Physics-Informed Koopman Network

Yuying Liu[†] Aleksei Sholokhov[†] Hassan Mansour[§] Saleh Nabi[§]

[†] Department of Applied Mathematics, University of Washington, Seattle, WA 98105
[§] Mitsubishi Electric Research Laboratories, Cambridge, MA 02139

Abstract

Koopman operator theory is receiving increased attention due to its promise to linearize nonlinear dynamics. Neural networks that are developed to represent Koopman operators have shown great success thanks to their ability to approximate arbitrarily complex functions. However, despite their great potential, they typically require large training data-sets either from measurements of a real system or from high-fidelity simulations. In this work, we propose a novel architecture inspired by physics-informed neural networks, which leverage automatic differentiation to impose the underlying physical laws via soft penalty constraints during model training. We demonstrate that it not only reduces the need of large training data-sets, but also maintains high effectiveness in approximating Koopman eigenfunctions.

1 Introduction

Nonlinear dynamical systems give rise to a rich diversity of complex phenomena. Dealing with nonlinearity is the central task of many areas of science and engineering such as climate science [46], neuroscience [13], ecology [16], finance [50], and epidemiology [66]. A classic yet popular view of dynamical systems is geometric, where the behavior of multiple trajectories can be simultaneously studied in a qualitative manner. However, the geometry in state space becomes more complex when the dynamics become nonlinear and high dimensional, making the system hard to analyze, predict and control [54].

In 1931, Koopman introduced the operator-theoretic perspective of dynamical systems [36], complementing the traditional geometric perspectives. In this framework, a Koopman operator is defined which acts on observation functions (observables) in an appropriate function space. Under the action of this operator, the evolution of the observables are linear although the function space may be infinite-dimensional. As a consequence, approximating the Koopman operator and seeking its eigenfunctions become a key to linearize the nonlinear dynamics [52, 53, 15].

The leading computational method for approximating the Koopman operator is dynamic mode decomposition (DMD) [69]. Using the snapshots of state measurements of a system, the DMD algorithm seeks the best linear operator that approximately advances these states. There are many variants of DMD, however, most of them require an a priori, judicious selection of the observables, and there is no guarantee that these observables span an invariant Koopman subspace [40]. Another limitation is that a typical application of DMD starts from a single trajectory and thererfore the

approximation of the Koopman operator is only restricted to these measurements, leading to failures of correctly predicting unseen trajectories for non-ergodic systems [2].

To address these challenges, neural networks were proposed to approximate the Koopman operator. In terms of architecture, autoencoder is the most commonly used building block. Furthermore, a linearity constraint is added to the loss function to approximate the Koopman operator [74, 58, 48, 24]. The neural network approach generalizes well mainly due to three reasons: (i) it automatically selects the observables [74], (ii) it can approximate arbitrarily complex functions [30], and (iii) trained with multiple trajectories, leading to a good interpolation of the dynamics. A successfully trained neural network can identify coordinate transformations that make strongly nonlinear dynamics approximately linear and, therefore, enabling linear prediction, estimation, and control. However, one would need to acquire a large enough data-set to successfully train a neural network which is neither efficient nor practical.

On another line of research, physics-informed neural networks (PINNs)[67] were introduced in 2019. They can seamlessly integrate the measurement data and physical governing laws by penalizing the residuals of the differential equation in the loss function using automatic differentiation. This approach alleviates the need for a large amount of data by assimilating the knowledge of the equations into the training process. However, (i) PINNs can only solve one solution instance at a time, and (ii) the solution is not accurate outside the training time horizon.

In this work, we propose physics-informed Koopman networks (PIKNs) which combines the strengths of both PINNs and autoencoder-based Koopman networks. More specifically, by incorporating the knowledge of dynamics, we reduce the need for large training data-sets for identifying Koopman eigenvalues and eigenfunctions. Moreover, since the network performs (Koopman) operator learning, it enables the model to predict beyond the training horizon, and also to be used for compressed sensing, estimation, and control.

The paper is organized as follows. In Section 2, we briefly introduce Koopman operator theory and our methodology. Then we mention some related works in Section 3 and highlight the connections and differences. Our approach is then tested on several benchmark problems in Section 4. In Section 5, we conclude and discuss future directions.

2 Method

2.1 Koopman operator theory

We begin with introducing an autonomous ordinary differential equation

$$\frac{d}{dt}\mathbf{x}(t) = \mathbf{f}(\mathbf{x}(t)) \tag{1}$$

with $\mathbf{x} \in \mathcal{X} \subseteq \mathbb{R}^n$. We define the time-$t$ flow map operator $\mathbf{F}^t : \mathcal{X} \to \mathcal{X}$ as

$$\mathbf{x}(t_0 + t) = \mathbf{F}^t(\mathbf{x}(t_0)) \tag{2}$$

In 1931, B.O.Koopman [36] provided an alternative description for dynamical systems in terms of evolution of functions of possible measurements $\mathbf{y} = g(\mathbf{x})$. The function $g : \mathcal{X} \to \mathbb{C}$ is called a measurement function and it belongs to some set of functions $\mathcal{G}(\mathcal{X})$. This set is often not defined *a priori*, and Hilbert spaces such as $L^2(\mathcal{X}, d\mu)$ or reproducing kernel Hilbert spaces (RKHS) are common choices [18, 54]. In all cases, however, $\mathcal{G}(\mathcal{X})$ is of significantly higher dimension than $\mathcal{X}$,

so we are trading dimensionality for linearity.

The family of Koopman operators $\mathcal{K}^t : \mathcal{G}(\mathcal{X}) \rightarrow \mathcal{G}(\mathcal{X})$, parameterized by t are given by

$$\mathcal{K}^t g(\mathbf{x}) = g(\mathbf{F}^t(\mathbf{x})) \tag{3}$$

One can check that $\mathcal{K}^t$ is linear. In general it is infinite-dimensional, and constructing finite-dimensional representations of Koopman operator remains an open question.

If the dynamics in Equation 1 is sufficiently smooth, one can also define the infinitesimal generator $\mathcal{L}$ of the Koopman operator family as

$$\mathcal{L}g := \lim_{t \to 0} \frac{\mathcal{K}^t g - g}{t} = \lim_{t \to 0} \frac{g \circ \mathbf{F}^t - g}{t} \tag{4}$$

From the definition, we can easily see

$$\mathcal{L}g(\mathbf{x}(t)) = \lim_{\tau \to 0} \frac{g(\mathbf{x}(t + \tau)) - g(\mathbf{x}(t))}{\tau} = \frac{d}{dt} g(\mathbf{x}(t)) \tag{5}$$

The generator $\mathcal{L}$ is sometimes referred to as the Lie operator: it is the Lie derivative of g along the vector field $\mathbf{f}(\mathbf{x})$ when the dynamics is given by Equation 1. On the other hand, we also have

$$\frac{d}{dt} g(\mathbf{x}(t)) = \nabla g \cdot \frac{d}{dt} \mathbf{x}(t) = \nabla g \cdot \mathbf{f}(\mathbf{x}(t)) \tag{6}$$

Therefore, we conclude

$$\mathcal{L}g = \nabla g \cdot \mathbf{f} \tag{7}$$

Equation 7 will be the key for the implementation of PIKN.

Applied Koopman analysis seeks key measurement functions that behave linearly in time, and the eigenfunctions of the Koopman operator are functions that exhibit such behaviour [15, 55]. A Koopman eigenfunction $\varphi(\mathbf{x})$ corresponding to an eigenvalue λ^t satisfies

$$\mathcal{K}^t \varphi(\mathbf{x}) = \lambda^t \varphi(\mathbf{x}) \tag{8}$$

It is straightforward to show that Koopman eigenfunctions $\varphi(\mathbf{x})$ that satisfies Equation 8 for $\lambda^t \neq 0$ are also eigenfunctions of the Lie operator, although with a different eigenvalue $\mu = \log(\lambda^t)/t$.

Once we have a set of eigenfunctions $\{\varphi_1, \varphi_2, \cdots, \varphi_M\}$, observables that can be formed as a linear combination of these eigenfunctions, i.e., $g \in span\{\varphi_k\}_{k=1}^M$ have a particularly simple evolution under the Koopman operator

$$g(\mathbf{x}) = \sum_{k=1}^M c_k \varphi_k(\mathbf{x}) \implies \mathcal{K}^t g(\mathbf{x}) = \sum_{k=1}^M c_k \lambda_k^t \varphi_k(\mathbf{x}) \tag{9}$$

This also implies $span\{\varphi_k\}_{k=1}^M$ is an invariant subspace under the Koopman operator $\mathcal{K}^t$ and can be viewed as the new coordinates on which the dynamics evolve linearly.

Although the promise of Koopman operator theory is tempting, there are certain challenges. For example,

- The Koopman operator of a system may not admit a discrete spectrum. Certain systems fundamentally fail to fit into this framework [48, 37].

3

- Asymptotic methods can be used to approximate certain eigenfunctions for simple dynamics (e.g., polynomial nonlinear dynamics), however, there is no analytical procedure to seek for the eigen-pairs of Koopman operator in general.

- Some computational methods (e.g., DMD) can be used to approximate eigenfunctions of Koopman operator, but the approximation is only restricted to those measurements leading to spurious modes. Moreover, the identified basis may not span a Koopman invariant subspace.

Since our ultimate goal is to study nonlinear dynamical systems using linear theory, we do not need to restrict ourselves to Equation 9. Following [48, 24], we can generalize it as

$$g(\mathbf{x}) = \psi(\varphi_1(\mathbf{x}), \varphi_2(\mathbf{x}), \dots, \varphi_M(\mathbf{x}); \omega)$$
$$\Downarrow$$
$$\mathcal{K}^t g(\mathbf{x}) = \psi(\lambda_1^t \varphi_1(\mathbf{x}), \lambda_2^t \varphi_2(\mathbf{x}), \dots, \lambda_M^t \varphi_M(\mathbf{x}); \omega)$$

(10)

where ψ is an arbitrary transformation parameterized by ω.

2.2 Auto-encoder based architecture

An auto-encoder is a special type of neural network which is particularly suitable for our application and widely used in the literature [48, 24]. The encoder ϕ learns the representation of the relevant Koopman eigenfunctions, which provides intrinsic coordinates that linearize the dynamics, and the decoder ψ seeks an inverse transformation to reconstruct the original measurements. One may hope that if we define $\phi : \mathbf{x} \to (\varphi_1(\mathbf{x}), \varphi_2(\mathbf{x}), \dots, \varphi_M(\mathbf{x}))^T$, then up to a constant, the encoder learns this transformation ϕ and the decoder learns the ψ as shown in Equation 10. (Alternatively, if we specify a linear decoder, then the learning regime would correspond to Equation 9.)

Within the auto-encoder's latent space, the dynamics is constrained to be linear. Therefore in previous works, a squared matrix K is often used to drive the evolution of the dynamics. Most often there is no invariant, finite-dimensional Koopman subspace that captures the evolution of all the measurements, then that matrix K will only be an approximation of the true underlying linear operator.

There are different ways to train the auto-encoder architecture in the literature [74, 58, 48, 24, 4], however, they all require a large amount of measurement data. Normally, the training data-set X is arranged as a 3D tensor, with its dimensions to be (i) number of sequences (with different initial states), (ii) number of snapshots, and (iii) dimensionality of the measurements, respectively. Then the constraint of linear dynamics can be enforced by a loss term resembling $\|\phi(\mathbf{x}_{n+1}) - K\phi(\mathbf{x}_n)\|$, or more generally, linearity is enforced over multiple steps $\|\phi(\mathbf{x}_{n+p}) - K^p\phi(\mathbf{x}_n)\|$, generating recurrencies in the neural network architecture. We will see, however, such large data-sets are not necessary (but beneficial) for PIKN.

2.3 Physics-informed Koopman network

In physical sciences, data is scarce while governing equations are available in literature. In physics-informed Koopman networks (PIKNs), we aim to leverage such knowledge of the dynamics, e.g., of Equation 7, to enforce the linearity constraint. The basic idea is to train the network by minimizing the quantity $\|\nabla\varphi_k(\mathbf{x}) \cdot \mathbf{f} - \mu_k\varphi_k(\mathbf{x})\|, \quad \forall k = 1, 2, 3, \cdots, M$. More generally, a squared matrix L is used to approximate the Lie operator $\mathcal{L}$, which in turn is related to the Koopman operator, and we

minimize $\|L\phi(\mathbf{x}) - \nabla\phi(\mathbf{x}) \cdot \mathbf{f}\|$. Finding the eigenvalue and eigenfunction pairs of the Lie operator corresponds to performing eigendecomposition to the matrix L.

2.3.1 For ODE

We first consider an ordinary differential equation of the form in Equation 1. We start from sampling a set of collocation points $X := \{\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_N\}$. This set of collocation points does not need to come from any trajectories of the dynamics but they can be sampled randomly, avoiding a bulk of simulations or measurement data collections.

The objective of the network is to identify a few key coordinates $\mathbf{z} = \phi(\mathbf{x})$ spannned by a set of Koopman eigenfunctions $\varphi_k(\mathbf{x}) : \mathcal{R}^n \to \mathcal{R}, \; k = 1, 2, \cdots, M$ along with a dynamical system $\dot{\mathbf{z}} = L\mathbf{z}$. Objective function is

$$\mathcal{J}_{linear} = \frac{1}{N} \sum_{i=1}^{N} (\omega_1 \| L\phi(\mathbf{x}_i) - \nabla\phi(\mathbf{x}_i) \cdot \mathbf{f}(\mathbf{x}_i)\|^2 + \omega_2 \|\mathbf{x}_i - C\mathbf{z}_i\|^2) \tag{11}$$

if the decoder is linear (where C represents the reconstruction coefficients), or for generic decoder

$$\mathcal{J}_{nonlinear} = \frac{1}{N} \sum_{i=1}^{N} (\omega_1 \| L\phi(\mathbf{x}_i) - \nabla\phi(\mathbf{x}_i) \cdot \mathbf{f}(\mathbf{x}_i)\|^2 + \omega_2 \|\mathbf{x}_i - \psi(\mathbf{z_i})\|^2) \tag{12}$$

Here, ω_1 and ω_2 are the weights for each loss term. The first term encourages linear dynamics within the latent space and the second term makes it a valid auto-encoder. One can further diagonalize L such that the diagonal elements approximate the Koopman eigenvalues and the corresponding outputs of the encoder approximate the Koopman eigenfunctions, respectively. But this constraint is not necessary as it is equivalent to performing eigendecomposition of a general-structured L after training. Once trained, it can be used for state predictions beyond the time horizon used in training.

2.3.2 For PDE

For application to partial differential equations of the form

$$\mathbf{u}_t = \mathbf{f}(\mathbf{u}, \mathbf{u_x}, \dots) \tag{13}$$

similar to ODEs, the goal is to seek coordinates $\mathbf{v} = \phi(\mathbf{u})$ that has linear evolution $\dot{\mathbf{v}} = L\mathbf{v}$ and can be used to reconstruct the original measurements $\hat{\mathbf{u}} = \psi(\mathbf{v})$. The main difference is, however, the input and output of the network are functions of spatio-temporal variables $\mathbf{u}(\mathbf{x}, \mathbf{t})$ instead of the temporal variables $\mathbf{x}(t)$ as in the ODE cases. Therefore, we need to sample a set of "collocation points" in an appropriate function space where $\mathbf{u}_t$ can be cheaply evaluated. We provide examples of such suitable function families in the following chapters.

2.4 Data integration

Like PINNs, we can seamlessly integrate information from measurement data. Suppose we have snapshots of measurements $X_{data} := \{\mathbf{x}(t_0), \mathbf{x}(t_1), \cdots, \mathbf{x}(t_p)\}$ for an ODE system or $U_{data} :=$

$\{\mathbf{u}(\mathbf{x}, t_0), \mathbf{u}(\mathbf{x}, t_1), \cdots, \mathbf{u}(\mathbf{x}, t_p)\}$ for a PDE, by adding extra loss terms

$$\mathcal{J}_{data} = \frac{1}{p} \sum_{j=0}^{p} (\omega_3 \|e^{L\Delta t_j} \phi(\mathbf{x}(t_0)) - \phi(\mathbf{x}(t_j))\|^2 + \omega_4 \|\mathbf{x}(t_j) - \psi(\mathbf{z}(t_j))\|^2) \quad \text{(for ODE)}$$

$$\mathcal{J}_{data} = \frac{1}{p} \sum_{j=0}^{p} (\omega_3 \|e^{L\Delta t_j} \phi(\mathbf{u}(\mathbf{x}, t_0)) - \phi(\mathbf{u}(\mathbf{x}, t_j))\|^2 + \omega_4 \|\mathbf{u}(\mathbf{x}, t_j) - \psi(\mathbf{v}(\mathbf{u}(\mathbf{x}, t_j)))\|^2) \quad \text{(for PDE)}$$

$$(14)$$

we can penalize the network predictions that do not match the real measurements, where $\Delta t_j = t_j - t_0, \quad \forall j = 1, 2, \cdots, p$. Again, the first term is the linearity loss and the second term is the reconstruction loss. It should be noted that Eq. 14 is consistent with previous literature [74, 58, 48] on using autoencoders to find approximation of Koopman eigenfunctions and can be seen as a baseline on how physics-informed loss improve the performance of PIKN.

3 Related Work

3.1 Dynamic Mode Decomposition

Dynamic mode decomposition (DMD), which was originally introduced by Schmid[70], is the leading computational method to approximate the Koopman operator from data[40, 77]. Rowley et al. were the first to establish the connection between DMD and Koopman operator theory[69]. One of the major advantages of DMD is its simple formulation in terms of linear regression. For this reason, many methodological innovations have been introduced, for example, Jovanovic et al.[33] uses sparsity promoting optimzation to identify the fewest DMD modes; [12, 21] accelerate DMD using randomized linear algebra; extended DMD[81] suggests to include nonlinear measurements; higher order DMD[43] acts on delayed coordinates and generates more complex models; multiresolution DMD[41] deals with multiscale systems that exhibit transient or intermittent dynamics; Proctor et al.[65] extended the algorithm to disambiguate the natural dynamics and actuation; algorithms include total least-squares DMD[27], forward-backward DMD[19] and variable projection[3] improve the performance of DMD over noise sensitivity. These methods are now widely applied to many fields in science and engineering such as fluid dynamics and heat trasnfer[5, 7, 8, 57, 34, 39], epidemiology[66], neuroscience[13], finance[50], plasma physics[75], robotics[9, 10] and video processing[25, 20, 11]. For a more comprehensive review, one can refer to [14].

3.2 Deep Learning for Linear Embedding

Hand-crafted basis functions or measurements from DMD sometimes fail to represent the complex Koopman eigenfunctions. Neural networks turn out to be more effective in approximating them, leading to linear embedding of the nonlinear dynamics[74, 44, 48, 84, 4]. They have achieved great successes in long-term dynamic predictions[42], fluid control[58] and also recently be extended to account for uncertainties[59], modeling PDEs[24] and jointly used with optimal control[26, 1]. There are also innovations on the side of neural network architectures, for example, neural ODEs are used for dictionary learning[76] and graphical neural networks are used to learn compositional Koopman operators[45]. However, all the listed works are purely data-driven and do not address the issue of data efficiency.

3.3 Physics-Informed Neural Network

Physics-informed neural networks (PINNs) were first proposed in[68, 67] and have received lots of attention due to its flexibility to integrate measurement data and the physics (governing equations). They have been applied to various classes of PDEs[63, 23, 85] and extended to deal with uncertainties[83, 86, 87, 73, 82]. Another line of research of PINNs focus on its training and performance. For example, domain decomposition is considered in some variations of PINNs[35, 32, 31], leading to parallel implementations[72, 28]; multi-fidelity framework[51], dynamic weights of the loss function[78] and hard constraints[47] have also been thoroughly studied, in order to achieve stable training results. Theoretical insights into the convergence of PINNs are presented in[71, 56, 80]. Recently, PINNs have also been jointly used with DeepONets[79], entering the realm of operator learning. Similar to our work, the key motivation of using PINNs there is to eliminate the need of large data-sets for training DeepONets, and to achieve this we use automatic differentiation in evaluation of the loss function related to 7.

4 Experiments

4.1 Simple nonlinear system with discrete spectrum

First, we consider a simple nonlinear system with a single fixed point and a discrete eigenvalue spectrum:

$$
\begin{aligned}
\dot{x}_1 &= \mu x_1 \\
\dot{x}_2 &= \lambda(x_2 - x_1^2)
\end{aligned}
\tag{15}
$$

For $\lambda < \mu < 0$, the system exhibits a slow attracting manifold given by $x_2 = x_1^2$. In our experiment, we use $\mu = -0.1$ and $\lambda = -1$. This example is simple and widely adopted as a benchmark for Koopman/DMD related algorithms because it's possible to explicitly define a three-dimensional Koopman invariant subspace (spanned by Koopman eigenvalue and eigenfunction pairs) that contains the state variables x_1 and x_2:

$$
\frac{d}{dt}
\begin{bmatrix} \varphi_\mu \\ \varphi_{2\mu} \\ \varphi_\lambda \end{bmatrix}
=
\begin{bmatrix} \mu & 0 & 0 \\ 0 & 2\mu & 0 \\ 0 & 0 & \lambda \end{bmatrix}
\begin{bmatrix} \varphi_\mu \\ \varphi_{2\mu} \\ \varphi_\lambda \end{bmatrix}
$$
$$
\begin{bmatrix} x_1 \\ x_2 \end{bmatrix}
=
\begin{bmatrix} 1 & 0 & 0 \\ 0 & b & 1 \end{bmatrix}
\begin{bmatrix} \varphi_\mu \\ \varphi_{2\mu} \\ \varphi_\lambda \end{bmatrix}
\tag{16}
$$

where $\phi(\mathbf{x}) = [\varphi_\mu, \varphi_{2\mu}, \varphi_\lambda] = [x_1, x_1^2, x_2 - bx_1^2]$ and $b = \frac{\lambda}{\lambda - 2\mu}$. If nonlinear decoding is allowed, a two-dimensional Koopman invariant subspace spanned only by $\{\varphi_\mu, \varphi_\lambda\}$ is sufficient because

$$
\begin{aligned}
x_1 &= \varphi_\mu \\
x_2 &= \varphi_\lambda + b\varphi_\mu^2
\end{aligned}
\tag{17}
$$

Therefore, we should at least expect our network to find one three-dimensional Koopman invariant subspace with a linear decoder or a two-dimensional Koopman invariant subspace with a nonlinear decoder. In Fig 1, we show the results of trained PIKNs with and without a linear decoder in two
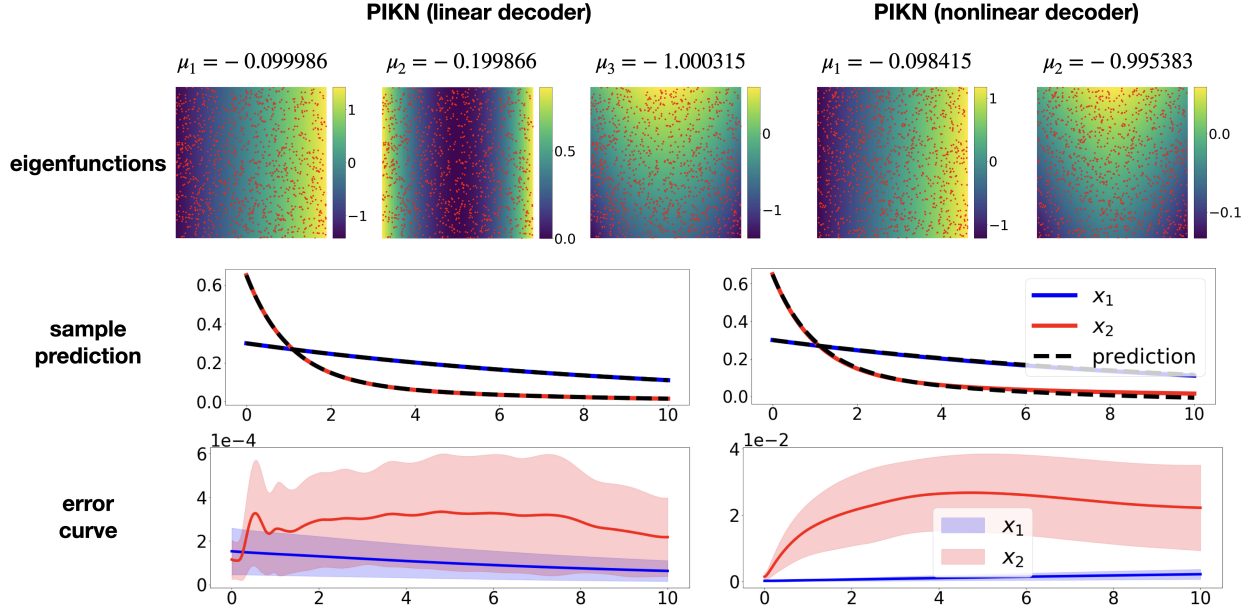
Figure 1: PIKN results using autoencoder with linear (left column) and nonlinear (right column) decoder network. The first row shows the eigenvalues and eigenfunctions identified by proposed PIKNs where the red dots represent the collocation points for training; the second row shows a 10000-step forward prediction of the dynamics starting from a initial state $(x_1, x_2) = (0.3, 0.65)$ and with $\Delta t = 0.001$; the third row visualizes the mean absolute error over 1000 different trajectories with initial conditions uniformly sampled from $[-1, 1] \times [-1, 1]$, the shaded region covers one standard deviation away from the mean error.

columns respectively. The networks have accurately identified the Koopman eigenvalues and eigenfunctions[1] within the corresponding invariant subspace. Both models exhibit accurate predictive performance. Note that these two networks are both trained in a purely physics-informed manner, indicating there is no need for simulation data. Specific details about the network architecture, training procedure and other extensive studies are provided in the Appendix A.1.1.

4.2 Heat equation

The first PDE we consider is the one-dimensional heat equation:

$$u_t = u_{xx}, \qquad x \in (-\pi, \pi) \tag{18}$$

with periodic boundary conditions. Using Fourier transform, it can be shown that discrete-time eigenvalues are [22]

$$\mu_k = -k^2, \quad k = 0, \pm 1, \pm 2, \dots \tag{19}$$

---

[1]We don't expect eigenfunctions to be exactly $[x_1, x_1^2, x_2 - bx_1^2]$ because each one is still a valid eigenfunction (associated with the same eigenvalue) when being multiplied by some factors. Therefore, this statement is correct up to some constant scale.
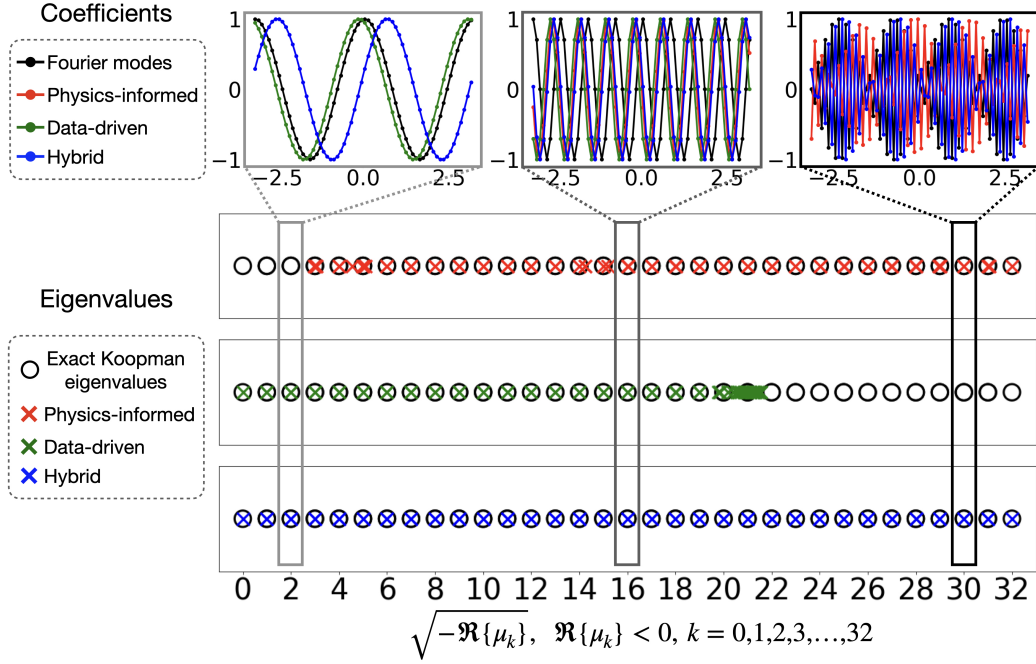
Figure 2: The eigenvalues (with negative real part) of the matrix L from different neural networks are plotted along with the exact, discrete-time eigenvalues of the heat equation at the bottom. The top row shows the coefficients of the linear transformation corresponding to the selected eigenvalues.
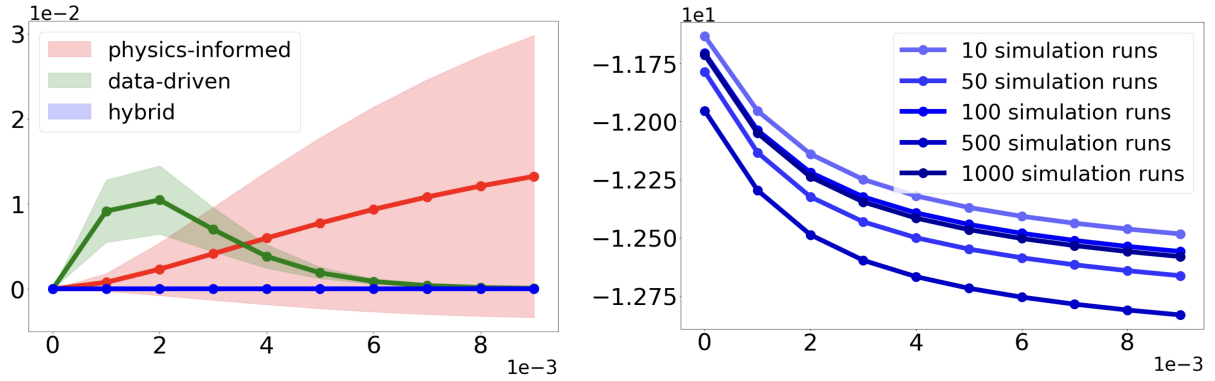

Figure 3: The left plot shows the predictive power of three different neural networks. The prediction task is evaluated on 1000 newly simulated test trials. The shaded region highlights one standard deviation from the mean squared error. The right plot shows the error curves of 5 hybrid models. These models differ on the number of simulated trajectories used for training. And the error is visualized in a logarithmic scale of base 10. Both plots are 10-step forward predictions with $\Delta t = 0.001$.

To approximately represent the trajectories u and the collocations u_t, we discretize the spatial domain with $n = 64$ equally spaced grids. As a consequence, the value of k only ranges from

9

−32 to 31. Therefore, we expect our network to at least mimic a discrete Fourier transform and its inverse transform, identifying the right eigenvalues after training. In addition, we study the effects of data integration. More specifically, we compare the results obtained from the networks that is purely physics-informed, purely data-driven and a mixture of both which we call it hybrid model. We use sums of harmonic functions with random coefficients as collocations for which the analytic spacial derivatives are available. More details of the experimental setup can be found in the Appendix A.1.2.

Indeed, Fig 2 shows that the transformation coefficients collide with the frequencies of the corresponding Fourier modes. The phase difference is expected because Discrete Fourier transformation is not unique for diagonalization of the heat equation. The networks nearly identify all the correct eigenvalues of the heat equation, however, the purely physics-informed network fails to discover the low-frequency modes. In addition, we observe it identifies an eigenvalue with a small positive real part (around 0.00005) which is not visualized in this plot. On the contrary, the purely data-driven network misses the high-frequency modes but all identified eigenvalues have negative real parts; the hybrid model presents the most satisfying accuracy among all.

The reason behind this result might be the multiscale feature of the system. Namely (i) for data-driven model, we use simulation data which are integrated over time. Slow dynamics (which are associated with the slow frequencies) are more persistent which dominate in the overall loss. (ii) For physics-informed model, however, we use the spatial derivatives of the states which are more sensitive to the high-frequency (transient) modes. (iii) The hybrid model leverages these two time scales and achieves a nice balance for identifying all eigenvalues. This important result suggests using collocations to inform the model of the aspects of the phenomenon that is missing in the data.

Fig 3 shows the error curves of the networks trained in different ways. The left plot shows physics-informed network is good at short-term predictions while the data-driven network is more promising in long-term predictions. The hybrid network merges the best part of these two and consistently offer the most accurate predictions. We also find from the right plot that the performance of the hybrid model is not sensitive to the number of simulation data used for training. More importantly, training with the largest number of simulation data does not necessarily gives the best prediction results. This is probably because in principle the physics-informed training is sufficient for finding all modes, adding simulation data can only help better discover the persistent modes. This suggests that if the network is physics-informed, data demand from simulation is low although data integration is beneficial for identifying a more accurate model.

4.3 Burger's equation

For the next example, we consider the nonlinear PDE known as the Burger's equation

$$u_t = -uu_x + \nu u_{xx}, \ x \in (-\pi, \pi) \tag{20}$$

with periodic boundary conditions. We choose a small value of $\nu = 0.01$ such that the solution is advection-dominated, for which the linearization is more challenging [64, 40].

We show that one can use collocations to improve model's performance on types of initial conditions that are missing in the available training data. For that we train two models. The data-driven model only uses 1024 trajectories with bell-curve initial conditions (ICs) (we provide an example at the top-right frame of the Figure 4). The hybrid model additionally observes
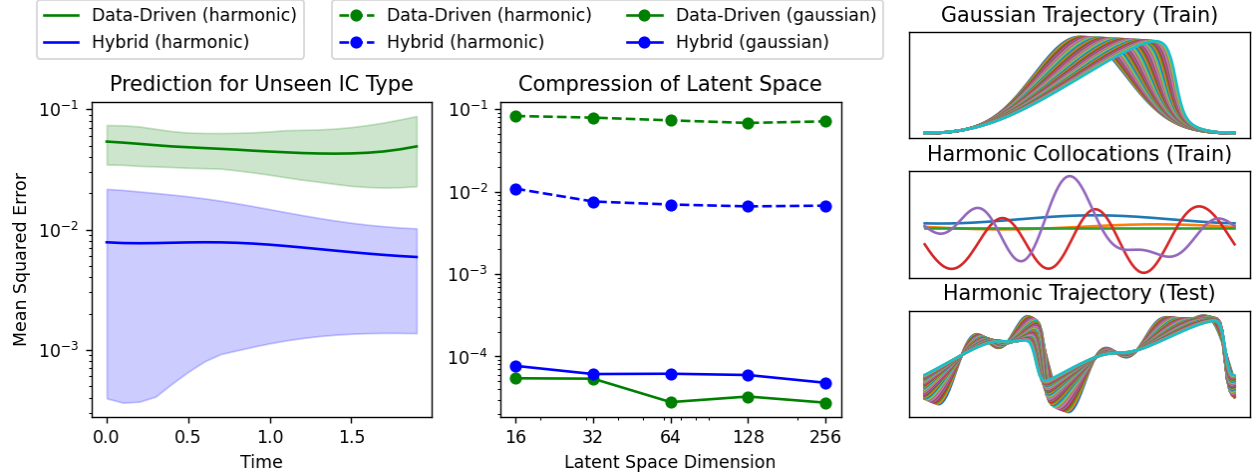
Figure 4: The right plots give examples of data snapshots used: trajectories with bell-curve ICs (top) and harmonic ICs (bottom). The middle-right pane shows harmonic collocations used by hybrid models in addition to the snapshots. The left plot compares the prediction errors (MSE) of two models, data-driven and hybrid with 128-dimensional latent spaces, on harmonic initial conditions that were not present in the trained data. The shaded regions represent 95% confidence intervals based on 100 test trajectories. The middle plot shows the prediction error (MSE) on each type of test data for a variety of models with different latent-space sizes. We see that the use of harmonic collocations significantly improves the model performance on unseen harmonic ICs without increasing the errors on bell-curve ICs.

80000 harmonic collocations formed by summing first 10 sinusoidal modes with random coefficients (Figure 4, middle-right frame), for which we evaluate u_t analytically using Equation 20. Next, we evaluate the performance of both models using unseen trajectories with both harmonic and bell-curve ICs, 100 trajectories each. We observe that the Hybrid model predicts the sinusoidal trajectories 10 times better than the data-driven one (Figure 4, left frame, shown for the 128-dimensional latent-space model). Since neither models had any trajectories of that type in its training set we conclude that the difference in performance comes from using harmonic collocations. We also note that better performance of the hybrid model on harmonic ICs does not come at an expense of worse performance on bell-curve ICs, as shown in the central frame of Figure 4. This evidence suggests that one can improve a model's extrapolation power by supplementing its training with sufficiently diverse set of collocations, especially when additional simulations are expensive to obtain but the collocations are cheap to generate. The details of the network's architecture and training procedure are provided in the Appendix A.1.3.

Finally, we highlight a remarkable compressibility of the latent space of PIKN models: a model with a latent space size 16 predicts only two times worse, by MSE, than a model with 256-dimensional latent space (central frame of Figure 4). It provides an empirical evidence that using a large latent space is not necessary for achieving good practical performance, albeit one may not discover Koopman eigenfunctions. To illustrate it, we compare our results with that of the exact Koopman eigenvalues and eigenfunctions for the Burger's equation, that can be found using Cole-Hopf transformation in the appendix A.2.4. The learned transformations are not Cole-Hopf; indeed, Cole-Hopf transformations may not be the only one to linearize Burger's equation, similar

to the problem arised in [24]. However, the linear dynamics in the latent space still give good prediction, especially after adding physics-informed regularization, as evidenced by Figure 4. Ultimately, high compressibility enables using PIKN for compressed sensing and online control applications. [38]

5   Discussion and Conclusion

In this work, we presented an effective deep learning framework for identifying Koopman eigenvalue and eigenfunction pairs for reconstructing high-dimensional nonlinear dynamics. In order to validate our method, we carefully went through three examples on which the analytical form of the Koopman eigen-decomposition can be derived. To the best of our knowledge, this is the first work that leverages knowledge of physics to improve the performance of auto-encoder-based Koopman learning. Our results show that (i) by imposing the Lie equation via soft penalty one can reduce the need of large training data-sets as being required in previous works; (ii) since the framework is under the scope of operator learning, our model can be used for future state predictions on unseen initial states, and (iii) using appropriate collocations one can improve the prediction accuracy on those unseen states by assimilating the knowledge of the dynamics into the model.

This work also suggests a number of future research directions. For example, many nonlinear systems may have different Koopman eigen-decompositions over different domains[62]. Being able to identify the boundaries of these domains would greatly expand the scope of applications of this approach. It is also interesting to see how these model can benefit the control problems, as in the real world, it is the ultimate goal of studying nonlinear dynamics.

References

[1] Mostafa Al-Gabalawy. Deep learning for koopman operator optimal control. *ISA transactions*, 2021.

[2] Hassan Arbabi and Igor Mezic. Ergodic theory, dynamic mode decomposition, and computation of spectral properties of the koopman operator. *SIAM Journal on Applied Dynamical Systems*, 16(4):2096–2126, 2017.

[3] Travis Askham and J Nathan Kutz. Variable projection methods for an optimized dynamic mode decomposition. *SIAM Journal on Applied Dynamical Systems*, 17(1):380–416, 2018.

[4] Omri Azencot, N Benjamin Erichson, Vanessa Lin, and Michael Mahoney. Forecasting sequential data using consistent koopman autoencoders. In *International Conference on Machine Learning*, pages 475–485. PMLR, 2020.

[5] Shervin Bagheri. Koopman-mode decomposition of the cylinder wake. *Journal of Fluid Mechanics*, 726:596–623, 2013.

[6] Mikhael Balabane, Miguel Alfonso Mendez, and Sara Najem. Koopman operator for burgers's equation. *Physical Review Fluids*, 6(6):064401, 2021.

[7] Jérémy Basley, Luc R Pastur, Nathalie Delprat, and François Lusseyran. Space-time aspects of a three-dimensional multi-modulated open cavity flow. *Physics of Fluids*, 25(6):064105, 2013.

[8] Gabriele Bellani. *Experimental studies of complex flows through image-based techniques*. PhD thesis, KTH Royal Institute of Technology, 2011.

[9] Erik Berger, Mark Sastuba, David Vogt, Bernhard Jung, and Heni Ben Amor. Dynamic mode decomposition for perturbation estimation in human robot interaction. In *The 23rd IEEE International Symposium on Robot and Human Interactive Communication*, pages 593–600. IEEE, 2014.

[10] Erik Berger, Mark Sastuba, David Vogt, Bernhard Jung, and Heni Ben Amor. Estimation of perturbations in robotic behavior using dynamic mode decomposition. *Advanced Robotics*, 29(5):331–343, 2015.

[11] Chongke Bi, Ye Yuan, Jiawan Zhang, Yun Shi, Yiqing Xiang, Yuehuan Wang, and RongHui Zhang. Dynamic mode decomposition based video shot detection. *IEEE Access*, 6:21397–21407, 2018.

[12] Diana Alina Bistrian and Ionel Michael Navon. Randomized dynamic mode decomposition for nonintrusive reduced order modelling. *International Journal for Numerical Methods in Engineering*, 112(1):3–25, 2017.

[13] Bingni W Brunton, Lise A Johnson, Jeffrey G Ojemann, and J Nathan Kutz. Extracting spatial–temporal coherent patterns in large-scale neural recordings using dynamic mode decomposition. *Journal of neuroscience methods*, 258:1–15, 2016.

[14] S. L. Brunton and J. N. Kutz. *Data-Driven Science and Engineering: Machine Learning, Dynamical Systems, and Control*. Cambridge University Press, 2019.

[15] Steven L Brunton, Marko Budišić, Eurika Kaiser, and J Nathan Kutz. Modern koopman theory for dynamical systems. *arXiv preprint arXiv:2102.12086*, 2021.

[16] TJ Clark and Angela D Luis. Nonlinear population dynamics are ubiquitous in animals. *Nature ecology & evolution*, 4(1):75–81, 2020.

[17] Julian D Cole. On a quasi-linear parabolic equation occurring in aerodynamics. *Quarterly of applied mathematics*, 9(3):225–236, 1951.

[18] Suddhasattwa Das and Dimitrios Giannakis. Koopman spectra in reproducing kernel hilbert spaces. *Applied and Computational Harmonic Analysis*, 49(2):573–607, 2020.

[19] Scott TM Dawson, Maziar S Hemati, Matthew O Williams, and Clarence W Rowley. Characterizing and correcting for the effect of sensor noise in the dynamic mode decomposition. *Experiments in Fluids*, 57(3):42, 2016.

[20] N Benjamin Erichson, Steven L Brunton, and J Nathan Kutz. Compressed dynamic mode decomposition for background modeling. *Journal of Real-Time Image Processing*, 16(5):1479–1492, 2019.

[21] N Benjamin Erichson, Lionel Mathelin, J Nathan Kutz, and Steven L Brunton. Randomized dynamic mode decomposition. *SIAM Journal on Applied Dynamical Systems*, 18(4):1867–1891, 2019.

[22] Lawrence C Evans. *Partial differential equations*, volume 19. American Mathematical Soc., 2010.

[23] Zhiwei Fang and Justin Zhan. A physics-informed neural network framework for pdes on 3d surfaces: Time independent problems. *IEEE Access*, 8:26328–26335, 2019.

[24] Craig Gin, Bethany Lusch, Steven L Brunton, and J Nathan Kutz. Deep learning models for global coordinate transformations that linearise pdes. *European Journal of Applied Mathematics*, 32(3):515–539, 2021.

[25] Jacob Grosek and J Nathan Kutz. Dynamic mode decomposition for real-time background/foreground separation in video. *arXiv preprint arXiv:1404.7592*, 2014.

[26] Yiqiang Han, Wenjian Hao, and Umesh Vaidya. Deep learning of koopman representation for control. In *2020 59th IEEE Conference on Decision and Control (CDC)*, pages 1890–1895. IEEE, 2020.

[27] Maziar S Hemati, Clarence W Rowley, Eric A Deem, and Louis N Cattafesta. De-biasing the dynamic mode decomposition for applied koopman spectral analysis of noisy datasets. *Theoretical and Computational Fluid Dynamics*, 31(4):349–368, 2017.

[28] Oliver Hennigh, Susheela Narasimhan, Mohammad Amin Nabian, Akshay Subramaniam, Kaustubh Tangsali, Zhiwei Fang, Max Rietmann, Wonmin Byeon, and Sanjay Choudhry. Nvidia simnet™: An ai-accelerated multi-physics simulation framework. In *International Conference on Computational Science*, pages 447–461. Springer, 2021.

[29] Eberhard Hopf. The partial differential equation u sub t+ uu sub x= mu sub xx. Technical report, INDIANA UNIV AT BLOOMINGTON, 1950.

[30] Kurt Hornik, Maxwell Stinchcombe, and Halbert White. Multilayer feedforward networks are universal approximators. *Neural networks*, 2(5):359–366, 1989.

[31] Ameya D Jagtap and George Em Karniadakis. Extended physics-informed neural networks (xpinns): A generalized space-time domain decomposition based deep learning framework for nonlinear partial differential equations. *Communications in Computational Physics*, 28(5):2002–2041, 2020.

[32] Ameya D Jagtap, Ehsan Kharazmi, and George Em Karniadakis. Conservative physics-informed neural networks on discrete domains for conservation laws: Applications to forward and inverse problems. *Computer Methods in Applied Mechanics and Engineering*, 365:113028, 2020.

[33] Mihailo R Jovanović, Peter J Schmid, and Joseph W Nichols. Sparsity-promoting dynamic mode decomposition. *Physics of Fluids*, 26(2):024103, 2014.

[34] Aniketh Kalur, Saleh Nabi, and Mouhacine Benosman. Robust adaptive dynamic mode decomposition for reduce order modelling of partial differential equations. In *2021 American Control Conference (ACC)*, pages 4497–4502. IEEE, 2021.

[35] Ehsan Kharazmi, Zhongqiang Zhang, and George Em Karniadakis. hp-vpinns: Variational physics-informed neural networks with domain decomposition. *Computer Methods in Applied Mechanics and Engineering*, 374:113547, 2021.

[36] Bernard O Koopman. Hamiltonian systems and transformation in hilbert space. *Proceedings of the national academy of sciences of the united states of america*, 17(5):315, 1931.

[37] Bernard O Koopman and J v Neumann. Dynamical systems of continuous spectra. *Proceedings of the National Academy of Sciences*, 18(3):255–263, 1932.

[38] Milan Korda and Igor Mezić. Linear predictors for nonlinear dynamical systems: Koopman operator meets model predictive control. *Automatica*, 93:149–160, 2018.

[39] Boris Kramer, Piyush Grover, Petros Boufounos, Saleh Nabi, and Mouhacine Benosman. Sparse sensing and dmd-based identification of flow regimes and bifurcations in complex flows. *SIAM Journal on Applied Dynamical Systems*, 16(2):1164–1196, 2017.

[40] J Nathan Kutz, Steven L Brunton, Bingni W Brunton, and Joshua L Proctor. *Dynamic mode decomposition: data-driven modeling of complex systems*. SIAM, 2016.

[41] J Nathan Kutz, Xing Fu, and Steven L Brunton. Multiresolution dynamic mode decomposition. *SIAM Journal on Applied Dynamical Systems*, 15(2):713–735, 2016.

[42] Henning Lange, Steven L Brunton, and Nathan Kutz. From fourier to koopman: Spectral methods for long-term time series prediction. *arXiv preprint arXiv:2004.00574*, 2020.

[43] Soledad Le Clainche and José M Vega. Higher order dynamic mode decomposition. *SIAM Journal on Applied Dynamical Systems*, 16(2):882–925, 2017.

[44] Qianxiao Li, Felix Dietrich, Erik M Bollt, and Ioannis G Kevrekidis. Extended dynamic mode decomposition with dictionary learning: A data-driven adaptive spectral decomposition of the koopman operator. *Chaos: An Interdisciplinary Journal of Nonlinear Science*, 27(10):103111, 2017.

[45] Yunzhu Li, Hao He, Jiajun Wu, Dina Katabi, and Antonio Torralba. Learning compositional koopman operators for model-based control. *arXiv preprint arXiv:1910.08264*, 2019.

[46] Edward N Lorenz. *Empirical orthogonal functions and statistical weather prediction*, volume 1. Massachusetts Institute of Technology, Department of Meteorology Cambridge, 1956.

[47] Lu Lu, Raphael Pestourie, Wenjie Yao, Zhicheng Wang, Francesc Verdugo, and Steven G Johnson. Physics-informed neural networks with hard constraints for inverse design. *arXiv preprint arXiv:2102.04626*, 2021.

[48] Bethany Lusch, J Nathan Kutz, and Steven L Brunton. Deep learning for universal linear embeddings of nonlinear dynamics. *Nature communications*, 9(1):1–10, 2018.

[49] Suryanarayana Maddu, Dominik Sturm, Christian L Müller, and Ivo F Sbalzarini. Inverse dirichlet weighting enables reliable training of physics informed neural networks. *Machine Learning: Science and Technology*, 2021.

[50] Jordan Mann and J Nathan Kutz. Dynamic mode decomposition for financial trading strategies. *Quantitative Finance*, 16(11):1643–1655, 2016.

[51] Xuhui Meng and George Em Karniadakis. A composite neural network that learns from multi-fidelity data: Application to function approximation and inverse pde problems. *Journal of Computational Physics*, 401:109020, 2020.

[52] Igor Mezic. *On the geometrical and statistical properties of dynamical systems: theory and applications*. PhD thesis, California Institute of Technology, 1994.

[53] Igor Mezić. Spectral properties of dynamical systems, model reduction and decompositions. *Nonlinear Dynamics*, 41(1):309–325, 2005.

[54] Igor Mezic. Koopman operator, geometry, and learning. *arXiv preprint arXiv:2010.05377*, 2020.

[55] Igor Mezić. On numerical approximations of the koopman operator. *Mathematics*, 10(7):1180, 2022.

[56] Siddhartha Mishra and Roberto Molinaro. Estimates on the generalization error of physics informed neural networks (pinns) for approximating pdes. *arXiv preprint arXiv:2006.16144*, 2020.

[57] Yoshinori Mizuno, Daniel Duke, Callum Atkinson, and Julio Soria. Investigation of wall-bounded turbulent flow using dynamic mode decomposition. In *Journal of Physics: Conference Series*, volume 318, page 042040. IOP Publishing, 2011.

[58] Jeremy Morton, Antony Jameson, Mykel J Kochenderfer, and Freddie Witherden. Deep dynamical modeling and control of unsteady fluid flows. *Advances in Neural Information Processing Systems*, 31, 2018.

[59] Jeremy Morton, Freddie D Witherden, and Mykel J Kochenderfer. Deep variational koopman models: Inferring koopman observations for uncertainty-aware dynamics modeling and control. *arXiv preprint arXiv:1902.09742*, 2019.

[60] J Nathan Kutz, Joshua L Proctor, and Steven L Brunton. Applied koopman theory for partial differential equations and data-driven modeling of spatio-temporal systems. *Complexity*, 2018, 2018.

[61] Jacob Page and Rich R Kerswell. Koopman analysis of burgers equation. *Physical Review Fluids*, 3(7):071901, 2018.

[62] Jacob Page and Rich R Kerswell. Koopman mode expansions between simple invariant solutions. *Journal of Fluid Mechanics*, 879:1–27, 2019.

[63] Guofei Pang, Lu Lu, and George Em Karniadakis. fpinns: Fractional physics-informed neural networks. *SIAM Journal on Scientific Computing*, 41(4):A2603–A2626, 2019.

[64] Benjamin Peherstorfer. Breaking the kolmogorov barrier with nonlinear model reduction. *Notices of the American Mathematical Society*, 69(5), 2022.

[65] Joshua L Proctor, Steven L Brunton, and J Nathan Kutz. Dynamic mode decomposition with control. *SIAM Journal on Applied Dynamical Systems*, 15(1):142–161, 2016.

[66] Joshua L Proctor and Philip A Eckhoff. Discovering dynamic patterns from infectious disease data using dynamic mode decomposition. *International health*, 7(2):139–145, 2015.

[67] Maziar Raissi, Paris Perdikaris, and George E Karniadakis. Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. *Journal of Computational Physics*, 378:686–707, 2019.

[68] Maziar Raissi, Paris Perdikaris, and George Em Karniadakis. Physics informed deep learning (part i): Data-driven solutions of nonlinear partial differential equations. *arXiv preprint arXiv:1711.10561*, 2017.

[69] Clarence W Rowley, Igor Mezić, Shervin Bagheri, Philipp Schlatter, and Dan S Henningson. Spectral analysis of nonlinear flows. *Journal of fluid mechanics*, 641:115–127, 2009.

[70] Peter J Schmid. Dynamic mode decomposition of numerical and experimental data. *Journal of fluid mechanics*, 656:5–28, 2010.

[71] Yeonjong Shin, Jerome Darbon, and George Em Karniadakis. On the convergence of physics informed neural networks for linear second-order elliptic and parabolic type pdes. *arXiv preprint arXiv:2004.01806*, 2020.

[72] Khemraj Shukla, Ameya D Jagtap, and George Em Karniadakis. Parallel physics-informed neural networks via domain decomposition. *arXiv preprint arXiv:2104.10013*, 2021.

[73] Luning Sun and Jian-Xun Wang. Physics-constrained bayesian neural network for fluid flow reconstruction with sparse and noisy data. *Theoretical and Applied Mechanics Letters*, 10(3):161–169, 2020.

[74] Naoya Takeishi, Yoshinobu Kawahara, and Takehisa Yairi. Learning koopman invariant subspaces for dynamic mode decomposition. *Advances in Neural Information Processing Systems*, 30, 2017.

[75] Roy Taylor, J Nathan Kutz, Kyle Morgan, and Brian A Nelson. Dynamic mode decomposition for plasma diagnostics and validation. *Review of Scientific Instruments*, 89(5):053501, 2018.

[76] Hiroaki Terao, Sho Shirasaka, and Hideyuki Suzuki. Extended dynamic mode decomposition with dictionary learning using neural ordinary differential equations. *Nonlinear Theory and Its Applications, IEICE*, 12(4):626–638, 2021.

[77] Jonathan H Tu, Clarence W Rowley, Dirk M Luchtenburg, Steven L Brunton, and J Nathan Kutz. On dynamic mode decomposition: Theory and applications. *arXiv preprint arXiv:1312.0041*, 2013.

[78] Sifan Wang, Yujun Teng, and Paris Perdikaris. Understanding and mitigating gradient flow pathologies in physics-informed neural networks. *SIAM Journal on Scientific Computing*, 43(5):A3055–A3081, 2021.

[79] Sifan Wang, Hanwen Wang, and Paris Perdikaris. Learning the solution operator of parametric partial differential equations with physics-informed deeponets. *arXiv preprint arXiv:2103.10974*, 2021.

[80] Sifan Wang, Xinling Yu, and Paris Perdikaris. When and why pinns fail to train: A neural tangent kernel perspective. *Journal of Computational Physics*, 449:110768, 2022.

[81] Matthew O Williams, Ioannis G Kevrekidis, and Clarence W Rowley. A data–driven approximation of the koopman operator: Extending dynamic mode decomposition. *Journal of Nonlinear Science*, 25(6):1307–1346, 2015.

[82] Liu Yang, Xuhui Meng, and George Em Karniadakis. B-pinns: Bayesian physics-informed neural networks for forward and inverse pde problems with noisy data. *Journal of Computational Physics*, 425:109913, 2021.

[83] Yibo Yang and Paris Perdikaris. Adversarial uncertainty quantification in physics-informed neural networks. *Journal of Computational Physics*, 394:136–152, 2019.

[84] Enoch Yeung, Soumya Kundu, and Nathan Hodas. Learning deep neural network representations for koopman operators of nonlinear dynamical systems. In *2019 American Control Conference (ACC)*, pages 4832–4839. IEEE, 2019.

[85] Dongkun Zhang, Ling Guo, and George Em Karniadakis. Learning in modal space: Solving time-dependent stochastic pdes using physics-informed neural networks. *SIAM Journal on Scientific Computing*, 42(2):A639–A665, 2020.

[86] Dongkun Zhang, Lu Lu, Ling Guo, and George Em Karniadakis. Quantifying total uncertainty in physics-informed neural networks for solving forward and inverse stochastic problems. *Journal of Computational Physics*, 397:108850, 2019.

[87] Yinhao Zhu, Nicholas Zabaras, Phaedon-Stelios Koutsourelakis, and Paris Perdikaris. Physics-constrained deep learning for high-dimensional surrogate modeling and uncertainty quantification without labeled data. *Journal of Computational Physics*, 394:56–81, 2019.

[88] Kirill Zubov, Zoe McCarthy, Yingbo Ma, Francesco Calisto, Valerio Pagliarino, Simone Azeglio, Luca Bottero, Emmanuel Luján, Valentin Sulzer, Ashutosh Bharambe, et al. Neuralpde: Automating physics-informed neural networks (pinns) with error approximations. *arXiv preprint arXiv:2107.09443*, 2021.

A Appendix

A.1 Experimental setups

For all networks in the experiments, L is set to a zero matrix initially and all other parameters are randomly initialized using the default initializer of Pytorch. The exponential linear unit (ELU) is used as the nonlinear activation function, unless stated otherwise, as we expect the transformations in PIKN to be relatively smooth.

A.1.1 Simple nonlinear system with discrete spectrum

In this example, we have trained two different types of PIKN: a PIKN with a linear decoder and a PIKN with a nonlinear decoder. In both experiments, 1000 collocation points were uniformly sampled from $[-1, 1] \times [-1, 1]$ for training, an Adam optimizer with a learning rate of $1e - 4$ has been applied. The total number of training epochs is set to 50000 and the weights in the loss function are set to $\omega_1 = \omega_2 = 1$. The encoder part of both architectures are the same: a 2-layer fully-connected neural network with hidden layer containing 50 neurons. The major differences between the two architectures are as follows:

- The linear decoder is simply a linear layer without a bias term whereas the nonlinear decoder is symmetric to the encoder: a 2-layer fully-connected neural network with hidden width 50.

- For the PIKN with a linear decoder, we have a three-dimensional latent space whereas for the PIKN with a nonlinear decoder, it is two-dimensional.

A.1.2 Heat equation

In the PIKN architecture for Heat equation, encoder and decoder are both linear and the latent dimension is set to 64, the same as the number of spatial grids. Number of training epochs is 100000 and an Adam optimizer has been applied for the training algorithm. In this set of experiments, we use an adaptive learning rate: initially set to 0.01, it keeps decreasing by a factor of 0.5 if no improvements are made over the recent 5000 epochs, until it hits the minimal value of 0.000001. With the network architecture and training parameters fixed, we train it in three different ways. Namely, we set different values for the weights $\omega_1, \omega_2, \omega_3, \omega_4$ in the loss function, leading to three different training regimes:

- $\omega_1 = 0.0001, \omega_2 = 1, \omega_3 = 0, \omega_4 = 0$: the network is purely physics-informed.

- $\omega_1 = 0, \omega_2 = 0, \omega_3 = 1, \omega_4 = 1$: this corresponds to a purely data-driven learning.

- $\omega_1 = 0.0001, \omega_2 = 1, \omega_3 = 1, \omega_4 = 1$: this represents the scenario where we train a physics-informed network with data integration. For simplicity, we call it hybrid training.

Notice that the value of ω_1 is set on a different order compared to other weights, it is because this loss term involves calculation of numerical derivatives which usually has a greater amplitude. This is a well-known issue for physics-informed neural network and has been thoroughly studied. Adaptive re-weighting schemes were proposed to fix it [78, 49, 88]. In our case, however, we find

	PIKN(linear decoder)	PIKN(nonlinear decoder)
Experiment 1	$\mu_1 = -0.10000689$, $\mu_2 = -0.19914131$, $\mu_3 = -0.9996788$	$\mu_1 = -0.09756267$, $\mu_2 = -0.99973810$
Experiment 2	$\mu_1 = -0.10009335$, $\mu_2 = -0.19947967$, $\mu_3 = -1.0004123$	$\mu_1 = -0.09644943$, $\mu_2 = -0.99867420$
Experiment 3	$\mu_1 = -0.10005733$, $\mu_2 = -0.19892442$, $\mu_3 = -1.0003881$	$\mu_1 = -0.09592330$, $\mu_2 = -0.99915651$
Experiment 4	$\mu_1 = -0.10008135$, $\mu_2 = -0.20019206$, $\mu_3 = -0.9986593$	$\mu_1 = -0.09784412$, $\mu_2 = -1.00298023$
Experiment 5	$\mu_1 = -0.09990316$, $\mu_2 = -0.20018657$, $\mu_3 = -0.9991975$	$\mu_1 = -0.09837312$, $\mu_2 = -1.00187280$
Experiment 6	$\mu_1 = -0.09997816$, $\mu_2 = -0.19966313$, $\mu_3 = -1.0005931$	$\mu_1 = -0.09764695$, $\mu_2 = -1.00517617$
Experiment 7	$\mu_1 = -0.10009032$, $\mu_2 = -0.19991782$, $\mu_3 = -1.0000535$	$\mu_1 = -0.09866422$, $\mu_2 = -0.99622254$
Experiment 8	$\mu_1 = -0.09998395$, $\mu_2 = -0.19948480$, $\mu_3 = -0.9996783$	$\mu_1 = -0.09856838$, $\mu_2 = -1.00204348$
Experiment 9	$\mu_1 = -0.09996726$, $\mu_2 = -0.19905518$, $\mu_3 = -1.0005406$	$\mu_1 = -0.09929386$, $\mu_2 = -0.99679565$
Experiment 10	$\mu_1 = -0.10004932$, $\mu_2 = -0.19994377$, $\mu_3 = -0.9993069$	$\mu_1 = -0.09756234$, $\mu_2 = -1.00279380$
summary	$\boldsymbol{\mu_1 = -0.10 \pm 6.22\mathrm{e}-04}$, $\boldsymbol{\mu_2 = -0.20 \pm 4.37\mathrm{e}-04}$, $\boldsymbol{\mu_3 = -1.00 \pm 5.99\mathrm{e}-05}$	$\boldsymbol{\mu_1 = -0.10 \pm 2.74\mathrm{e}-03}$, $\boldsymbol{\mu_2 = -1.00 \pm 9.68\mathrm{e}-04}$

Table 1: Eigenvalues identified by PIKNs at all training runs. The first column represents the PIKN with a linear decoder and the second column is the one with a nonlinear decoder.

that choosing a fixed, small value $\omega_1 = 0.0001$ is sufficient for achieving a fast convergence[2].

For the physics-informed learning, we create 1000 trial functions $\{u^{(1)}, u^{(2)}, \ldots, u^{(1000)}\}$ for training purpose. Each trial function $u^{(j)}$ is a superposition of the Fourier modes that satisfy the periodic boundary conditions. In our case, this amounts to using $\sin{(kx)}$ and $\cos{(kx)}$ for $k = 0, 1, 2, \ldots$ as basis functions. The value of k is restricted to be no greater than 32 due to the choice of our grid spatial resolution. The spatial derivatives of the trail functions $\{u^{(1)}_{xx}, u^{(2)}_{xx}, \ldots, u^{(1000)}_{xx}\}$ are calculated using numerical spectral method.

In the data-driven regime, we use simulation data obtained from a solver based on spectral method. We run the simulation with 1000 different initial states. For each run, the initial state of u is obtained through the same procedure as we obtain the trial functions $u^{(k)}$. Then we sample snapshots of the state u with a temporal gap $\Delta t = 0.01$ for 5 steps (i.e. $p = 5$).

For the hybrid training regime, both of the above two data-sets are used. However, to study the effects of the amount of simulation data, we conduct 10, 50, 100, 500, 1000 simulation runs in 5

---

[2]Another way to get around this is to learn the pseudo-inverse of L instead. Then the loss term reads $\|\phi(\mathbf{x_i}) - L^{\dagger}\nabla\phi(\mathbf{x_i}) \cdot \mathbf{f}(\mathbf{x_i})\|$. In that case ω_1 and ω_2 can be both set to 1 because the two loss terms are approximately on the same scale.

Table 2: Performance measures for Burger's experiment depending on the dimensions of the latent space (Figure 4)

separate groups of experiments.

A.1.3 Burger's Equation

In the PIKN architecture for Burger's equation, the encoder is implemented as a a feed-forward network with the size of input layer set to be equal 128 (the number of spatial grid-points). It has two hidden layers with 512 neurons. The size of the output layer of the encoder – the size of the latent dimension – varies from 16 to 256 to study the latent space compressibility (see Figure 4). The decoder's architecture mirrors the architecture of the encoder with the sizes of the inputs and outputs switched. An Adam optimizer has been applied for 500 epochs for training. We use an adaptive learning rate: initially set to 10^{-4}, it keeps decreasing by a factor of 0.5 if no improvement has been made over the recent 20 epochs, until it hits the minimal value of 10^{-7}.
Each network was trained in two different ways. Namely, we set different values for the weights $\omega_1, \omega_2, \omega_3, \omega_4$ in the loss function, leading to two different training regimes:

- $\omega_1 = 0, \omega_2 = 0, \omega_3 = 1, \omega_4 = 1$. These settings lead to a purely data-driven learning; it corresponds to green lines on Figure 4.

- $\omega_1 = 1, \omega_2 = 1, \omega_3 = 1, \omega_4 = 1$. These settings represent a model that uses both data trajectories and collocations (a hybrid model); it corresponds to blue lines on Figure 4.

We sample 80000 functions $\{u^{(1)}, u^{(2)}, \ldots, u^{(80000)}\}$ to use them as collocations. Each function $u^{(j)}$ is a superposition of the Fourier modes that satisfy the periodic boundary conditions. We call them "harmonic" initial conditions, five examples of which are displayed on the middle-right pane of Figure 4. We used $\sin(kx)$ and $\cos(kx)$ for $k = 0, 1, 2, \ldots$ evaluated on $x \in [-\pi; \pi]$-interval as basis functions. The value of k is restricted to be no greater than 10. To evaluate u_t at each collocation point we evaluated the spatial derivatives $\{(u_x^{(i)}, u_{xx}^{(i)})\}_{i=1}^{80000}$ using a numerical spectral method.

For training trajectories, we use simulation data obtained from a solver base on spectral method. We run the simulation with 1024 different initial states. Each initial state of u is a Gaussian (bell-) curve with mean 0 and a randomly generated variance $\sigma \sim U(0.1, 1)$, evaluated on $[-\pi; \pi]$-interval. Then we sample snapshots of the state u with a temporal gap $\Delta t = 0.1$ for 20 steps (i.e. $p = 20$). An example trajectory is displayed on the top-right pane of Figure 4.

To evaluate and compare the performance of both models we use 200 trajectories: 100 trajectories with harmonic ICs and 100 trajectories with Gaussian ICs generated exactly in the same way as described above. Then we sample snapshots of the state u with a temporal gap $\Delta t = 0.1$ for 20 steps (i.e. $p = 20$). The results are aggregated in Table 2 and visualized on Figure 2.

A.1.4 Hardware

All experiments were computed on a `slurm`-allocation that had 2 CPUs of an `Intel(R) Xeon(R) CPU E5-2630 v4 @ 2.20GHz`, 16 GB of memory, and one `Tesla K80` GPU. The experiments were implemented using `Python 3.9.12` and `PyTorch 1.11.0` that was using the GPU for compute.

(#collocations, #snapshots)	$\mu = -0.1$	$\lambda = -1$	eigenvalues=(-1, -0.1)
$(0, 1000)$	$-$	$-$	$(-0.962 \pm 0.118, -0.096 \pm 0.282)$
$(250, 750)$	-0.099 ± 0.002	-0.980 ± 0.033	$(-0.974 \pm 0.043, -0.099 \pm 0.001)$
$(500, 500)$	-0.099 ± 0.000	-0.991 ± 0.005	$(-0.985 \pm 0.010, -0.099 \pm 0.001)$
$(750, 250)$	-0.099 ± 0.003	-0.972 ± 0.043	$(-0.951 \pm 0.073, -0.096 \pm 0.010)$

Table 3: Networks are trained with different combinations of data-sets shown in the first column. The first row represents data-driven training and others are hybrid PIKN. The identified system parameters μ and λ and eigenvalues of Koopman operator are shown, respectively, in second, third, and fourth column.

A.2 Additional discussions

A.2.1 Potential pitfalls of using nonlinear transformations

For the example of ODE with discrete spectrum, we have run the training algorithm for 10 times for both architectures and the results are robust in the sense that the identified eigenvalues are all centered around the real ones we derived analytically, which are presented in Table 1. One can see our PIKNs faithfully recover the desired Koopman eigenvalues. The one with the linear decoding provides slightly more robust results, indicating the benefits of adding more known constraints. We notice that, however, the PIKN with nonlinear decoder doesn't always identify the eigenvalue $\mu_1 = -0.1$. If we significantly change the initialization of the network parameters, sometimes other values emerge. This indicates other transformations exist to linearize the dynamics and reconstruct the state variables. As an example, one can easily check $\varphi_\mu^\beta = x_1^\beta$ for all $\beta \in \mathbb{N}$ are all valid Koopman eigenfunctions associated with eigenvalues $\mu\beta$ and can be used for reconstruction. The lesson here is that by using a nonlinear decoder, we not only increase the flexibility of the Koopman operator theory framework, but also dramatically increase the searching space, which may lead to different learning outcomes.

A.2.2 Unknown parameters in physical systems

In this experiment, we demonstrate another advantage of PIKN in comparison to data-driven only approaches. We demonstrate that PIKN can leverage partial knowledge of physics, e.g. when some parameters of the system are unknown and even estimate those missing parameters. To illustrate this, we consider the dynamics of the form of Eq. 15 but we let μ and λ to be unknown parameters, which can be treated as trainable parameters with random initialization. We use different combinations of training data-sets and each model is trained for 10 times with different random initializations for the unknown parameters. The difference of data-sets is related to the combination of snapshots (obtained by simulator) versus collocation points (sampled from appropriate function space with no requirement of simulation). The results are shown in Table 3: all hybrid models successfully identified Koopman eigenvalues with higher accuracy than the data-driven model (first row). Note that, the data driven model does not involve a parameter estimation procedure and thus does not provide any knowledge of the physical parameters. On the other hand, the hybrid models are able to solve for the unknown parameters μ and λ correctly by filling gaps in the knowledge of physics with data, effectively operating as a simple model-discovery tool. The experiment shows

that incorporating a physics-informed loss is beneficial even when only partial knowledge of physics is accessible to the practitioner.

A.2.3 Alternative approach to enforce linearity

In fact, minimizing $\|L\phi(\mathbf{x}) - \nabla\phi(\mathbf{x}) \cdot \mathbf{f}(\mathbf{x})\|$ is not the only way to enforce linearity. Take ODE as an example, the decoder reads

$$\hat{\mathbf{x}}(t) = \psi(\mathbf{z}(t)) \tag{21}$$

This implies

$$\frac{d\hat{\mathbf{x}}}{dt} = \nabla\psi(\mathbf{z}) \cdot (L\mathbf{z}) \tag{22}$$

Therefore, minimizing $\|\nabla\psi(\mathbf{z}) \cdot (L\mathbf{z}) - \mathbf{f}(\hat{\mathbf{x}})\|$ is an alternative way to enforce linearity. This alternative approach is also consistent with 'future state prediction' loss in the work of [48]. The difference between these two ways is whether the linear constraint is applied to the encoder or decoder. In practice, however, we find using either form of the linearity loss or use both of them all work well and does not lead to significantly different results.

A.2.4 Cole-Hopf transfomration and comparison with exact Koopman eigenvalues and eigenfunctions

It is known that the Burger's equation can be linearized through Cole-Hopf transformation which was discovered in 1950 and 1951 by Eberhard Hopf[29] and Julian Cole[17], independently of one another, and later noticed by Kutz et al [60] and exploited by many others[61, 6]. The Cole-Hopf transformations are defined as:

$$u := C(v) = -\frac{2v_x}{v}$$
$$v := H(u) = \frac{e^{-\frac{1}{2}\int_{-\pi}^{x} u(s,t)ds}}{\int_{-\pi}^{\pi} e^{-\frac{1}{2}\int_{-\pi}^{x} u(s,t)ds} dx} \tag{23}$$

If $u(x,t)$ satisfies the Burger's equation 20 with a homogeneous boundary condition $u(-\pi,t) = u(\pi,t) = 0$ and an initial condition $u(x,0) = u_0(x)$, then $v(x,t) = H(u(x,t))$ solves the heat equation:

$$v_t = v_{xx}, \; x \in (-\pi,\pi)$$
$$v_x(-\pi,t) = v_x(\pi,t) = 0 \tag{24}$$
$$v(x,0) := v_0(x) = H(u_0(x))$$

The solution of it, by using a standard separation of variables approach, can be derived as

$$v(x,t) = C_0 + \sum_{k=1}^{\infty} C_k \cos(kx)e^{-k^2 t} \tag{25}$$

where $C_0 = \frac{1}{2\pi}\int_{-\pi}^{\pi} v_0(x)dx$ and $C_k = \frac{1}{\pi}\int_{-\pi}^{\pi} v_0(x)\cos(kx)dx$.
Conversely, it is also shown in [6] that if $v(x,t)$ solves the above heat equation and further satisfies

$$v(x,t) > 0, \; x \in (-\pi,\pi), t > 0$$
$$\int_{-\pi}^{\pi} v(x,t)dx = 1 \tag{26}$$

then $u(x,t) = C(v(x,t))$ is also a solution for the original Burger's equation. To satisfy these constraints, we can require $C_0 = \frac{1}{2\pi}$ and $\sum_{k=1}^{\infty} |C_k| < \frac{1}{2\pi}$. Therefore, we use the following procedure to generate data:

- Choose a number of modes K (i.e. $C_k = 0$ for $k > K$).

- Create a list of $\{\tilde{C}_1, \tilde{C}_2, \cdots, \tilde{C}_K\}$ with each $\tilde{C}_k$ uniformly sampled from $(-1, 1)$.

- Obtain a number α uniformly sampled from $(0, 1)$.

- Create a list of $\{C_1, C_2, \cdots, C_K\}$ with each $C_k = \frac{\alpha \tilde{C}_k}{\sum_{k=1}^{K} |\tilde{C}_k|}$ (so that $\sum_{k=1}^{K} |C_k| = \alpha < 1$)

- Generate $v(x,t) = \frac{1}{2\pi} + \sum_{k=1}^{K} C_k \cos(kx) e^{-k^2 t}$.

- Obtain $u(x,t) = C(v(x,t))$.

This data generation process guarantees the existence of at least one transformation (Cole-Hopf transformation followed by a Fourier transformation) that linearizes the nonlinear Burger's equation with eigenvalues $\mu_k = -k^2$ ($k = 1, 2, ..., K^2$). In our experiments, we set $K = 10$.
The architecture used for this experiment consists of a nonlinear encoder and a nonlinear decoder. Both encoder and decoder are 3-layer fully connected neural network with hidden width 512. The latent dimension is set to the same as the number of modes $K = 10$. The training parameters (learning rates and number of training epochs) are set to be the same as in the heat equation experiments. And the number of spatial grids is set to 512.
We also study the three different learning regimes, with different combinations of $\omega_1, \omega_2, \omega_3$ and ω_4. The only difference from the setup of the heat equation experiments is that ω_1 is set to 0.01.
For physics-informed learning, we create 1000 trial functions $\{u^{(1)}, u^{(2)}, \ldots, u^{(1000)}\}$ for training. Each trial function $u^{(j)}$ is a snapshot at a random time point $t^{(j)} \in [0, 0.1)$ of the $u(x,t)$ generated from the above procedure and projected onto the spatial grids. To generate $\{u_x^{(1)}, u_x^{(2)}, \ldots, u_x^{(1000)}\}$ and $\{u_{xx}^{(1)}, u_{xx}^{(2)}, \ldots, u_{xx}^{(1000)}\}$, we use $u = C(v) = -\frac{2v_x}{v}$, so that

$$
\begin{aligned}
u_x &= 2\left(\frac{v_x}{v}\right)^2 - 2\frac{v_{xx}}{v} \\
u_{xx} &= 6\frac{v_x v_{xx}}{v^2} - 2\frac{v_{xxx}}{v} - 4\left(\frac{v_x}{v}\right)^3
\end{aligned}
\tag{27}
$$

where v_x, v_{xx} and v_{xxx} are easy to derive.
For data-driven learning, we also obtain data from the above procedure. We sample snapshots of the state $u(x,t)$ with a temporal gap $\Delta t = 0.02$ for 5 steps. For the hybrid regime, both types of data are used.
In fig 5, the identified eigenvalues for Burger's equation of each training strategy are plotted along with the exact Koopman eigenvalues. The hybrid model, similar to the results of heat equation, seems to work slightly better for combination of low- and high- frequencies and reveal a more appealing spectrum, compared to data-driven or physics-informed regime alone (although significant improvement is not identified). This indicates the learned transformations are not exactly Cole-Hopf. However, in terms of prediction, we report significant improvement and generalization when unseen initial conditions are used as test data.
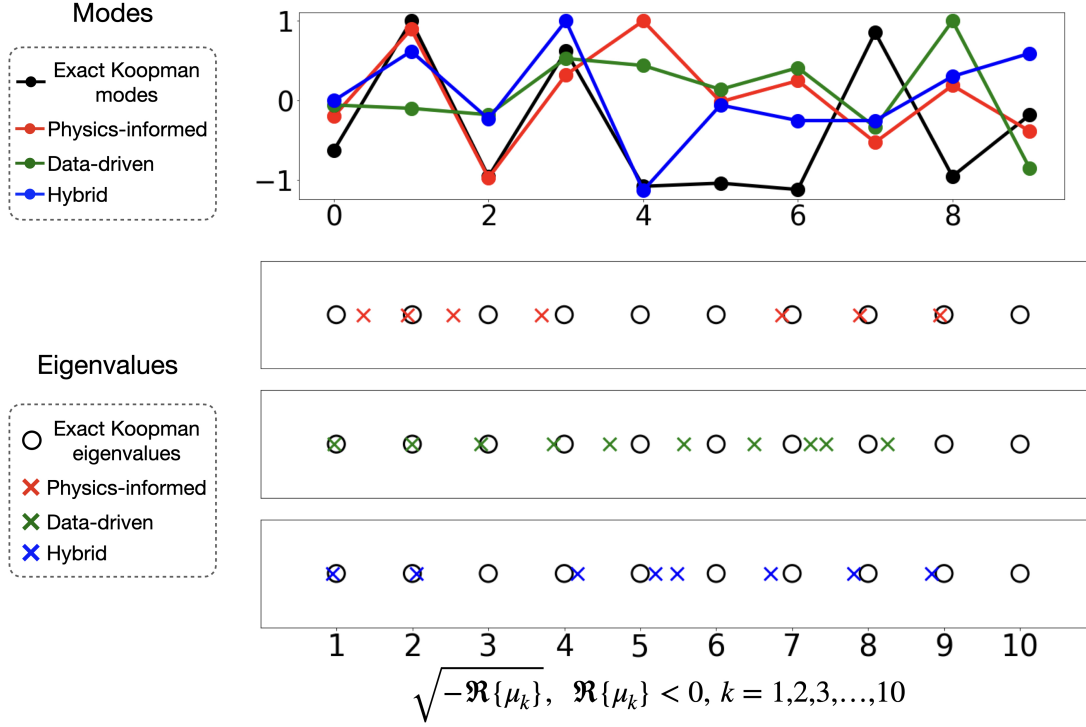
Figure 5: The first row shows the identified modes (or latent representation) of a sampled test input from different network architectures along with the exact Koopman modes. The second row shows eigenvalues (with negative real part) from different neural networks along with the exact, discrete-time eigenvalues of the Burger's equation identified through Cole-Hopf.