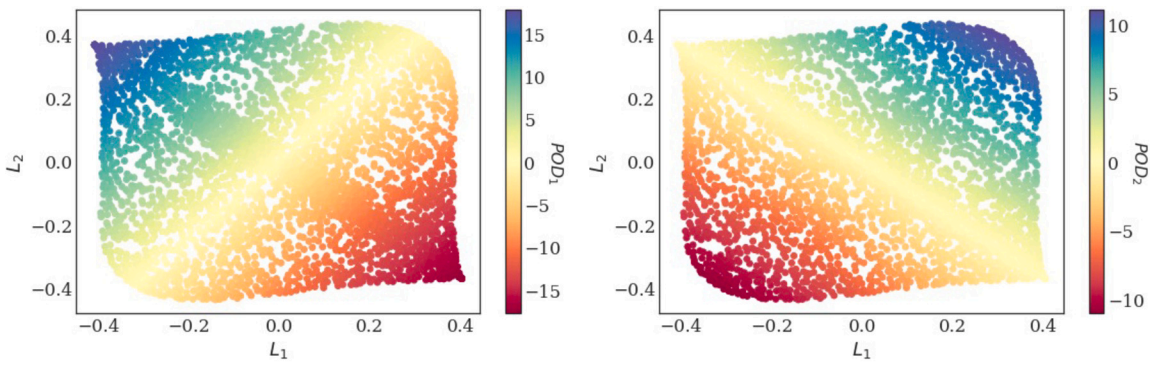

Fig. 17. Latent variables discovered by the autoencoder, whose inputs are POD coefficients values. The smooth color transition implies that the latent variables are 1-to-1 with the leading POD coefficients.

4.4. Using POD coefficients to parametrize the IM/AIM

Here, the implementation of the proposed workflow is presented in the case where the manifold is parametrized by data-driven POD coefficients, rather than sine coefficients, for the Chafee-Infante equation. To start with, the POD modes that contain the greatest percentage of variance of an ensemble of solutions in physical space, are identified. Three POD modes represent 99.99% of the energy of the dataset (cf. Fig. 16a), defined as the percentage of the cumulative sum of the leading three eigenvalues over the sum of all the eigenvalues.

The original dataset is then projected on the first three modes, leading to each solution vector, being represented by three coefficients. The mean absolute percentage error of the dataset, projected on a basis consisting of 3 POD vectors, is 0.06% (cf. Fig. 16b). We use this collection of POD coefficients, to discover the latent variables, here with an autoencoder with a 2-neuron bottleneck layer. The mean absolute percentage error achieved for the autoencoder-reconstructed POD coefficients is 1.2%. The latent variables are one-to-one with the two leading POD coefficients, as is evident in Fig. 17, where they are plotted and colored according to the values of the coefficients. The smooth color transition is indicative of the one-to-one relationship.

The time-evolution law of the ODE for the two leading POD coefficients, is then learned from data. This is achieved using a feed forward neural network consisting of two hidden layers with 20 neurons each. The *tanh* activation function is implemented and the mean squared error is used as a loss function. The learned ODE is integrated with a Runge-Kutta solver over time $T = 5$. From the values of the two POD coefficients at the final time-step, the latent variables are then inferred using an appropriately trained neural network. Then, the decoder of the autoencoder is used to recover the entire set of POD coefficients. It is then possible to "lift" from POD space to the sine coefficients and reconstruct the solution: the solution reconstructed using 3 ML-derived terms compares very favorably to the ground truth solution in Fig. 18. For reference, in the same figure, the solution reconstructed from only the 2 POD coefficients, is also included.

5. Discussion-conclusions

In conclusion, this study has attempted to bridge theoretical approaches to reduced order modeling of dynamical systems (theoretically, closed form, approximations of AIMs and AIFs) with appropriately derived data-driven workflows. The data in question may consist of either (a) theoretical parametrizations of the IM (here sine coefficients) or (b) equally possibly, data-driven parametrizations (POD coefficients, autoencoder latent variables, manifold learning Diffusion Map coordinates). The use of machine learning

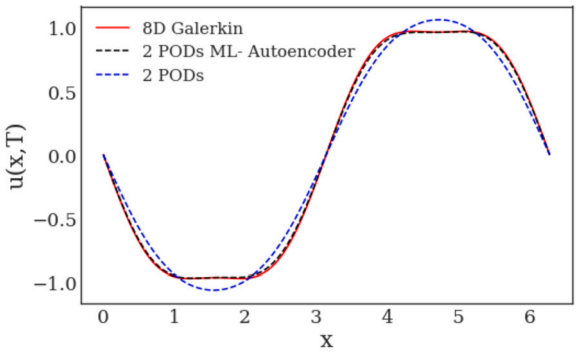

Fig. 18. Reconstructed solution of the Chafee-Infante equation, at time $T = 5$. The red line represents the ground truth, 8D Galerkin solution, which corresponds to 3 POD coefficients. The black broken line corresponds to the data-driven post-processed solution of the evolution of 2 POD coefficients. The uncorrected solution derived by 2 POD coefficients, is also depicted with a blue broken line.

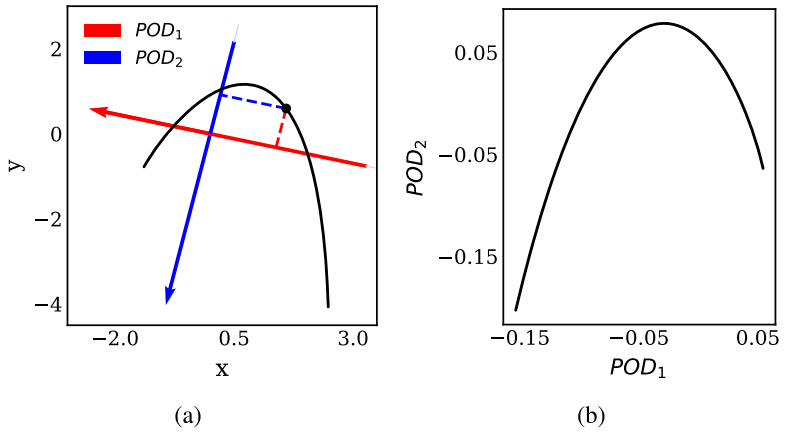

Fig. 19. (a) A data set sampled from the singularly perturbed system of ODEs is shown with a black solid line. The span of the first POD mode (POD_1) is shown with a red vector and the span of the second POD mode (POD_2) is shown with a blue vector. The projection of a data point (black solid circle) to POD_1 and POD_2 is depicted. (b) The components of the first POD vector (POD_1) versus the components of the second POD vector (POD_2). POD_2 can be seen as a quadratic function of POD_1.

techniques, specifically autoencoders and Diffusion Maps, allows for accurate and efficient modeling of high-dimensional systems while overcoming the limitations of traditional post-processing Galerkin methods.

Moreover, the proposed approach has demonstrated promising results in scenarios where the low-dimensional ROM significantly deviates from the correct long-term dynamics, which was previously challenging to address with post-processing Galerkin techniques. The introduction of a "Gray-Box" model that adds a correction to the truncated Galerkin helps it regain its accuracy; it then allows for post-processing steps to recover even higher levels of accuracy, in ambient space.

At this point, it is worth discussing how noise affects the robustness of the different methods; regarding the DMaps-based approaches, the question of noisy data has been addressed in detail in the paper by Coifman and Lafon [13]: the DMaps parametrization will be robust to output noise as long as the scale parameter $\sqrt{\epsilon}$, in the denominator of equation (25) in the Appendix, remains larger than the amplitude of the noise. Therefore, as long as the scale parameter is larger than the noise (and the noise is not large enough to distort the manifold) we expect the embedding to remain consistent. Along these lines, a more rigorous means of selection of this parameter was recently introduced in order to handle noisy data, based on a semigroup property for parameter tuning [52]. Autoencoders, have been used for denoising data; therefore, we expect them capable of handling data with small amplitudes of noise and act as filters to them [62].

For the Post-processing Galerkin, we assume that noise affects the solution in the same way that error in the truncated Galerkin. An error of the order $||e_m||$ in the computed Galerkin method yields an error of the $O(m^{-\alpha_2}) + ||e_m||$, where $O(m^{-\alpha_2})$ is the error between the exact solution of the system and the post-processing Galerkin based on the exactly computed solution of the Galerkin system, u_m. Therefore one has to require $||e_m|| = O(m^{-\alpha_2})$ in order to conclude robustness of the post-Processing Galerkin method with respect to errors committed in the computed Galerkin method. Exploiting this analysis we can also conclude that, by analogy, if we have noise of size smaller or equal to $O(m^{-\alpha_2})$ on the results of the truncated Galerkin we will also retain the robustness of the post-processing Galerkin (and our approach). A detailed analysis along these lines in presented in section A.3.

Overall, this work contributes to the growing body of literature on data-driven reduced order modeling techniques for dynamical systems and provides a valuable alternative to traditional post-processing Galerkin methods. The proposed workflows have the

potential to significantly improve the accuracy and efficiency of reduced order models, which has important implications for a wide range of applications, including but not limited to, aerospace engineering, biomedical engineering, and climate modeling.

A promising future direction of our current work for the construction of reduced-order models is the combination of data-driven techniques with physics-based techniques. The work of R. Geelen et al. [25], in which the parameterization of the data is achieved by combining linear subspaces - spanned by the first few POD vectors - and quadratic components, is the most pertinent to this direction. One could express the dynamics in terms of the first few POD vectors and use the quadratic correction only as a post-processing step to obtain a more accurate reconstruction at the end of the integration. The ability to find a quadratic correction could provide improved explainability to the post-processing step, that we lose by learning a black-box post-processing step in our current work. A visualizable example is shown in Fig. 19 where the 2-dimensional singularly perturbed system ($\dot{x} = 2 - x - y; \dot{y} = 1/\epsilon(x - y)$) was used to sample data. For this example one could *write* the dynamics in terms of POD_1 and express the correction from $POD_2 = f(POD_1)$ through a quadratic correction since POD_2 can be seen as a quadratic function of POD_1 (Fig. 19(b)).

6. Acknowledgments

The motivation for this work comes in part from initial efforts on reduced modeling of multiphase flows, as part of CML's Thesis. I.G.K. acknowledges partial support from the US AFOSR FA9550-21-0317 and the U.S. Department of Energy SA22-0052-S001. E.D.K. was funded by the Luxembourg National Research Fund (FNR), grant reference 16758846. For the purpose of open access, E.D.K. has applied for a Creative Commons Attribution 4.0 International (CC BY 4.0) license to any Author Accepted Manuscript version arising from this submission. C.M.L. received the support of a 'la Caixa' Foundation Fellowship (ID 100010434), code LCF/BQ/AA19/11720048. The research of E.S.T. was made possible by NPRP grant #S-0207-200290 from the Qatar National Research Fund (a member of Qatar Foundation), and is based upon work supported by King Abdullah University of Science and Technology (KAUST) Office of Sponsored Research (OSR) under Award No. OSR-2020-CRG9-4336. The work of E.S.T. has also benefited from the inspiring environment of the CRC 1114 "Scaling Cascades in Complex Systems", Project Number 235221301, Project A02, funded by Deutsche Forschungsgemeinschaft (DFG). For the purpose of open access, E.S.T. has applied a Creative Commons Attribution (CC BY) licence to any Author Accepted Manuscript version arising from this submission.

CRediT authorship contribution statement

Eleni D. Koronaki: Data curation, Formal analysis, Investigation, Writing – original draft, Writing – review & editing. **Nikolaos Evangelou:** Data curation, Formal analysis, Methodology, Writing – original draft, Writing – review & editing. **Cristina P. Martin-Linares:** Investigation, Methodology, Validation. **Edriss S. Titi:** Conceptualization, Funding acquisition, Supervision, Writing – review & editing. **Ioannis G. Kevrekidis:** Conceptualization, Formal analysis, Methodology, Supervision, Writing – original draft, Writing – review & editing.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

Data will be made available on request.

Appendix A

A.1. Post-processing Galerkin for the finite element method

Let X be the phase space of a nonlinear dissipative evolution equation of the form

$$\frac{du}{dt} + \nu Au = F(u).$$

Let X_H be a finite dimensional (e.g. finite element) space of spatial scale H with $\mathcal{P}_H$: $X \to X_H$ an orthogonal projection. The Galerkin approximate solution $u_H \in X_H$ satisfies the equation

$$\frac{du_H}{dt} + \nu \mathcal{P}_H A u_H = \mathcal{P}_H F(u_H), \text{ for } t \in [0, T].$$

Therefore, for a given Galerkin solution u_H and its time-derivative $\frac{du_H}{dt}$ over the interval $[0, T]$, the post-processing Galerkin solution is a function $v \in X$ (notice it is not in the complement of X_H, i.e. not in $X \ominus X_H$, but in X, so v involves both coarse as well as fine spatial scales), such that v satisfies

$$\nu Av = -\frac{du_H}{dt} + F(u_H)\,|_{t=T}\,.$$

The right-hand side is a given function at time $t = T$, and v solves a linear elliptic equation. However, in practice we solve an approximation of v say $\tilde{v} \in X_h$, where $h \ll H$, and X_h is a finer finite element space

$$\nu \mathcal{P}_h A \tilde{v} = \mathcal{P}_h \left(-\frac{du_H}{dt} + f(u_H) \right) |_{t=T} .$$

A.2. Euler-Galerkin algorithm applied to Chafee-Infante PDE

The implementation of the Euler-Galerkin algorithm described in Sec. 3.1.1 is shown here for the Chafee-Infante reaction-diffusion equation. For this PDE, as discussed in Sec. 4.1, a two-dimensional inertial manifold exists ($n = 2$) parameterized by the first two sine Fourier modes α_1, α_2. By using the Galerkin projection $u(x,t) \approx \sum_{i=1}^{m=3} a_i(t) sin(ix)$ a system of three coupled ordinary differential equations is derived. The derived system of equations reads

$$\dot{a_1} = -a_1 \nu + a_1 - \frac{3}{4}a_1^3 - \frac{3}{2}a_1 a_2^2 - \frac{3}{4}a_1^2 a_3 - \frac{3}{4}a_2^2 a_3 - \frac{3}{4}a_1 a_3^2 \tag{16}$$

$$\dot{a_2} = -4a_2 \nu + a_2 - \frac{3}{2}a_1^2 a_2 - \frac{3}{4}a_2^3 - \frac{3}{2}a_1 a_2 a_3 - \frac{3}{2}a_2 a_3^2 \tag{17}$$

$$\dot{a_3} = -9a_3 \nu + a_3 + \frac{a_1^3}{4} - \frac{3}{2}a_3^3 - \frac{3}{2}a_2^2 a_3 - \frac{3}{2}a_1^2 a_3 - \frac{3}{4}a_1 a_2^2. \tag{18}$$

The term $-9\alpha_3 \nu$ of the right-hand side of Equation (18) corresponds to the diffusion term and all the other terms of the right-hand side to the reaction terms. We take an implicit Euler step of Equation (18) of length τ by using as initial condition $\alpha_3(t = 0) = 0$. This gives us the expression

$$\alpha_3(\tau) = \alpha_3(0) + \tau \dot{a_3} \tag{19}$$

By moving the diffusive term to the left-hand side and solving in terms of α_3 we get the expression

$$\alpha_3 = \frac{\tau}{(1 + 9\tau\nu)} \left(a_3 + \frac{a_1^3}{4} - \frac{3}{2}a_3^3 - \frac{3}{2}a_2^2 a_3 - \frac{3}{2}a_1^2 a_3 - \frac{3}{4}a_1 a_2^2 \right). \tag{20}$$

We then perform one fixed point iteration by considering $a_3 = 0$ and $\tau = 1$. This leads to the Euler-Galerkin approximation

$$\alpha_3 = \frac{1}{4(1 + 9\nu)} \left(\alpha_1^3 - \alpha_1 \alpha_2^2 \right). \tag{21}$$

In our case, the Euler-Galerkin approximation in Equation (21) was used as one of the post-processing schemes to correct the solution of $\hat{u}(x,T)$ computed from the truncated dynamics.

A.3. Effect of error/noise in the Galerkin solution to the robustness of the ML-driven post-processing

For the post-processing Galerkin, we assume that noise affects the solution in the same way that error in the truncated Galerkin solution does. For the sake of simplicity we outline the response in the context of spectral Galerkin method in the space $H_m = span < w_1, w_2, ..., w_m >$; where $< w_j >$ form a basis of the phase space of solutions, say H.

Denote by $P_m : H \to H_m$ the orthogonal projection, and by $u_m \in H_m$ the solution of the m-truncated Galerkin method. The post-processing Galerkin method, capitalizes on the fact that

$$\sup_{0 \le t \le T} ||u(t) - u_m(t)|| = O(m^{-\alpha_1}) \gg \sup_{0 \le t \le T} ||P_m u(t) - u_m(t)|| = O(m^{-\alpha_2}), 0 < \alpha_1 < \alpha_2.$$

Let $q = \Phi_m(p)$, for $p \in H_m$, be the relevant approximate inertial manifold used in the post processing step. Observe that Φ_m is a m-dimensional Lipschiz manifold with a Lipschitz constant $K_m = O(m^{-\alpha_3})$ for some $\alpha_3 > 0$.

Let the actual numerically computed solution of the m-truncated Galerkin be $v_m = u_m + e_m$, where u_m is the exact solution of the m-truncated Galerkin system with error e_m.

By using the Lipsichtz property of $\Phi_m(p)$ one has

$$||\Phi_m(v_m) - \Phi_m(u_m)|| \le K_m ||e_m||.$$

Consequently the total error committed in the post-processing step is

$$||(v_m + \Phi_m(v_m)) - (u_m + \Phi_m(u_m))|| \le ||e_m||(1 + K_m).$$

However, using the fundamental approximation property of AIM, namely that

$$\sup_{0 \le t \le T} ||(I - P_m)u(t) - \Phi_m(P_m u)|| = O(m^{-\alpha_2}),$$

together with the above, implies the following total error in the post-processing Galerkin method:

$$||u - (v_m + \Phi_m(v_m))|| = ||P_m u + (I - P_m)u - (v_m + \Phi_m(v_m))||. \tag{22}$$

If we add and subtract in Equation (22) $u_m, \Phi_m(P_m u), \Phi(u_m)$ and use the triangle inequality we get

$$||P_m u + (I - P_m)u - (v_m + \Phi_m(v_m)) + u_m - u_m + \Phi_m(P_m u) - \Phi_m(P_m u) + \Phi(u_m) - \Phi(u_m)|| \leq$$
$$||(P_m u - u_m)|| + ||u_m - v_m|| + ||(I - P_m)u - \Phi_m(P_m u)|| + ||\Phi_m(P_m u) - \Phi_m(u_m)|| + ||\Phi_m(u_m) - \Phi_m(v_m)|| =$$
$$O(m^{-\alpha_2}) + ||e_m|| + O(m^{-\alpha_2}) + K_m||P_m u - u_m|| + K_m||e_m||. \quad (23)$$

Therefore,

$$||u - (v_m + \Phi_m(v_m)|| \leq O(m^{-\alpha_2}) + ||e_m|| + O(m^{-(\alpha_2+\alpha_3)}). \quad (24)$$

Conclusion An error of the order $||e_m||$ in the computed Galerkin method yields an error of the $O(m^{-\alpha_2}) + ||e_m||$, where $O(m^{-\alpha_2})$ is the error between the exact solution of the system and the post processing Galerkin based on the exactly computed solution of the m-truncated Galerkin system, u_m. Therefore one has to require $||e_m|| = O(m^{-\alpha_2})$ in order to conclude robustness of the Post Processing Galerkin method with respect to errors committed in the numerically computed Galerkin method. Exploiting this analysis we can also conclude that, by analogy that if we have noise of size smaller or equal to $O(m^{-\alpha_2})$ on the results of the truncated Galerkin we will also retain the robustness of the Post Processing Galerkin (and our approach)

A.4. Convergence in the presence and absence of time-scale separation

Dissipative evolution equations are known to possess an absorbing ball in the phase space, whose radius depends on the physical parameters of the particular equation.

An absorbing ball is a ball that is invariant under the dynamics and attracts all the solutions with initial data outside the absorbing ball.

Notably, for most dissipative evolution equations that arise in physical applications, the absorbing ball attracts solutions starting outside of it *exponentially fast* with a rate that depends on the physical parameters of the particular equation.

Moreover, once a solution enters into the absorbing ball it converges toward the global attractor at rates whose speed (fast or slow) cannot be determined in advance.

In the presence of separation of time-scales:

It is worth mentioning that, in the presence of large enough gap (separation of time scales), say of $O(m^{\alpha})$, for some $\alpha > 0$, for m large enough positive integer, between the backward dynamics of the m resolved modes and the forward dynamics of the unresolved, faster ones, one should be able to show the existence of a Lipschitz m-dimensional globally invariant manifold. In addition, such a manifold will attract every solution, with initial data inside the absorbing ball, exponentially fast with a rate of $O(m^{\alpha})$.

Such a manifold is called an Inertial Manifold (IM), and in particular it contains the global attractor.

Observe that the existence of IM relies on the separation of time-scales; however, global attractors exist regardless of the presence or absence of separation of time-scales.

In the absence of separation of time-scales:

As we stated above, the dissipative evolution equation will still possess a global attractor. Even though in this case we are unable to show the existence of an invariant IM that contains the global attractor, we are able to construct a k-dimensional Lipschitz manifold, that is still called an AIM. At the moment k is an arbitrary positive integer.

In fact, one can construct the AIM $M_0 = graph(\Phi_0)$, where Φ_0 is a Lipschitz function from the k resolved modes to the complement.

Properties of the AIM:

- For k large enough, depending on the physical parameters, one can show that, even though the global attractor is not contained in the AIM M_0 it is contained in a thin layer of thickness of the order $O(k^{-\beta})$, for some $\beta > 0$, about M_0.
- For every initial data inside the absorbing ball the corresponding solution of the dissipative evolution equation converges exponentially fast to the above mentioned thin layer about the graph of the AIM M_0, with a rate of the order $O(k^{\gamma})$, for some $\gamma > 0$.

Summary All solutions outside the absorbing ball converge exponentially fast to the absorbing ball at a rate that depends on the physical parameters. All solutions starting inside the absorbing ball converge to a thin layer about the AIM M_0, whose thickness is of the order $O(k^{-\beta})$, for some $\beta > 0$ (k is the dimension of M_0). Furthermore the convergence is exponentially fast with rate of the order $O(k^{\gamma})$, for some $\gamma > 0$.

A.5. Diffusion maps

The Diffusion Maps algorithm reveals the *intrinsic geometry* of a data set $\mathbf{X} = \{\boldsymbol{x}_i\}_{i=1}^{N}$, where each data point $\boldsymbol{x}_i \in \mathbb{R}^m$, by constructing a random walk on $\mathbf{X}$. This random walk is constructed by means of an affinity matrix $\mathbf{A}$ with entries computed in terms of a kernel

$$A_{ij} = \exp\left(\frac{-||\boldsymbol{x}_i - \boldsymbol{x}_j||_2^2}{2\varepsilon}\right) \quad (25)$$

where ϵ is a positive hyperparameter that specifies the rate of decay of the kernel (kernel bandwidth). The Gaussian kernel, Equation (25), is typically chosen for the construction of the affinity matrix $\mathbf{A}$.

To obtain a random walk (parametrization) of $\mathbf{X}$ regardless of the sampling density the normalization

$$\mathbf{K} = \mathbf{P}^{-\alpha}\mathbf{K}\mathbf{P}^{-\alpha}, \text{ where } P_{ii} = \sum_{j=1}^{N} A_{ij} \tag{26}$$

is applied, with $\alpha = 1$ to factor out the density effects.

A second normalization,

$$\tilde{\mathbf{K}} = \mathbf{D}^{-1}\mathbf{K} \text{ where } D_{ii} = \sum_{j=1}^{N} K_{ij}, \tag{27}$$

is applied to recover the row-stochastic matrix $\tilde{\mathbf{K}}$. The eigendecomposition of $\tilde{\mathbf{K}}$,

$$\tilde{\mathbf{K}}\boldsymbol{\phi}_i = \lambda_i \boldsymbol{\phi}_i \tag{28}$$

gives a set of eigenvectors $\boldsymbol{\phi}$ and eigenvalues λ. Proper selection of the eigenvectors that parameterize independent directions, (known as non-harmonics) is needed. This selection in practice can be achieved by using the local linear regression algorithm proposed in Dsilva et al. [16]. If the number of non-harmonic eigenvectors is smaller than the original dimension $\tilde{n} < n$ then those eigenvectors $\boldsymbol{\Phi} = \{\phi_1, \dots, \phi_{\tilde{n}}\}$ can provide a more parsimonious representation of the original data and thus to obtain dimensionality reduction.

It is therefore important to discover which eigenvectors parametrize independent directions, and do not span the same direction with different frequencies (*harmonics*).

To achieve this, the local linear regression algorithm, proposed in Dsilva et al. [16] is used, according to which, each DMAP coordinate is fitted as a function of the previous ones. To select the DMAP coordinates that are independent, the "goodness of fit" of this functions is used: A good fit is associated with a ϕ_k that is a harmonic function of the previous eigenmodes, whereas a bad fit signifies that ϕ_k is a new independent direction on the data manifold.

A.6. Geometric harmonics and double diffusion maps

Geometric Harmonics [12] is a regression scheme *traditionally* applied on a data set $\mathbf{X}$ to extend a function f. Extending means that we are able to evaluate the function f for points "outside" of $\mathbf{X}$, for $x_{new} \notin \mathbf{X}$.

In our previous work [17] we introduced a special case of Geometric Harmonics, termed Double Diffusion Maps (Double DMAPs), able to regress functions directly on the reduced Diffusion Maps coordinates $\boldsymbol{\Phi}$. In this case, similar to the first round of Diffusion Maps, an affinity matrix is computed

$$A^*_{ij} = \exp\left(\frac{-||\boldsymbol{\phi}_i - \boldsymbol{\phi}_j||_2^2}{2\epsilon^*}\right). \tag{29}$$

An eigendecomposition of the symmetric and positive semidefinite matrix $\mathbf{A}^*$ is then computed. From this eigendecomposition we obtain a set of orthonormal eigenvectors $\boldsymbol{\Psi} = \{\psi_0, \dots, \psi_{N-1}\}$ ranked with their non-negative eigenvalues $\boldsymbol{\sigma} = \{\sigma_0, \dots, \sigma_{N-1}\}$. A set of those eigenvalues $S_\delta = \{i : \sigma_i > \delta\sigma_0\}$, where $\delta > 0$, is considered as the basis in which we project and subsequently extend the function f. The projection of f in this truncated step is given as

$$f \mapsto P_\delta f = \sum_{i \in \Sigma_\delta} \langle f, \psi_i \rangle \psi_i \tag{30}$$

where $\langle \cdot, \cdot \rangle$ denotes the inner product. For $\boldsymbol{\phi}_{new} \notin \boldsymbol{\Phi}$ we obtain $(Ef)(\boldsymbol{\phi}_{new})$ by firstly extending each eigenvector $\psi_i \in \boldsymbol{\Psi}$,

$$\Psi_i(\boldsymbol{\phi}_{new}) = \sigma_i^{-1} \sum_{j=1}^{m} A(\boldsymbol{\phi}_{new}, \boldsymbol{\phi}_j)\psi_i(\phi_j), \tag{31}$$

where σ_i is the i^{th} eigenvalue and $\psi_i(\phi_j)$ is the j^{th} component of the eigenvector $\boldsymbol{\psi}_i$. The extended eigenvectors can then used to estimate $(Ef)(\boldsymbol{\phi}_{new})$ as,

$$(Ef)(\boldsymbol{\phi}_{new}) = \sum_{i \in S_\delta} \langle f, \psi_i \rangle \Psi_i(\phi_{new}). \tag{32}$$

A.7. Inverse function theorem

Consider the vector function $F(\boldsymbol{x}) = \boldsymbol{y}$ and assume that $\boldsymbol{x} \in \mathbb{R}^n$ is a solution of F and that $F : \mathbb{R}^n \to \mathbb{R}^n$ is differentiable. The Inverse Function Theorem [45] states that, if the Jacobian matrix

$$\mathbf{J}_f(\boldsymbol{x}) = \begin{pmatrix} \frac{\partial f_1}{\partial x_1} & \dots & \frac{\partial f_1}{\partial x_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial f_n}{\partial x_1} & \dots & \frac{\partial f_n}{\partial x_n} \end{pmatrix} \tag{33}$$

is invertible, then in a neighborhood of $\boldsymbol{x}$ and $\boldsymbol{y}$ the function f^{-1} exists. This suggests a unique *local* solution close to any $\boldsymbol{y}$. The Jacobian matrix is invertible if and only if its determinant is nonzero, therefore, showing that the $\det(\mathbf{J}_f(\boldsymbol{x}))$ has values of a single sign guarantees that the mapping is locally invertible and thus one-to-one.

A.8. Additional results

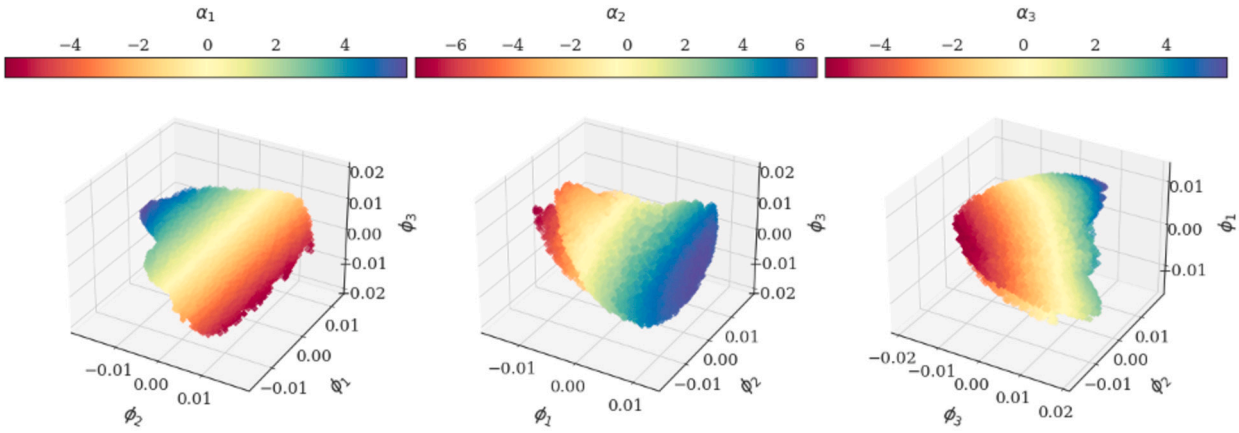

Fig. 20. Diffusion Maps coordinates the parametrize the latent space: ϕ_1, ϕ_2 and ϕ_3; Left: colored by sine coefficient α_1, center: colored by sine coefficient α_2, right: colored by sine coefficient α_3.

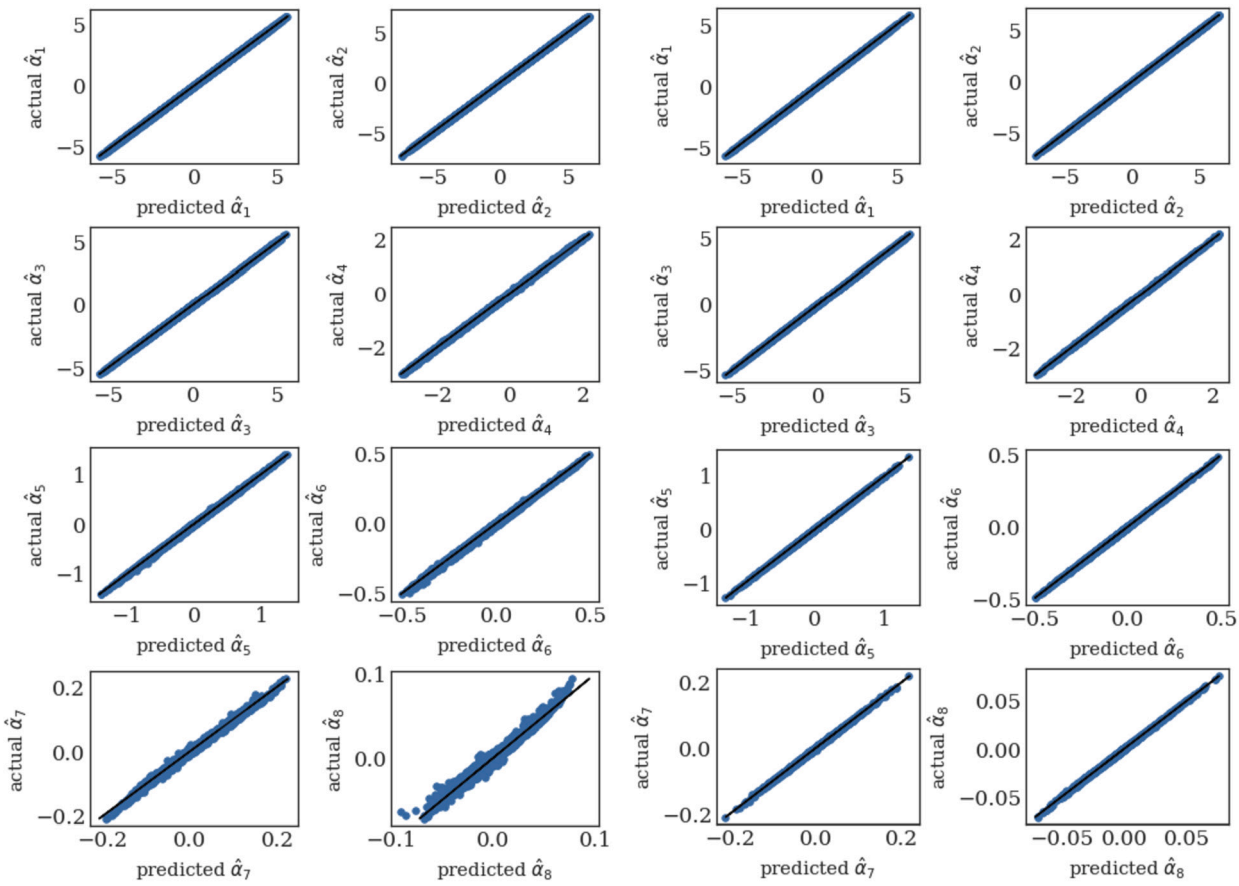

Fig. 21. Reconstruction of sine coefficients; first two columns: by the autoencoder, last two columns: by Double DMAPs.

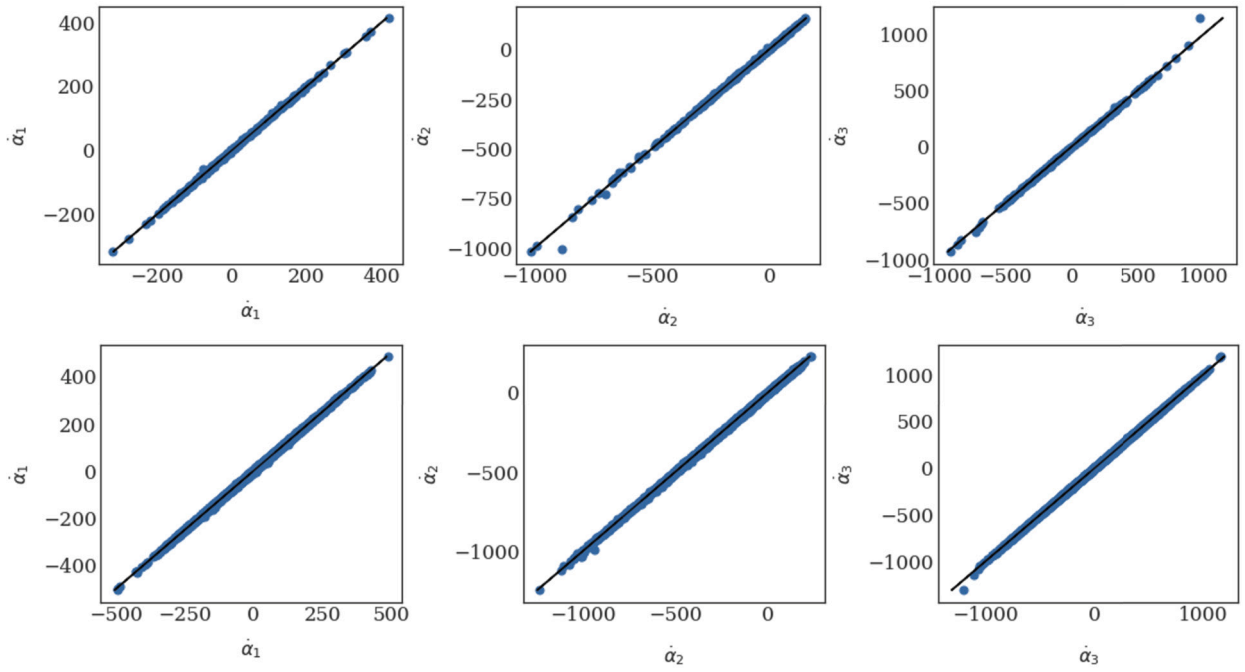

Fig. 22. Performance of the neural network predicting the right-hand-side of the learned ODE of the first three coefficients. Actual versus learned $\dot{\alpha}_1$ (left), $\dot{\alpha}_2$ (center) and $\dot{\alpha}_3$ (right), from the autoencoder (top) and Double DMAPs (bottom) derived values of sine coefficients.

References

[1] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dandelion Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, Xiaoqiang Zheng TensorFlow, Large-scale machine learning on heterogeneous systems, Software available from tensorflow.org, https://www.tensorflow.org/, 2015.

[2] A. Adrover, G. Continillo, S. Crescitelli, M. Giona, L. Russo, Construction of approximate inertial manifold by decimation of collocation equations of distributed parameter systems, Comput. Chem. Eng. 26 (1) (2002) 113–123.

[3] Maryam Akram, Malik Hassanaly, Venkat Raman, A priori analysis of reduced description of dynamical systems using approximate inertial manifolds, J. Comput. Phys. 409 (2020) 109344.

[4] S.V. Alekseenko, V.E. Nakoryakov, B.G. Pokusaev, Wave formation on vertical falling liquid films, Int. J. Multiph. Flow 11 (5) (1985) 607–627.

[5] Rushil Anirudh, Jayaraman J. Thiagarajan, Peer-Timo Bremer, Brian K. Spears, Improved surrogates in inertial confinement fusion with manifold and cycle consistencies, Proc. Natl. Acad. Sci. 117 (18) (2020) 9741–9746.

[6] Yohai Bar-Sinai, Stephan Hoyer, Jason Hickey, Michael P. Brenner, Learning data-driven discretizations for partial differential equations, Proc. Natl. Acad. Sci. 116 (31) (2019) 15344–15349.

[7] Peter Benner, Serkan Gugercin, Karen Willcox, A survey of projection-based model reduction methods for parametric dynamical systems, SIAM Rev. 57 (4) (2015) 483–531.

[8] Steven L. Brunton, Joshua L. Proctor, J. Nathan Kutz, Discovering governing equations from data by sparse identification of nonlinear dynamical systems, Proc. Natl. Acad. Sci. 113 (15) (2016) 3932–3937.

[9] Hsueh-Chia Chang, Nonlinear waves on liquid film surfaces—i. Flooding in a vertical tube, Chem. Eng. Sci. 41 (10) (1986) 2463–2476.

[10] Hsueh-Chia Chang, Traveling waves on fluid interfaces: normal form analysis of the Kuramoto–Sivashinsky equation, Phys. Rev. A 29 (10) (1986) 3142–3147.

[11] Alexandre J. Chorin, Fei Lu, Discrete approach to stochastic parametrization and dimension reduction in nonlinear dynamics, Proc. Natl. Acad. Sci. 112 (32) (2015) 9804–9809.

[12] Ronald R. Coifman, Stéphane Lafon, Geometric harmonics: a novel tool for multiscale out-of-sample extension of empirical functions, Applied and Computational Harmonic Analysis 21 (1) (2006) 31–52.

[13] Ronald R. Coifman, Stéphane Lafon, Diffusion maps, Appl. Comput. Harmon. Anal. 21 (1) (2006) 5–30.

[14] Ronald R. Coifman, Ioannis G. Kevrekidis, Stéphane Lafon, Mauro Maggioni, Boaz Nadler, Diffusion maps, reduction coordinates, and low dimensional representation of stochastic systems, Multiscale Model. Simul. 7 (2) (2008) 842–864.

[15] Carlos E. Pérez De Jesús, Michael D. Graham, Data-driven low-dimensional dynamic model of Kolmogorov flow, Phys. Rev. Fluids 8 (4) (2023) 044402.

[16] Carmeline J. Dsilva, Ronen Talmon, Ronald R. Coifman, Ioannis G. Kevrekidis, Parsimonious representation of nonlinear dynamical systems through manifold learning: a Chemotaxis case study, Appl. Comput. Harmon. Anal. 44 (3) (2018) 759–773.

[17] Nikolaos Evangelou, Felix Dietrich, Eliodoro Chiavazzo, Daniel Lehmberg, Marina Meila, Ioannis G. Kevrekidis, Double diffusion maps and their latent harmonics for scientific computations in latent space, J. Comput. Phys. 485 (2023) 112072.

[18] C. Foias, M.S. Jolly, I.G. Kevrekidis, George R. Sell, E.S. Titi, On the computation of inertial manifolds, Phys. Lett. A 131 (7–8) (1988) 433–436.

[19] Ciprian Foias, George R. Sell, Roger Temam, Inertial manifolds for nonlinear evolutionary equations, J. Differ. Equ. 73 (2) (1988) 309–353.

[20] Ciprian Foias, George R. Sell, Edriss S. Titi, Exponential tracking and approximation of inertial manifolds for dissipative nonlinear equations, J. Dyn. Differ. Equ. 1 (1989) 199–244.

[21] Bosco García-Archilla, Edriss S. Titi, Postprocessing the Galerkin method: the finite-element case, SIAM J. Numer. Anal. 37 (2) (1999) 470–499.

[22] Bosco García-Archilla, Julia Novo, Edriss S. Titi, Postprocessing the Galerkin method: a novel approach to approximate inertial manifolds, SIAM J. Numer. Anal. 35 (3) (1998) 941–972.

[23] Bosco García-Archilla, Julia Novo, Edriss Titi, An approximate inertial manifolds approach to postprocessing the Galerkin method for the Navier-Stokes equations, Math. Comput. 68 (227) (1999) 893–911.
[24] Charles William Gear, I.G. Kevrekidis, B.E. Sonday, Slow Manifold Integration on a Diffusion Map Parameterization, AIP Conference Proceedings, vol. 1389, American Institute of Physics, 2011, pp. 13–16.
[25] Rudy Geelen, Stephen Wright, Karen Willcox, Operator inference for non-intrusive model reduction with quadratic manifolds, Comput. Methods Appl. Mech. Eng. 403 (2023) 115717.
[26] J-L. Guermond, Serge Prudhomme, A fully discrete nonlinear Galerkin method for the 3D Navier–Stokes equations, Numer. Methods Partial Differ. Equ. 24 (3) (2008) 759–775.
[27] F. Jauberteau, C. Rosier, R. Temam, A nonlinear Galerkin method for the Navier-Stokes equations, Comput. Methods Appl. Mech. Eng. 80 (1–3) (1990) 245–260.
[28] Michael S. Jolly, Explicit construction of an inertial manifold for a reaction diffusion equation, J. Differ. Equ. 78 (2) (1989) 220–261.
[29] Michael S. Jolly, I.G. Kevrekidis, Edriss S. Titi, Approximate inertial manifolds for the Kuramoto-Sivashinsky equation: analysis and computations, Phys. D, Nonlinear Phenom. 44 (1–2) (1990) 38–60.
[30] M.S. Jolly, I.G. Kevrekidis, E.S. Titi, Preserving dissipation in approximate inertial forms for the Kuramoto-Sivashinsky equation, J. Dyn. Differ. Equ. 3 (1991) 179–197.
[31] Wei Kang, Jia-Zhong Zhang, Sheng Ren, Peng-Fei Lei, Nonlinear Galerkin method for low-dimensional modeling of fluid dynamic system using pod modes, Commun. Nonlinear Sci. Numer. Simul. 22 (1–3) (2015) 943–952.
[32] Ioannis G. Kevrekidis, Basil Nicolaenko, James C. Scovel, Back in the saddle again: a computer assisted study of the Kuramoto-Sivashinsky equation, SIAM J. Appl. Math. 50 (3) (1990) 760–790, ISSN 00361399, http://www.jstor.org/stable/2101886.
[67] Eleni D. Koronaki, Georgios P. Gakis, Nikolaos Cheimarios, Andreas G. Boudouvis, Efficient tracing and stability analysis of multiple stationary and periodic states with exploitation of commercial CFD software, Chem. Eng. Sci. 150 (2016) 26–34.
[34] Mark A. Kramer, Nonlinear principal component analysis using autoassociative neural networks, AIChE J. 37 (2) (1991) 233–243.
[35] K. Krischer, R. Rico-Martínez, I.G. Kevrekidis, H.H. Rotermund, G. Ertl, J.L. Hudson, Model identification of a spatiotemporally varying catalytic reaction, AIChE J. 39 (1) (1993) 89–98, https://doi.org/10.1002/aic.690390110.
[36] Yoshiki Kuramoto, Toshio Tsuzuki, Persistent propagation of concentration waves in dissipative media far from thermal equilibrium, Prog. Theor. Phys. 55 (2) (1976) 356–369.
[37] Kookjin Lee, Kevin T. Carlberg, Model reduction of dynamical systems on nonlinear manifolds using deep convolutional autoencoders, J. Comput. Phys. 404 (2020) 108973.
[38] Daniel Lehmberg, Felix Dietrich, Gerta Köster, Hans-Joachim Bungartz, Datafold: data-driven models for point clouds and time series on manifolds, J. Open Sour. Softw. 5 (51) (2020) 2283, https://doi.org/10.21105/joss.02283.
[39] Alec J. Linot, Michael D. Graham, Deep learning to discover and predict dynamics on an inertial manifold, Phys. Rev. E 101 (6) (2020) 062209.
[40] Alec J. Linot, Michael D. Graham, Data-driven reduced-order modeling of spatiotemporal chaos with neural ordinary differential equations, Chaos, Interdiscip. J. Nonlinear Sci. 32 (7) (2022) 073110.
[41] Alec J. Linot, Kevin Zeng, Michael D. Graham, Turbulence control in plane Couette flow using low-dimensional neural ode-based models and deep reinforcement learning, Int. J. Heat Fluid Flow 101 (2023) 109139.
[42] Fei Lu, Kevin K. Lin, Alexandre J. Chorin, Data-based stochastic model reduction for the Kuramoto–Sivashinsky equation, Phys. D, Nonlinear Phenom. 340 (2017) 46–57.
[43] Len G. Margolin, Edriss S. Titi, Shannon Wynne, The postprocessing Galerkin and nonlinear Galerkin methods—a truncation analysis point of view, SIAM J. Numer. Anal. 41 (2) (2003) 695–714.
[44] Marion Martine, R. Temam, Nonlinear Galerkin methods: the finite elements case, Numer. Math. 57 (1990) 205–226.
[45] Jerrold E. Marsden, Michael J. Hoffman, et al., Elementary Classical Analysis, Macmillan, 1993.
[46] Cristina P. Martin-Linares, Yorgos M. Psarellis, Georgios Karapetsas, Eleni D. Koronaki, Ioannis G. Kevrekidis, Physics-agnostic and physics-infused machine learning for thin films flows: modeling, and predictions from small data, arXiv preprint, arXiv:2301.12508, 2023.
[47] Shane A. McQuarrie, Cheng Huang, Karen E. Willcox, Data-driven reduced-order models via regularised operator inference for a single-injector combustion process, J. R. Soc. N. Z. 51 (2) (2021) 194–211.
[48] Boaz Nadler, Stéphane Lafon, Ronald R. Coifman, Ioannis G. Kevrekidis, Diffusion maps, spectral clustering and reaction coordinates of dynamical systems, Appl. Comput. Harmon. Anal. 21 (1) (2006) 113–127.
[49] Elizabeth Qian, Ionut-Gabriel Farcas, Karen Willcox, Reduced operator inference for nonlinear partial differential equations, SIAM J. Sci. Comput. 44 (4) (2022) A1934–A1959.
[50] Maziar Raissi, Paris Perdikaris, George E. Karniadakis, Physics-informed neural networks: a deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations, J. Comput. Phys. 378 (2019) 686–707.
[51] Ramiro Rico-Martinez, K. Krischer, I.G. Kevrekidis, M.C. Kube, J.L. Hudson, Discrete-vs. continuous-time nonlinear signal processing of cu electrodissolution data, Chem. Eng. Commun. 118 (1) (1992) 25–48.
[52] Shan Shan, Ingrid Daubechies, Diffusion maps: using the semigroup property for parameter tuning, in: Theoretical Physics, Wavelets, Analysis, Genomics: An Indisciplinary Tribute to Alex Grossmann, Springer, 2022, pp. 409–424.
[53] Jie Shen, Long time stability and convergence for fully discrete nonlinear Galerkin methods, Appl. Anal. 38 (4) (1990) 201–229.
[54] Stanislav Y. Shvartsman, Ioannis G. Kevrekidis, Nonlinear model reduction for control of distributed systems: a computer-assisted study, AIChE J. 44 (7) (1998) 1579–1595.
[55] Gregory I. Sivashinsky, Nonlinear analysis of hydrodynamic instability in laminar flames—i. Derivation of basic equations, Acta Astronaut. 4 (11) (1977) 1177–1206.
[56] Benjamin Sonday, Systematic Model Reduction for Complex Systems Through Data Mining and Dimensionality Reduction, Princeton University, 2011.
[57] Benjamin E. Sonday, Amit Singer, C. William Gear, Ioannis G. Kevrekidis, Manifold learning techniques and model reduction applied to dissipative pdes, arXiv preprint, arXiv:1011.5197, 2010.
[58] R. Temam, Do inertial manifolds apply to turbulence?, Phys. D, Nonlinear Phenom. 37 (1–3) (1989) 146–152.
[59] Roger Temam, Induced trajectories and approximate inertial manifolds, ESAIM: Math. Model. Numer. Anal. 23 (3) (1989) 541–561.
[60] C. Theodoropoulos, I.G. Kevrekidis, T.J. Mountziaris, Order reduction for nonlinear dynamic models of distributed reacting systems, J. Process Control 10 (2–3) (2000) 177–184.
[61] Edriss S. Titi, On approximate inertial manifolds to the Navier-Stokes equations, J. Math. Anal. Appl. 149 (2) (1990) 540–557.
[62] Pascal Vincent, Hugo Larochelle, Yoshua Bengio, Pierre-Antoine Manzagol, Extracting and composing robust features with denoising autoencoders, in: Proceedings of the 25th International Conference on Machine Learning, 2008, pp. 1096–1103.
[63] Lars Wahlbin, Superconvergence in Galerkin Finite Element Methods, Springer, 2006.
[64] Benjamin G. Zastrow, Anirban Chaudhuri, Karen E. Willcox, Anthony S. Ashley, Michael C. Henson, Data-driven model reduction via operator inference for coupled aeroelastic flutter, in: AIAA SCITECH 2023 Forum, 2023, p. 0330.
[65] Kevin Zeng, Michael D. Graham, Autoencoders for discovering manifold dimension and coordinates in data from complex dynamical systems, 2023.
[66] Kevin Zeng, Alec J. Linot, Michael D. Graham, Data-driven control of spatiotemporal chaos with reduced-order neural ode-based models and reinforcement learning, Proc. R. Soc. A 478 (2267) (2022) 20220297.