Physica D 483 (2025) 134991

Contents lists available at ScienceDirect

Physica D

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

Data-driven identification of Koopman eigenmodes constrained by Koopman eigenvalues using a neural network-based approach☆

Talha Ahmed *, Dan Wilson

Department of Electrical Engineering and Computer Science, University of Tennessee, Knoxville, TN 37996, USA

ARTICLE INFO

Communicated by Victor M. Perez-Garcia

Keywords:
Data-driven model identification
Koopman eigenfunctions
Neural networks
Dynamic mode decomposition
Nonlinear dynamical systems

ABSTRACT

Data-driven strategies can be used to infer representations of dynamical systems when the underlying model is unknown. Dynamic mode decomposition (DMD) is one such approach that can be used to identify the finite dimensional linear mapping that minimizes the squared error of the residuals for data recorded at successive timesteps. The associated modes of the resulting linear operator are an approximation of the Koopman eigenmodes of the underlying dynamical system, but these modes are not explicitly considered in the model identification process. General approaches that explicitly consider information about the Koopman eigenmodes themselves to inform the model fitting process have yet to be developed. In this work, we propose and implement an artificial neural network-based strategy for inferring Koopman eigenmodes from data that explicitly considers information about the Koopman eigenvalues during the model identification stage. Results are illustrated through a variety of examples that show improved accuracy of the proposed strategy in comparison to traditional DMD, especially when long term predictions are required.

1. Introduction

Modeling and analysis of dynamical systems is of paramount importance in a wide variety of fields such as climate, biology, ecology, traffic and finance [1]. Forecasting such systems also allows for the subsequent design of controllers [2–4]. It is often nontrivial to formulate a mathematical model for complex processes that can accurately emulate the dynamics. In situations where only observable data is accessible, data-driven approaches are often utilized. Numerous data-driven strategies have been proposed that not only find an accurate approximation of the dynamics through data but also perform dimensionality reduction to derive a low-dimensional model representation. One such strategy is proper orthogonal decomposition (POD) [5,6] and its variants [7,8], which aims to represent large multidimensional datasets with an optimal set of modes in an L2 energy sense. These modes can then be utilized to obtain a low-dimensional dynamical model representation by projecting the model equations onto the POD modes [9,10]. Authors in [11] propose a data-driven balanced truncation for model reduction where observed system response data is utilized to find the representation of the reduced model without utilizing any prior realization of the original model.

Dynamic mode decomposition (DMD) [12,13] is another well-established data-driven model identification strategy that fits a linear operator to describe the mapping from one timestep to the next. The resulting system representation minimizes the squared norm of the residual and can be used to predict the temporal evolution of observables. Extended DMD [14] considers a set of user-specified basis functions, lifting to a higher dimensional space and fitting the model to the resulting data. Another variation, DMD with control [15], accounts for external actuation in order to distinguish the autonomous system dynamics with those shifted through external input. DMD provides a good approximation of the system dynamics in many applications. However, one shortcoming is that standard DMD only considers the dynamics on a single time scale. When the system evolves on multiple timescales or when long term predictions are required, the accuracy of the predictions may suffer. Multi-resolution DMD [16], an extension of standard DMD that separates the slow decaying modes from fast decaying modes at specified levels, is able to account for different temporal resolutions, but its recursive implementation is not straightforward.

DMD has a close connection to Koopman operator theory and can be thought of as finding a finite-dimensional realization for the action of the (generally) infinite-dimensional Koopman operator [17–19]. The eigenvalues and eigenvectors of the resulting model obtained from DMD provide an approximation for the true Koopman eigenvalues and Koopman eigenfunctions of the Koopman operator [20]. However, at present, there are no general techniques available to incorporate *a priori*

☆ This article is part of a Special issue entitled: 'MLDSAIT' published in Physica D.

* Corresponding author.
E-mail address: tahmed4@alum.utk.edu (T. Ahmed).

https://doi.org/10.1016/j.physd.2025.134991
Received 14 May 2024; Received in revised form 26 September 2025; Accepted 18 October 2025
Available online 21 October 2025

knowledge about the Koopman eigenvalues and eigenvectors into the model fitting procedure. Such methods, for instance, could be used to prioritize slow-decaying eigenmodes or eigenmodes corresponding to a specific frequency of oscillation, ultimately providing a clearer picture about the Koopman eigenmodes that are most relevant to the underlying dynamics and subsequently yielding models that provide better predictions on longer prediction horizons.

With these considerations in mind, in this work we propose a strategy to learn the eigenspace representation of a system from data by structuring a neural network in such a way that its trainable weights learn information related to Koopman eigenvalues of the underlying dynamical model. In addition to minimizing the error associated with the mapping between observations, this approach allows for the inclusion of explicit constraints on the associated Koopman eigenvalues that can account for domain specific knowledge. A similar idea is considered in [21] which devises a novel strategy for DMD by reformulating it to access its eigenvalues for a given system and enforcing frequency constraints to change the eigenvalue placement while ensuring minimal changes to DMD. The constrained DMD approach relies on specifying frequencies through eigenvalues by utilizing prior domain knowledge and applying it using a mathematical derivation unlike our proposed approach which finds the eigenspace representation by utilizing a neural network. There are other strategies present in literature that find eigenvalues through neural networks. For example, the work in [22] proposes a simple feed forward neural network structure to compute eigenvalues and eigenvectors of the underlying system by training it to match desired output patterns. Similarly, in [23], convolutional neural networks (CNN) are employed for eigenvalue prediction of 1-D and 2-D photonic crystals by recognizing the underlying patterns observed in the given output data. However, both of these works focus on solving the eigenvalue problem and finding the eigenvalues of a given system unlike our proposed strategy which learns the underlying model dynamics through data by approximating the associated Koopman eigenfunctions.

The organization of this paper is as follows: Section 2 provides necessary background on time-delay embeddings, the dynamic mode decomposition (DMD) algorithm, and its relationship to Koopman operator theory. Section 3 describes the mathematical formulation that underlies our proposed approach, ultimately enabling artificial neural networks to learn an approximation of the Koopman eigenfunctions. Results are given in Section 4 where we illustrate the proposed technique on a variety of examples with both numerical and experimental data. Section 5 gives concluding remarks.

2. Background

In this section, a brief background of the standard DMD algorithm, its connection to Koopman operator theory, and the motivation for the proposed approach are provided.

2.1. Dynamic mode decomposition and time-delay embedding

As illustrated in [13], for a system with n real-valued observables, time series measurements can be obtained at fixed time intervals, Δt, and then stacked together to construct a matrix. Consider a data matrix $\mathbf{X} \in \mathbb{R}^{n\times m}$ containing time series measurements of the form

$$\mathbf{X} = \begin{bmatrix} \mathbf{x_1} & \mathbf{x_2} & \cdots & \mathbf{x_m} \end{bmatrix}, \tag{1}$$

where $\mathbf{x_1}, \ldots, \mathbf{x_m} \in \mathbb{R}^n$. Similarly, a matrix $\mathbf{Y}$ of the time shifted data is defined as

$$\mathbf{Y} = \begin{bmatrix} \mathbf{x_2} & \mathbf{x_3} & \cdots & \mathbf{x_{m+1}} \end{bmatrix}. \tag{2}$$

The primary objective of DMD is to approximate a system matrix $\mathbf{A}$, such that $\mathbf{Y} = \mathbf{AX}$. Here, $\mathbf{A} \in \mathbb{R}^{n\times n}$ can be found by considering a least-squares solution to the problem $\mathbf{A} = \mathbf{YX}^{\dagger}$ where $\dagger$ represents the Moore–Penrose pseudoinverse [13]. The pseudoinverse of $\mathbf{X}$ is taken by using singular value decomposition (SVD) which results in

$$\mathbf{X} = \mathbf{U\Sigma V}^*, \tag{3}$$

where $\mathbf{U} \in \mathbb{R}^{n\times r}$, $\mathbf{\Sigma} \in \mathbb{R}^{r\times r}$ is a diagonal matrix, $\mathbf{V}^* \in \mathbb{R}^{r\times m}$, r is the rank of $\mathbf{X}$ specified by the number of eigenvalues chosen and * denotes the conjugate transpose. An approximation to $\mathbf{A}$ can then be found using DMD based on the number of eigenvalues specified through SVD as

$$\mathbf{A} \approx \mathbf{YV\Sigma^{-1}U}^*, \tag{4}$$

Thus, the system $\hat{\mathbf{x}}_{\mathbf{k+1}} = \mathbf{A}\hat{\mathbf{x}}_{\mathbf{k}}$ can be obtained for the underlying model dynamics where $\hat{\mathbf{x}}$ is the by DMD.

Standard DMD as described above is generally performed using the system observables in the data matrix $\mathbf{X}$. Lifting the observables to a higher dimensional space before performing DMD can be used to obtain a richer representation of the system dynamics [24]. Time-delay embedding is one approach that can be used to lift data from (1) to a higher dimensional space. For data contained in $\mathbf{X}$ from (1), a time-delay embedding of length n_d can be constructed according to [25]

$$\mathbf{H} = \begin{bmatrix} \mathbf{x_1} & \mathbf{x_2} & \cdots & \mathbf{x_{m-n_d+1}} \\ \mathbf{x_2} & \mathbf{x_3} & \cdots & \mathbf{x_{m-n_d+2}} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{x_{n_d}} & \mathbf{x_{n_d+1}} & \cdots & \mathbf{x_m} \end{bmatrix}, \tag{5}$$

where $\mathbf{H} \in \mathbb{R}^{(n\times n_d)\times(m-n_d+1)}$. DMD can be subsequently be performed on the data set contained in $\mathbf{H}$. DMD performed in this manner is often referred to as Hankel-DMD [17].

2.2. Relationship between DMD and the Koopman operator and the motivation for the proposed approach

Koopman analysis can be used to represent the evolution of the observables of a nonlinear dynamical system in terms of a (generally) infinite dimensional linear operator [26]. Specifically, consider a discrete dynamical system of the form

$$\mathbf{z_{k+1}} = \mathbf{F}(\mathbf{z_k}), \tag{6}$$

where $\mathbf{F}$ denotes the dynamics of the (possibly nonlinear) mapping from $\mathbf{z_k} \to \mathbf{z_{k+1}}$ and $\mathbf{z_k} \in \mathcal{Z} \subseteq \mathbb{R}^m$ is the state. The evolution of the observables under this mapping can be described according to

$$\mathcal{K}\Phi(\mathbf{z_k}) = \Phi(\mathbf{F}(\mathbf{z_k})), \tag{7}$$

where $\mathcal{K}$ is the Koopman operator, which operates on the vector space of observables $\Phi(\mathbf{z_i}) = [x_i^1 \ \ldots \ x_i^n]^T \in \mathbb{R}^n$. Note that n and m are not necessarily the same. Owing to the linearity of the composition operator, the Koopman operator is linear, but generally infinite dimensional [26–28]. As such, its action can be decomposed in terms of eigenmodes/eigenvalues as

$$\Phi(\mathbf{z_{k+1}}) = \mathcal{K}\Phi(\mathbf{z_k}) = \sum_{j=1}^{\infty} \lambda_j \psi_j(\mathbf{z_k})\mathbf{v_j}, \tag{8}$$

where λ_j is a Koopman eigenvalue associated with the Koopman eigenfunction $\mathbf{v_j}$. It immediately follows that

$$\Phi(\mathbf{z_{k+n}}) = \sum_{j=1}^{\infty} \lambda_j^n \psi_j(\mathbf{z_k})\mathbf{v_j}. \tag{9}$$

Using (8) to represent the evolution of the system observables, the critical challenge is in identifying a suitable finite basis to represent the infinite dimensional Koopman operator [29]. DMD as described in Section 2.1 is one standard approach; the eigenvalues and eigenvectors of the inferred matrix $\mathbf{A}$ from Eq. (4) provide an approximation of the true Koopman eigenvalues and eigenvectors of the Koopman operator $\mathcal{K}$ [20]. This is true regardless of whether the observables are lifted to a higher dimension, for instance, as done when using extended

DMD [24] or Hankel DMD [17]. However, DMD does not explicitly consider the Koopman eigenfunctions as part of the model identification procedure, rather, it focuses on identifying an approximation of the Koopman operator that minimizes the error from one step to the next. The eigenmodes are only considered after the approximation of the Koopman operator has already been obtained. Given that eigenvectors are generally very sensitive to small matrix perturbations, it can be difficult to determine whether the resulting eigenmodes obtained from DMD are truly important to the dynamics. Furthermore, because DMD only provides a fit to the data that minimizes the error between steps, the prediction (9) will not always remain accurate as the prediction horizon, n, increases.

3. Problem formulation and general approach

3.1. Problem description

In this work, we propose and implement an artificial neural network-based strategy for inferring Koopman eigenmodes from data that explicitly considers information about the Koopman eigenvalues during the model identification stage. We propose an artificial neural network based strategy which is trained using mini batch gradient descent to find a representation for the Koopman eigenfunctions that is consistent with *a priori* knowledge about the system on sets containing the current observable of the system $\boldsymbol{\Phi}(\mathbf{z_i}) = [x_i^1 \; x_i^2 \; \dots \; x_i^n]^T$ as input and the next observables $\boldsymbol{\Phi}(\mathbf{z_{i+1}}) = [x_{i+1}^1 \; x_{i+1}^2 \; \dots \; x_{i+1}^n]^T$ as output. Furthermore, explicit constraints on the Koopman eigenvalues are enforced by our algorithm. In contrast to DMD, the proposed approach incorporates information about the Koopman eigenfunctions themselves into the model identification procedure. As illustrated in the examples provided in Section 4, this approach procedure gives allows for better qualitative and quantitative predictions of system behavior, especially as the prediction horizon increases.

The proposed approach can be viewed as learning the dominant Koopman eigenfunctions directly subject to a set of application specific constraints. By contrast, DMD provides a least squares fitting for the action of the Koopman operator from which Koopman eigenfunctions can subsequently be inferred.

3.2. Reframing the eigendecomposition for the neural network

Considering the evolution of observables according to (7), we seek a finite-dimensional linear approximation for the action of the Koopman operator on the model observables. The proposed approach aims to learn the eigendecomposition of this matrix, i.e., its eigenvalues as well as the left and right eigenvectors. In the description of the proposed approach, for notational convenience, below we let $\mathbf{x}^+ = \mathcal{K}\mathbf{x}$, i.e., where the set of observables $\mathbf{x} \in \mathbb{R}^n$ maps to $\mathbf{x}^+ \in \mathbb{R}^n$ under the action of the Koopman operator. Using this notation, we are trying to find the following approximation for the action of the Koopman operator:

$$\begin{aligned}\mathbf{x}^+ &= \mathcal{K}\mathbf{x} \\ &\approx \mathbf{V}'\boldsymbol{\Lambda}'\mathbf{W}'\mathbf{x} \\ &= \mathbf{A}_{\mathbf{neu}}\mathbf{x},\end{aligned} \tag{10}$$

where $\mathbf{A}_{\mathbf{neu}} \in \mathbb{R}^{n\times n}$ is a finite dimensional approximation of the Koopman operator, and $\mathbf{V}'$, $\boldsymbol{\Lambda}'$ and $\mathbf{W}'$ represent the right eigenvectors, eigenvalues and the left eigenvectors, respectively, of $\mathbf{A}_{\mathbf{neu}}$ arranged as matrices which are learned by the neural network. The goal of this work is to obtain the matrices $\mathbf{V}'$, $\boldsymbol{\Lambda}'$ and $\mathbf{W}'$ from data which provide an approximation for the action of the Koopman operator — this is different from standard DMD approaches where the linear approximation of the Koopman operator is obtained from data and then the eigendecomposition is obtained from this matrix approximation

Of course, eigenvalues and eigenvectors are in the set of complex numbers. Although complex-valued neural networks (CVNNs) [30,31] can be used for dealing with complex-valued eigendecompositions, the implementation is generally more computationally intensive and nontrivial when compared with traditional neural networks. Instead, we will use a conventional real-valued neural network and the issue of complex numbers can be ameliorated with appropriate transformations for complex eigenvalues using a Block-II form.

To proceed, let ϑ be the total number of eigenvalue/eigenvector pairs in the eigendecomposition and let $\vartheta_1 \leq \vartheta$ be the total number with nonzero imaginary part. For any pair of complex eigenvalues λ_a and $\lambda_b = \lambda_a^*$, with right and left eigenvector pairs $(\mathbf{v_a}, \mathbf{v_b})$ and $(\mathbf{w_a}, \mathbf{w_b})$, respectively, a Block-II representation is considered

$$\bar{\boldsymbol{\Lambda}}_{\mathbf{a,b}} = \begin{bmatrix} \mathrm{Re}(\lambda_a) & \mathrm{Im}(\lambda_a) \\ -\mathrm{Im}(\lambda_b) & \mathrm{Re}(\lambda_b) \end{bmatrix}. \tag{11}$$

In a similar manner, let

$$\bar{\mathbf{V}}_{\mathbf{a,b}} = \begin{bmatrix} \mathrm{Re}(\mathbf{v_a}) & \mathrm{Im}(\mathbf{v_b}) \end{bmatrix}, \bar{\mathbf{W}}_{\mathbf{a,b}} = \begin{bmatrix} \mathrm{Re}(\mathbf{w_a}^T) \\ \mathrm{Im}(\mathbf{w_b}^T) \end{bmatrix}. \tag{12}$$

Considering the Block-II form for complex eigenvalues given in Eqs. (11) and (12), one can write

$$\mathbf{L} = \hat{\mathbf{W}}\mathbf{x} = \begin{bmatrix} \bar{\mathbf{W}}_{\mathbf{1,2}} \\ \vdots \\ \bar{\mathbf{W}}_{\vartheta_1-1,\vartheta_1} \\ \mathbf{w}_{\vartheta_1+1}^T \\ \vdots \\ \mathbf{w}_{\vartheta}^T \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}, \tag{13}$$

where $\mathbf{L} \in \mathbb{R}^{\vartheta}$ is the product of $\mathbf{x}$ and the matrix $\hat{\mathbf{W}} \in \mathbb{R}^{\vartheta\times n}$ which contains the Block-II form for complex left eigenvectors and real left eigenvectors stacked together. This intermediate vector $\mathbf{L}$ can further be transformed as

$$\mathbf{L}^+ = \hat{\mathbf{\Lambda}}\mathbf{L} = \begin{bmatrix} \bar{\mathbf{\Lambda}}_{\mathbf{1,2}} & 0 & \cdots & \cdots & \cdots & 0 \\ 0 & \ddots & \ddots & & & \vdots \\ \vdots & \ddots & \bar{\mathbf{\Lambda}}_{\vartheta_1-1,\vartheta_1} & \ddots & & \vdots \\ \vdots & & \ddots & \lambda_{\vartheta_1+1} & \ddots & \vdots \\ \vdots & & & \ddots & \ddots & 0 \\ 0 & \cdots & \cdots & \cdots & 0 & \lambda_{\vartheta} \end{bmatrix} \begin{bmatrix} \mathbf{L}_{\mathbf{1,2}} \\ \vdots \\ \mathbf{L}_{\vartheta_1-1,\vartheta_1} \\ L_{\vartheta_1+1} \\ \vdots \\ L_{\vartheta} \end{bmatrix}. \tag{14}$$

Here, $\hat{\mathbf{\Lambda}} \in \mathbb{R}^{\vartheta\times\vartheta}$ is a diagonal matrix containing the real eigenvalues as well as complex eigenvalues of the underlying dynamical system in Block-II form and $\mathbf{L}^+ \in \mathbb{R}^{\vartheta}$ is the update of the intermediate vector $\mathbf{L}$. In order to predict the next observable $\mathbf{x}^+$ from this updated vector, one can take its product with another matrix containing information of right eigenvectors to get

$$\mathbf{x}^+ = \hat{\mathbf{V}}\mathbf{L}^+ = \begin{bmatrix} \bar{\mathbf{V}}_{\mathbf{1,2}} & \cdots & \bar{\mathbf{V}}_{\vartheta_1-1,\vartheta_1} & \mathbf{v}_{\vartheta_1+1} & \cdots & \mathbf{v}_{\vartheta} \end{bmatrix} \begin{bmatrix} \mathbf{L}^+_{\mathbf{1,2}} \\ \vdots \\ \mathbf{L}^+_{\vartheta_1-1,\vartheta_1} \\ L^+_{\vartheta_1+1} \\ \vdots \\ L^+_{\vartheta} \end{bmatrix}. \tag{15}$$

The real right eigenvectors and Block-II form for complex right eigenvectors are stacked together in the matrix $\hat{\mathbf{V}} \in \mathbb{R}^{n\times\vartheta}$. For the purpose of this work, the values in $\hat{\mathbf{W}}$, $\hat{\mathbf{\Lambda}}$ and $\hat{\mathbf{V}}$ from Eqs. (13), (14) and (15) respectively, will be learned using an artificial neural network approach. Once these are learned, a system matrix approximation of the form $\mathbf{x}^+ = \mathbf{A}_{\mathbf{neu}}\mathbf{x}$ as in (10) where

$$\mathbf{A}_{\mathbf{neu}} = \hat{\mathbf{V}}\hat{\mathbf{\Lambda}}\hat{\mathbf{W}}. \tag{16}$$

can be obtained.

3.3. Approximating Koopman eigenfunctions using neural networks

Given the current observable measurement, $\mathbf{x}$, of the system (6), the main objective of the proposed strategy is to learn the matrices

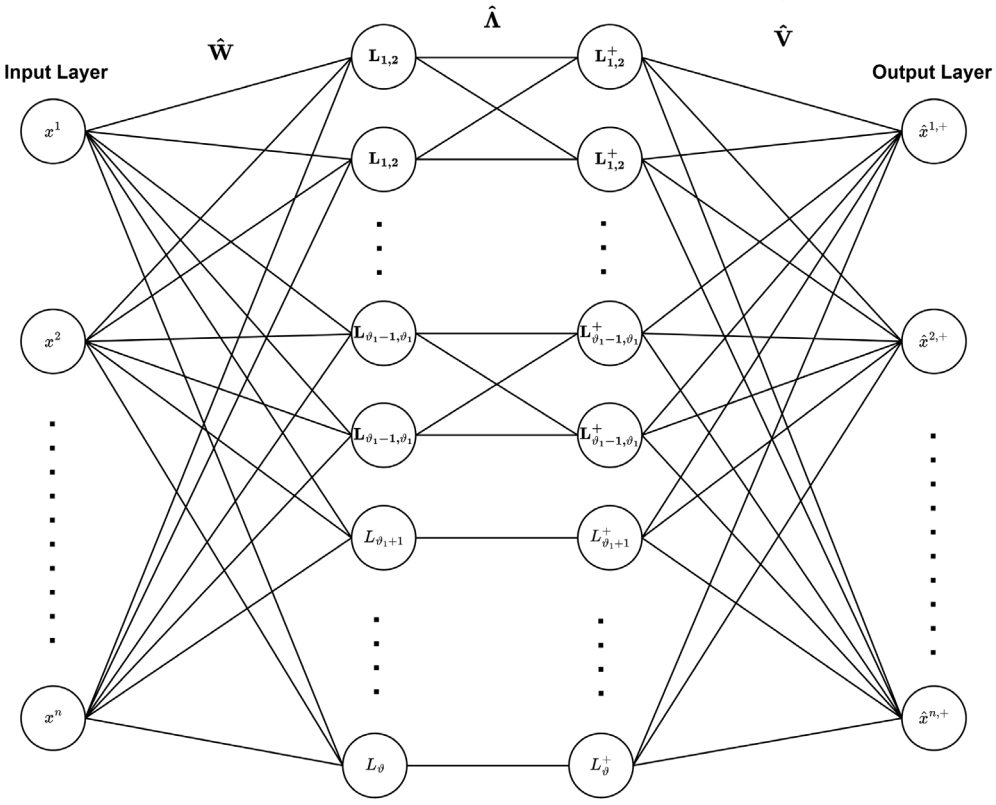

Fig. 1. General architecture for the proposed neural network based identification strategy.

from Eqs. (13)–(15) for a given system in such a way that we get an accurate approximation of the next observable measurement $\mathbf{x}^+$ by considering an optimization subject to different system constraints intended to reflect application specific knowledge. Here, the predicted next observable from the neural network will be denoted by $\hat{x}^+$ and the true next observable (ground truth) will be denoted by $\mathbf{x}^+$.

In order to learn the approximation of the underlying system dynamics, our strategy employs a multi-layer feed-forward artificial neural network. A simplified layout of the network structure is shown in Fig. 1. The architecture is based on the mathematical formulation discussed in Section 3.2. The input layer of the neural network takes the current observable $\mathbf{x}$ of the system and feeds it to the first hidden layer where it is multiplied by $\hat{\mathbf{W}}$ which contains information regarding the left eigenvectors of the system. The unknown coefficients within $\hat{\mathbf{W}}$ correspond to the weights of the first hidden layer and the product results in the intermediate vector $\mathbf{L}$. This intermediate vector is fed to a second hidden layer with weights corresponding to $\hat{\mathbf{\Lambda}}$ that is formed by combining real eigenvalues as well as complex eigenvalues converted to Block-II form. Subsequently, the update to the intermediate vector $\mathbf{L}^+$ is obtained. The number of hidden layers and ultimately, the number of intermediate vectors in this hidden layer part of the network depend upon the number of Koopman eigenmodes to be approximated. An additional hidden layer is added to this part for each pair of eigenvalues considering whether they are real or complex. The intermediate vector updates are then concatenated and fed forward to the output layer with weights from $\hat{\mathbf{V}}$ learning information about the right eigenmodes to obtain the approximation of the next observable $\hat{\mathbf{x}}^+$.

This predicted next observable can then be compared to the ground truth $\mathbf{x}^+$ obtained from observations of the underlying system. To compute the error between the prediction and the ground truth, both are fed to the following loss function

$$\mathcal{L}_{total} = \frac{\gamma_1}{\zeta}\sum_{i=1}^{\zeta}(\mathbf{x}_\mathbf{i}^+ - \hat{\mathbf{x}}_\mathbf{i}^+)^2 + \gamma_2(\hat{\mathbf{W}}\hat{\mathbf{V}} - \mathbf{I}) + \gamma_3\mathbf{H}(\hat{\mathbf{\Lambda}}, \hat{\mathbf{W}}, \hat{\mathbf{V}}). \tag{17}$$

The loss function consists of three terms; the first term is the MSE loss followed by the second term for orthogonality loss which regularizes the weights to learn the true Koopman eigenmodes that are orthogonal to each other, i.e., $\hat{\mathbf{W}}\hat{\mathbf{V}} = \mathbf{I}$ where $\mathbf{I}$ is the identity matrix. The orthogonality constraint is necessary to ensure that the columns of $\hat{\mathbf{V}}$ and rows of $\hat{\mathbf{W}}$ remain right and left eigenvectors of $\mathbf{A}_{\mathbf{neu}}$, and hence, approximate the Koopman eigenfunctions. Finally, the third term $\mathbf{H}$ includes a set of additional constraints, for instance, that might reflect domain specific knowledge. Here, ζ is the number of measurement pairs used for training and $\gamma_1, \ldots, \gamma_3$ are the weights for each of the penalty. The domain knowledge based regularization constraints, represented by the third term, can be added to the network depending upon the application, for example, magnitude and frequency based constraints. For instance, the magnitudes of the resulting eigenvalues can be penalized from being far from the unit circle to prioritize slowly decaying eigenmodes. Alternatively, penalties can be placed on the argument of the eigenvalue for deviating from set values to prioritize modes with specific frequencies. The weights of the trainable layers, i.e., $\hat{\mathbf{V}}$, $\hat{\mathbf{\Lambda}}$ and $\hat{\mathbf{W}}$ are updated using backpropagation. For the examples presented in this work, it is assumed that there is a combination of real and complex eigenvalues and the eigenvectors obtained for the underlying system. The complex eigenvalues and eigenvectors are converted into Block-II form to be learned through a real-valued neural network. Mini-batch gradient descent is used for training to take multiple observable pairs into account simultaneously so that the resulting system approximation is able to generalize for the underlying model efficiently.

Linear activation functions are used for each layer in the artificial neural network as the transformations to predict the next observable from the current observable using the learned weights. Once these weights from Eqs. (13)–(15) are learned, they can be utilized to approximate the system matrix for the underlying model using the form in Eq. (16). Note that the eigenvalues obtained by the proposed strategy are for a discrete dynamical system.

A simplified layer level diagram for the strategy's neural network architecture is illustrated in Fig. 1. The input layer of the network is comprised of the current observable of the dynamical system, $\mathbf{x}$. The input layer component is passed simultaneously through a set of parallel hidden layers to transform the current observable into intermediate vectors $\mathbf{L}$ with the coefficients comprising $\hat{\mathbf{W}}$ as the corresponding weights. A set of updates to the intermediate vectors, $\mathbf{L}^+$, is obtained by feeding the intermediate vectors to a second set of parallel hidden layers having $\hat{\mathbf{\Lambda}}$ from (14) as their weights. The resulting $\mathbf{L}^+$ vectors are then concatenated into a single tensor and then passed to the output layer of the network architecture. The output layer transforms these updated intermediate vectors into the predicted next observable $\hat{\mathbf{x}}^+$ by taking their dot product with the weights from $\hat{\mathbf{V}}$. The process is repeated for all pairs of current and next observable measurements to get the corresponding next observable predictions which are then compared with the ground truth to compute the error and update the network weights accordingly to learn a generalizable Koopman eigenfunction approximation through backpropagation. The next section details the type of penalties enforced on the Koopman eigenvalues in this work and the intuition behind them.

3.4. Domain knowledge constraints on Koopman eigenvalues

Koopman eigenvalues learned through training the neural network on observable data are constrained through our proposed approach in such a way that the resulting system approximation is able to emulate and replicate the underlying model dynamics more accurately as compared to traditional techniques like DMD. In this work, both magnitude based and frequency based constraints are applied to the examples in the form of function $\mathbf{H}(\hat{\mathbf{\Lambda}}, \hat{\mathbf{W}}, \hat{\mathbf{V}})$ from Eq. (17). Magnitude based constraints force the magnitude of learned Koopman eigenvalues to be close to one. These constraints are derived from the fact that eigenvalues close to the unit circle are slow decaying and hence, can capture the slow decaying oscillatory dynamics of systems with limit cycles more efficiently. Similarly, frequency based constraints penalize a given eigenvalue for having an argument far from some specified value, prioritizing modes with prespecified frequencies. Specific steps and heuristics required for the neural network training process and the network architecture are detailed in the next section.

3.5. Implementation of the Koopman eigenmode identification strategy using the neural network

Details for the implementation of the Koopman eigenmode identification strategy described in Section 3.3 are provided here. The neural network is coded in python by utilizing Tensorflow and Keras; Keras is a library for artificial neural networks which acts as an interface for Tensorflow, a machine learning platform in python [32]. In terms of training data, our implementation follows the traditional supervised learning approach with pairs of current observables and the corresponding next observables used for learning the approximation for the Koopman eigenfunctions of the underlying system. The overall steps for the identification strategy are outlined below. It is assumed that observable data from the underlying model is accessible and a dataset containing the current observables $\mathbf{x_1}, \ldots, \mathbf{x_n}$ and the corresponding next observables $\mathbf{x}_1^+, \ldots, \mathbf{x}_n^+$ has already been obtained. Depending upon the number of observables, time-delay embeddings are generated to lift to a higher dimension [17]; details for the implementation are given in Section 2.1. Also, DMD is used to obtain an initialization for the neural network weights.

Step (1) Initially, specify the number of eigenvalues and eigenvectors to be considered; as a general heuristic, per the structure of $\hat{\mathbf{\Lambda}}$ from (14), a combination of complex eigenvalue pairs and real eigenvalues are considered. Standard DMD is applied to the training set and the resulting eigenvalue and eigenvector approximation is then used for initializing the weights of the neural network after the complex values have been converted into the Block-II matrix form specified by (11) and (12).

Step (2) Structure the artificial neural network based on the number of real eigenvalues and complex eigenvalue pairs; for each complex eigenvalue pair and real eigenvalue, an additional hidden layer for $\hat{\mathbf{\Lambda}}$ and $\hat{\mathbf{W}}$ is considered along with a single hidden layer for $\hat{\mathbf{V}}$ per the architecture specified in Fig. 1. Linear activation functions are used in all network layers.

Step (3) Predictions for the next observables, $\hat{\mathbf{x}}^+$, are generated at the output layer of the network.

Step (4) Specify an optimizer, the loss function, and the learning rate. Define the additional constraints on the weights of the neural network. The multi-objective loss function from (17) is used for the approach.

Step (5) In each epoch during training, the current observable is fed to the neural network to compute the next observable prediction while using the error between the predicted next observable and the ground truth to update the weights of the neural network. This process continues until the training loss converges.

Step (6) Once the neural network is trained, the learned weights can be extracted to obtain an approximation of the system matrix that can predict future observable given the current observable.

A few general notes about the implementation of the training process described above are given below. The initialization in Step 4 is done by applying DMD to the data set and then, computing the eigenvalues and eigenvectors of the resulting system matrix approximation. The resulting eigenvalues and eigenvectors are generally a combination of complex eigenvalue pairs and real eigenvalues. These complex DMD eigenvalues and eigenvectors are then converted to the Block-II matrix form according to Eqs. (11)–(12). Based on Eqs. (13)–(15), they are then stacked together with the real eigenvalues and eigenvectors to be used as the weight initialization of the neural network. To yield a more generalizable approximation of the underlying system matrix, mini-batch stochastic gradient descent is utilized to train the network on multiple batches using the data.

4. Results

4.1. Spike rates of neural populations

For illustration purposes, we start by considering a population of spiking thalamic neurons for Koopman eigenmode identification for the underlying model through our neural network-based strategy. The equations for the dynamical system are [33]:

$$\begin{aligned} C_m \dot{V_k} &= -I_{\mathrm{L}}(V_k) - I_{\mathrm{Na}}(V_k, h_k) - I_{\mathrm{K}}(V_k, h_k) - I_{\mathrm{T}}(V_k, r_k) \\ &\quad + I_k^b - 0.2 \sum_{j=1}^{N} (V_k - V_j), \\ \dot{h_k} &= (h_\infty(V_k) - h_k)/\tau_h(V_k), \\ \dot{r_k} &= (r_\infty(V_k) - r_k)/\tau_r(V_k), \\ y(t) &= \frac{1}{N} \sum_{j=1}^{N} (V_j(t)), \end{aligned} \tag{18}$$

for $k = 1, \ldots, N$. The population in the model above considers $N = 10$ total neurons where V_k, h_k, and r_k represent the transmembrane voltage and two gating variables, respectively, that determine the ionic currents for neuron k. I_k^b is defined as the baseline current of the kth neuron computed by $I_k^b = 5 + 0.35k$. Other important parameters include $C_m = 1\mu\mathrm{F/cm}^2$ and $i_k(t) = \upsilon_k u(t)$ where the model incorporates the sensitivity parameter υ_k through $\upsilon_k = 1 + 0.05k$. A full description of all the ionic currents and gating variables is given in [33]. We take $y(t)$ as the observable for the neural population model. Fig. 2 illustrates that in this example, we are including the transient dynamics of the neural population as they converge to a periodic orbit.

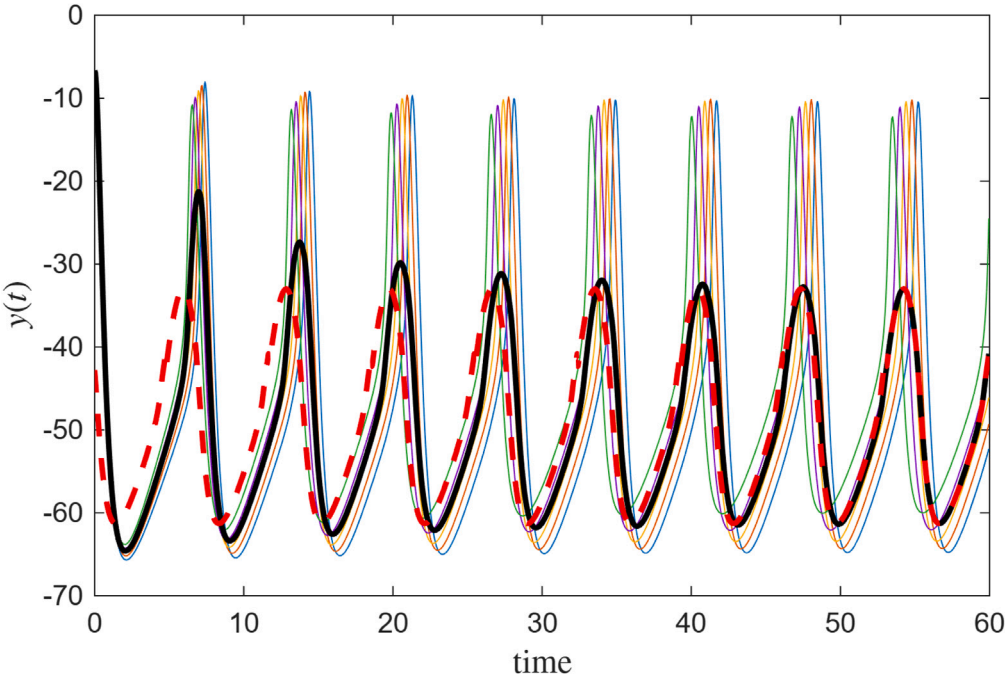

Fig. 2. Dynamics of the population of spiking neurons from (18) are illustrated. Five representative voltage traces $V(t)$ are being shown along with the average value taken as the observable $y(t)$ (black line). For comparison, the red dashed line shows the observable once the system has converged to the periodic orbit.

As an initial step, data is generated from the neural population model for $t \in [0, 120]$ and then split equally into training and test data. Time-delay embedding of length $n_d = 100$ is used on both training and test data, as described in Section 2.1 to lift to a higher dimension. This results in two matrices for the current observable **X** and the next observable **Y** and two for the test data, i.e., **X1** and **Y1** respectively. DMD is then applied to the current observable and the next observable by specifying the number of eigenvalues ϑ to be considered; in the case of the neural population model, $\vartheta = 20$. By following the procedure highlighted in Section 2.1, an approximation of the system matrix $\mathbf{A_{dmd}}$ for the underlying model is obtained through least-square minimization of the observable data. A combination of real and complex eigenvalues and eigenvectors are computed by decomposing $\mathbf{A_{dmd}}$. The complex eigenvalues and eigenvectors are then converted to the Block-II matrix form before getting stacked together with the real eigenvalues and eigenvectors according to Eqs. (13)–(15).

Based on the number of eigenvalues chosen, a neural network architecture is constructed using dense layers; the architecture follows the structure shown in Fig. 1. The weights of the neural network are then initialized by the DMD eigenvalues and eigenvectors in the form given by Eqs. (13)–(15). Once the weights are initialized, training is done by using ADAM as the optimizer with the learning rate set at 5×10^{-7}. The custom loss function of the form (17) used here is

$$\mathcal{L}_{total} = \frac{3}{\zeta}\sum_{i=1}^{\zeta}(\mathbf{x_i^+} - \hat{\mathbf{x}}_\mathbf{i}^+)^2 + (\hat{\mathbf{W}}\hat{\mathbf{V}} - \mathbf{I}) + 3 \times 10^{-3}\sum_{k=1}^{\vartheta}(|\hat{\lambda}_\mathbf{k}| - \mathbf{1}), \tag{19}$$

where ζ gives the number of current and next observable pairs used for training while ϑ represents the number of eigenvalues. The implementation follows the same steps highlighted in Section 3.5. Additional magnitude constraints are enforced on all the eigenvalues of the network by forcing their magnitude to be close to 1. This is essential so that the learned system approximation is able to emulate the model dynamics which evolve on a periodic limit cycle accurately by learning slow decaying Koopman eigenvalues close to the unit circle.

The obtained eigenvalues from both the DMD and neural network strategy are illustrated on the complex plane in Fig. 3. Panel A shows the whole unit circle in the complex plane; the DMD eigenvalues are represented by red crosses and the neural network ones are given by yellow circles. All of the eigenvalues lie to the right hand side of the complex plane close to the unit circle. Panel B shows the placement of the eigenvalues in more detail; as seen from the figure, almost all the eigenvalues from our proposed strategy are closer to the unit circle as compared to the ones obtained from DMD. This is because of the magnitude constraint applied to the neural network eigenvalues while training based on the assumption that by getting the magnitude of eigenvalues close to 1, the resulting system approximation will give better long term observable predictions by having slowly decaying eigenvalues.

After obtaining the system matrices for both DMD and the proposed strategy, an initial state is chosen randomly using the initial step value generated through $\text{initstep} = a_1 + (b_1 - a_1)\text{rand}(0, 1)$ with $a_1 = 0$ and $b_1 = 1000$. This initial step is then rounded and used to choose the corresponding initial state from the data. Given the initial state and corresponding observable, future observable estimates are obtained for both DMD and the neural network which are then compared with the ground truth through mean squared error. The future estimates are obtained by using

$$\hat{\mathbf{x}}_\mathbf{k} = \mathbf{A}_\mathbf{neu}^\mathbf{k}\mathbf{x_0}, \tag{20}$$

where $\mathbf{A_{neu}}$ is the system approximation learned by the neural network by using Eq. (16) and $\mathbf{x_0}$ is the initial observable. This process is repeated for 200 trials; each trial is run for 4000 timesteps and has a different initial state. The resulting MSE for all the timesteps is averaged over the trials for both the proposed strategy and DMD. Obtained results are shown in Fig. 4 which illustrate the average error trends for both DMD and the proposed neural network approach. The full model exhibits oscillatory dynamics with a period of approximately 6.72 or 672 time steps. On average, for the first 1300 timesteps, DMD has a lower MSE value (shown in blue) than the neural network (shown in red). However, as predictions are made for subsequent timesteps, the MSE for the neural network approach is significantly less than DMD.

We note that though the eigenvalues of the proposed approach are not substantially different than those obtained when using DMD, the long term state predictions from Fig. 4 are markedly better than

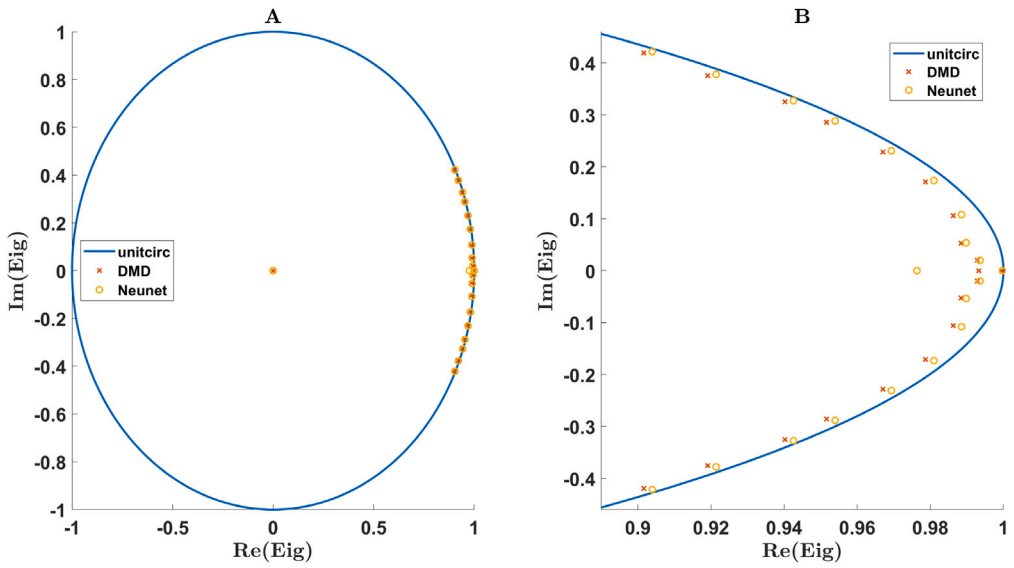

Fig. 3. Placement of eigenvalues obtained from system approximation of spiking neuron population dataset through DMD and our proposed neural network strategy. Panel A shows the unit circle with most of the eigenvalues on the right hand side whereas Panel B shows a detailed view of eigenvalues from both the DMD and neural network approach. DMD eigenvalues are given in red while yellow represents the neural network eigenvalues.

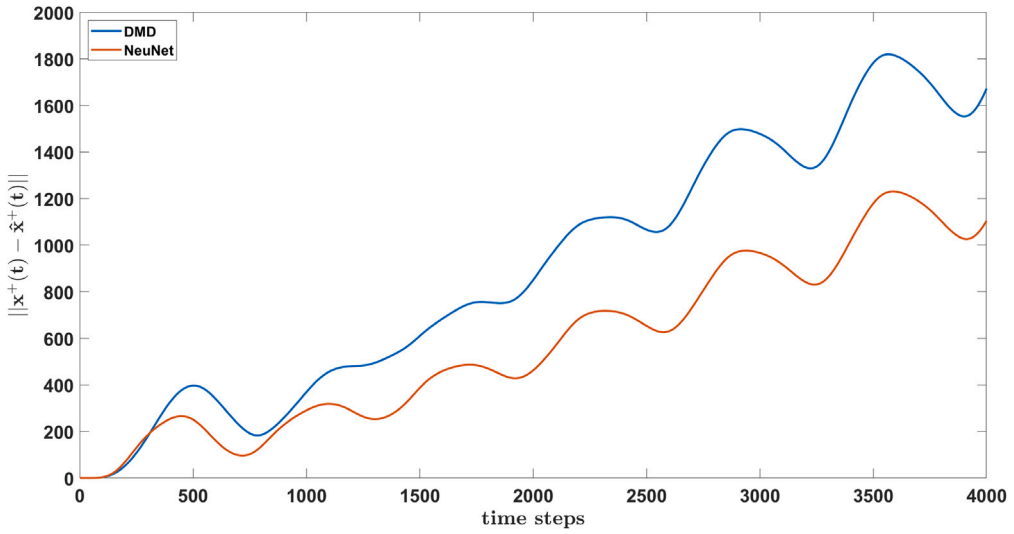

Fig. 4. Mean squared error plot for our proposed strategy is compared with DMD for the spiking neuron population model. The plot gives the error difference between the predicted next observables and the ground truth given the current observable. MSE trend for neural network strategy is given in red while blue shows the trend for DMD.

those obtained using standard DMD. Even though standard DMD yields a good approximation of the eigenmodes of the system, the small adjustments obtained when using the proposed method lead to substantial improvements in the long term state prediction. Notice that DMD does provide a slightly better short term prediction of the system state over the first 300 time steps. This is not surprising since DMD provides an estimate of the action of the Koopman operator which minimizes the error over a single time step. By focusing the structure of the eigendecomposition instead of simply focusing on the short term accuracy, the proposed approach is able to make substantially better long term predictions.

4.2. Mixed frequency synthetic image dataset

In this example, the proposed approach is tested on a synthetic dataset with multiple modes and frequencies to evaluate its performance against DMD. A synthetic movie is generated by creating frames through mixing different frequencies as described in [34]. The movie consists of 1000 frames in total for a duration of 10 seconds and each frame is $80 \times 80 = 6400$ pixels with a sampling rate of $dt = 0.01$ seconds. The movie is derived by mixing three different frequencies having varying spatial distributions as shown in Fig. 5. For the first half of the movie, mode 1 is dominant with a oscillating frequency of 5.55 Hz and an amplitude of 1. The second mode only exists from 3 to 7 seconds in the movie with amplitude 1 and a frequency of 0.9 Hz. The last mode has an slow oscillating frequency of 0.15 Hz for the entire duration of the movie with an amplitude of 0.5.

$$\text{mode}_i = A_i u_i \sin(2\pi f_i t). \tag{21}$$

Eq. (21) shows how each of the three modes are computed with the amplitude given by A_i and the frequency given by f_i for $i = 1, 2$ and 3. Also, u_i gives the distribution of the modes on the image frame. Gaussian white noise is added to every pixel randomly in the movie by using a normal distribution with a magnitude of 0.1.

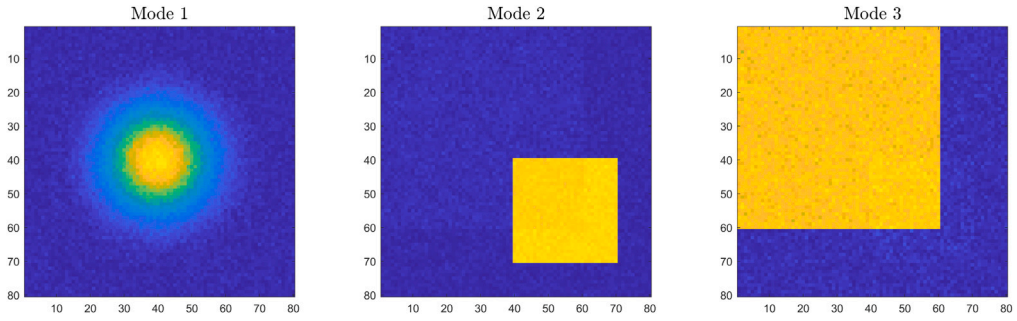

Fig. 5. Figure shows the three modes used to generate the synthetic image dataset to compare the proposed strategy with standard DMD and mrDMD. The modes have an oscillating frequency of 5.55, 0.9 and 0.15 Hz and overlap each other.

For this example, an additional test image dataset is generated by using mode 3 from Eq. (21) for $t = 10, \dots, 20$. Mean is computed for both the testing and training dataset separately and then subtracted from the original datasets. Proper orthogonal decomposition is then applied to reduce the dimensionality of the data by representing both the test and training data, retaining the 10 most important modes which contain more than 90 percent of the information. Resulting POD reduced data is then lifted to a new basis by generating time-dependent chunks through time-delay embeddings that can be stacked together in matrices to increase the number of observables from 10 to 50 for both current and next observable in training and testing data by using Eq. (5). For comparison purposes, we also implement multi-resolution DMD (mrDMD) [16] to predict dynamics given the current observable.

The authors in [34] present multi-resolution DMD or mrDMD as the optimal approach to deal with datasets containing multiple modes and frequencies like this example. The mrDMD approach splits the data in halves recursively based on the specified number of levels and thus separates fast modes from the slow decaying modes. The DMD algorithm is then applied at each level, and the overall solution is computed by summing up all the individual level components. Using the training matrices **X1** and **X2**, DMD approach is applied to the given data by prespecifying the number of eigenvalues to be considered for system approximation; for this example, $\vartheta = 30$ are considered. The same data matrices are fed to the multi-resolution DMD algorithm to generate approximations of underlying dynamics at specified levels; for the purpose of this illustration, number of levels are 6 with the same number of eigenvalues as standard DMD. The mrDMD algorithm is adapted from [16] with some minor changes like applying POD on the data to remove noise and generating system approximation in matrix form at each level of mrDMD. Once the DMD matrix $\mathbf{A_{dmd}}$ is obtained, it is decomposed into real and complex eigenvalues and corresponding eigenvectors. The complex eigenvalues and eigenvectors are converted into Block-II form to separate their real and imaginary components and then, put together with real eigenvalues using Eqs. (11)–(13) so that they can be utilized for our proposed strategy.

Based on the architecture presented in Fig. 1, a neural network framework is constructed considering the number of eigenvalues whose training weights are initialized using the eigenvalues and eigenvectors from DMD converted to the form from Eqs. (13)–(15). A loss function

$$\mathcal{L}_{total} = \frac{3}{\zeta}\sum_{i=1}^{\zeta}(\mathbf{x_i^+} - \hat{\mathbf{x}}_\mathbf{i}^+)^2 + (\hat{\mathbf{W}}\hat{\mathbf{V}} - \mathbf{I}) + 6\times 10^{-3}\sum_{k=1}^{\vartheta}(|\hat{\lambda}_\mathbf{k}| - \mathbf{1}), \tag{22}$$

is constructed. The final term is included to move the eigenvalues close to the unit circle. By getting the magnitude of these eigenvalues close to 1, the learned system approximation by our proposed strategy can capture the oscillating dynamics of the underlying system more efficiently than standard DMD as the learned Koopman eigenvalues will be slow decaying. Also, ϑ represents the number of eigenvalues whereas ζ gives the number of current and next observable pairs used for training. The learning rate for the ADAM optimizer is set as 1×10^{-3}; the overall implementation follows the same steps highlighted in Section 3.5.

Distribution of eigenvalues on the complex plane from the proposed strategy, mrDMD and DMD is illustrated in Fig. 6. The overall view of how these eigenvalues are placed inside the unit circle is presented in Panel A with yellow circles representing neural network eigenvalues, blue pluses and green stars for different levels of mrDMD and red crosses for DMD. As seen from Panel B in the figure which gives a magnified view of the eigenvalue distribution, most of the eigenvalues obtained from the proposed approach are significantly closer to the unit circle when compared to the ones from DMD and mrDMD. This is the effect of the magnitude based constraint, from Eq. (22), applied during training which pushes the learned eigenvalues to be close to the unit circle. The learned eigenvalues are thus more slowly decaying and the resulting system approximation from the proposed strategy will give better long term observable predictions. Eigenvalues for mrDMD are only shown for the first level and second level for comparison purposes; these eigenvalues are similar to the ones obtained from DMD.

To compare the system approximation matrices computed for DMD, mrDMD and the neural network strategy, an initial state is chosen randomly from the test dataset by using $\text{initstep} = a_2 + (b_2 - a_2)\text{rand}(0, 1)$ with $a_2 = 0$ and $b_2 = 200$. This initial step specifies which instance from the data is chosen as the initial current state. Comparison is done by computing the difference between the ground truth and the predicted next observables through MSE from both variations of DMD and neural network approach for 700 timesteps. Given the initial observable $\mathbf{x_0}$, the predicted observables are obtained by using Eq. (20). This is repeated for 200 trials with each trial having different initial state. Computed MSE is averaged over all the trials for both strategies; the obtained average error trends are shown in Fig. 7. From the observed trends, it can be seen that the error for the proposed strategy is notably less than the error computed for both DMD and mrDMD except for the first 100 timesteps where the trends are opposite with significantly less difference.

The proposed strategy is able to outperform standard DMD by predicting the future observables of the system more accurately. For this specific test dataset, although the proposed approach does perform better than mrDMD in terms of MSE, it should be noted that only one frequency, i.e., mode 3 from training data, is dominant for the whole data set. Multi-resolution DMD is primarily used for mode identification by splitting the data at different levels and identifying dynamic modes for each specified level as seen in [16,35,36]. Therefore, multi-resolution DMD is better suited for data involving multiple spatial distributions and hence, might give better performance in terms of MSE when compared to the proposed strategy at individual levels. Also, compared to both DMD and mrDMD, the key advantage of our proposed method is that it allows utilization of prior domain knowledge to steer the method to minimize prediction error further than the vanilla DMD or mrDMD where the sole objective is to minimize the loss for the observable data. Similar to the results from the previous section, this domain knowledge allows the proposed method to significantly outperform previously proposed model identification strategies when considering the long term prediction error.

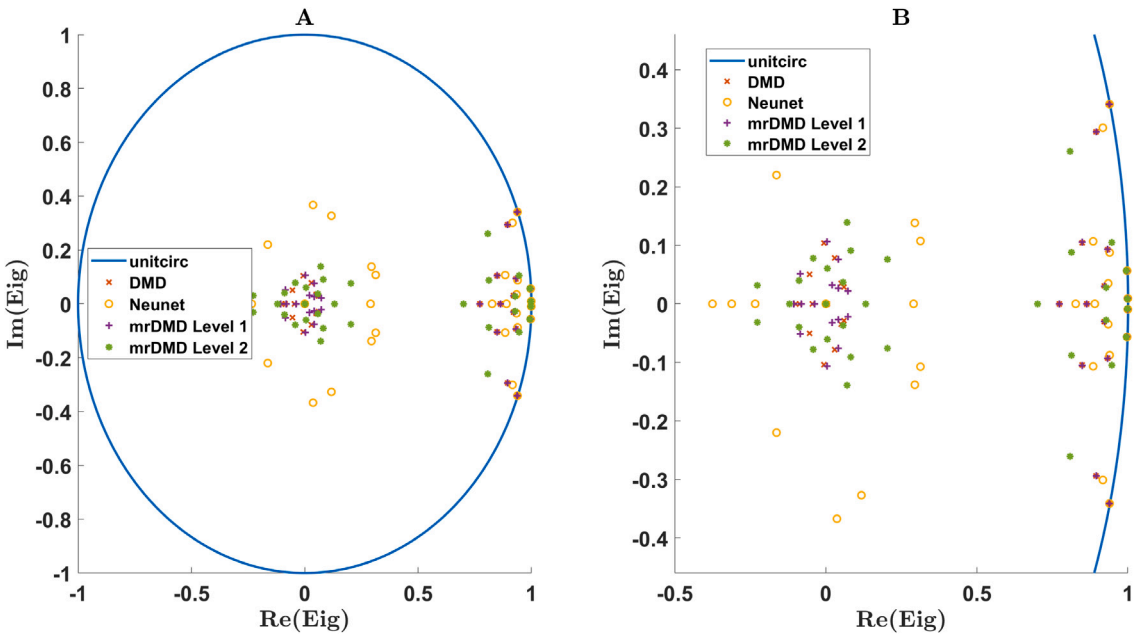

Fig. 6. Panel A shows the eigenvalues from DMD, mrDMD and the proposed neural network approach on the complex plane for the synthetic image dataset. Panel B gives a zoomed-in view of how these eigenvalues are placed when compared to each other. Only Level 1 and Level 2 eigenvalues from mrDMD are shown for comparison.

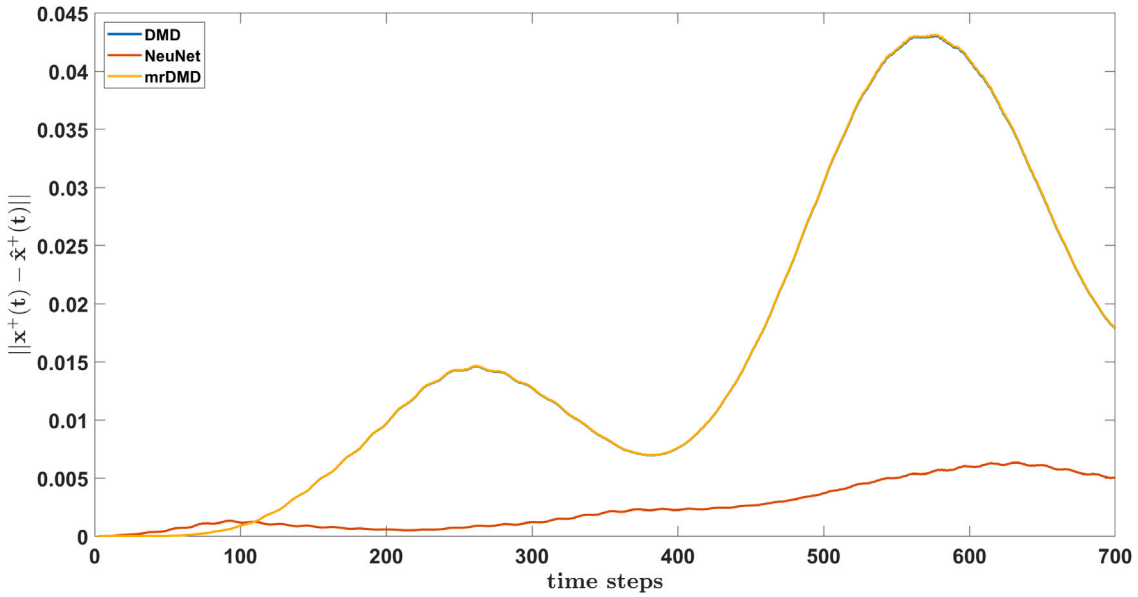

Fig. 7. Red and blue lines in the figure represent the MSE trends for the proposed neural network strategy and DMD. Meanwhile, yellow represents the error computed for multi-resolution DMD. Note here that the blue and yellow lines are nearly indistinguishable. The error is computed by measuring the difference between the predicted next observable and the ground truth given the current observable.

4.3. Schlieren images of supersonic flow past a cylinder

The proposed eigendecomposition strategy is finally applied to experimental data containing schlieren images of cylinder-generated transitional shock-wave/boundary layer interactions in response to supersonic (Mach 2) flow. These are taken at 50 kHz. In each image, the flow goes from left to right. Salient features of the image include a flat plate visible on the bottom edge and an upright cylinder on the right side. A key feature of these images is its characteristic frequency found through analyzing power spectral densities and employing linear data analysis techniques like proper orthogonal decomposition (POD) to be approximately 5 kHz as shown in [37]. Ref. [37] details the overall experimental setup and how data is collected.

The original dataset consists of 12 500 frames; each one of them contains 5472 pixels. These are then converted into grayscale and then resized to mostly contain features around the cylinder as shown in Panel B of Fig. 8. Fig. 8 illustrates an example image from the schlieren dataset in Panel A; differences in pixel intensities correspond to the difference in fluid density gradients. Meanwhile, Panel B in the figure shows a sized down version of the image centered close to the cylinder. The red rectangles show the regions from where the pixels intensities shown in panels B-D from Fig. 10 are taken. The resulting dataset is split into training and testing data equally; the testing data is kept aside to

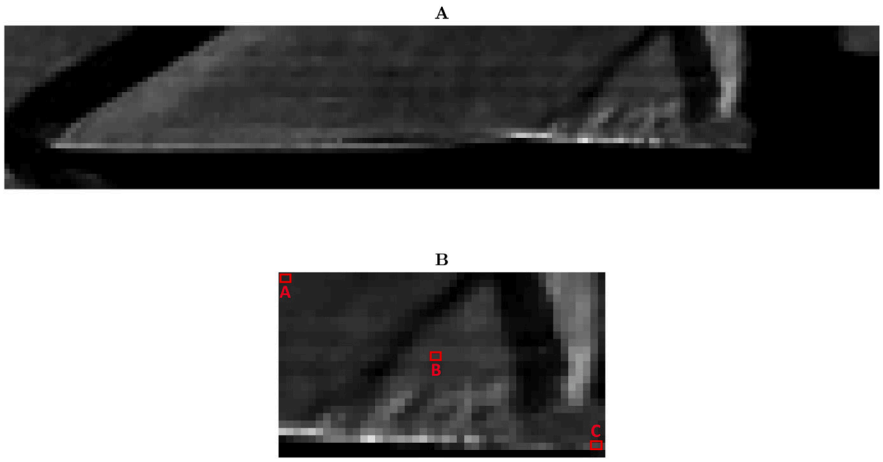

Fig. 8. Panel A in the figure shows an example snapshot from the experimental dataset showing how the Mach 2 flow interacts with the cylinder mounted to the bottom plate. A scaled down image centered close to the cylinder is shown in Panel B; the labeled red boxes show the region from which the three pixel intensities in Panel B-D of Fig. 10 are taken.

be used for checking the accuracy of the system approximation after it is obtained from the proposed strategy. To minimize the effects of noise, POD is applied to the training data reducing the dimension of each image of the dataset and representing them through a 5-mode POD basis. The POD basis captures almost 28 percent of the total energy of the measured data; only 28 percent is captured through the POD basis so that the resulting reduced datasets inherit a minimum amount of noise present in the raw data while retaining important information about the dynamics from the data. The obtained POD modes are then used to reduce the dimensionality of both the training and testing data that have been mean subtracted before POD strategy is applied.

Both the POD reduced training and test data are then stored in two separate matrices each containing the current observables and the corresponding next observables. Time-delay embedding technique is used to generate additional snapshots of data to lift the 5-mode POD basis in both the training and test data matrices to a new basis containing 50 observables as described in Section 2.1. As done for the previous neural population example, the number of eigenvalues used for both DMD and our proposed strategy are specified first; in this case, they are chosen to be 30. The initial observable is specified and the system matrix $\mathbf{A_{dmd}}$ for the given data is computed through DMD based on the chosen number of eigenvalues. By decomposing $\mathbf{A_{dmd}}$, the eigenvalues and the corresponding left and right eigenvectors are found; these contain both real and complex eigenvalues. For the proposed strategy, the complex eigenvalues and eigenvalues from DMD are transformed into Block-II form, given by Eqs. (11) and (12), and then stored with the real eigenvalues with corresponding eigenvectors.

Using the DMD eigenvalues and eigenvectors converted to the form from Eqs. (13)–(15) as the initial weights, a neural network is structured based on the implementation shown in Fig. 1. For training the neural network, the ADAM optimizer is used with a learning rate of 2×10^{-2}. The training loss

$$\mathcal{L}_{total} = \frac{10}{\zeta}\sum_{i=1}^{\zeta}(\mathbf{x_i^+}-\hat{\mathbf{x}}_\mathbf{i}^+)^2 + (\hat{\mathbf{W}}\hat{\mathbf{V}}-\mathbf{I}) + 1\times10^{-2}\sum_{k=1}^{\vartheta}(|\hat{\lambda}_\mathbf{k}|-\mathbf{1}) + \quad 2\times10^{-1}\sum_{k=1}^{5}(\mathbf{Arg}(\hat{\lambda}_\mathbf{k})-0.5519), \tag{23}$$

contains the MSE loss, the orthogonality loss term to ensure that learned left and right eigenvectors are orthogonal and finally, a combination of magnitude and frequency constraints. The magnitude constraint is applied to all the weights in $\hat{\mathbf{\Lambda}}$ to push their magnitude value close to the unit circle and learn eigenvalues that are slow decaying. The frequency constraint, in this example, utilizes the dominant 4.7 kHz frequency identified from the power spectrum in [37] (the corresponding argument in radians is 0.5519). This is included to prioritize this dominant frequency while learning the Koopman eigenfunctions. All these loss terms are added to each other with specific weights that are found through successive trials to learn an optimal set of eigenvalues and eigenvectors that give better long term observable predictions than DMD. The learning process follows the same steps highlighted in Section 3.5.

By utilizing the learned eigenvalues and eigenvectors from the neural network, a system matrix approximation $\mathbf{A_{neu}}$ is generated. To compare how these learned eigenvalues differ from the DMD eigenvalues, both of them are plotted on the complex plane through their real and imaginary components as shown in Fig. 9. Panel A shows the distribution of eigenvalues in the unit circle; all of them lie inside the unit circle in its right half. Panel B shows how the learned eigenvalues contrast with the eigenvalues obtained through DMD in more detail. Neural network eigenvalues, represented by yellow circles, are closer to the unit circle when compared with the red crosses that represent DMD eigenvalues. This is because of the magnitude constraint that pushes the learned eigenvalues towards the unit circle, i.e. their magnitude should be close to 1. Also, many learned eigenvalues have the same argument close to the dashed lines in Fig. 9 due to the frequency constraint applied through regularization that forces their argument to be close to a set value; this set value is computed using the dominant frequency 4.7 kHz obtained from power spectrum analysis of the dataset.

By randomly choosing the initial observable, observable estimates for 100 timesteps are generated by using the system matrix approximations $\mathbf{A_{dmd}}$ and $\mathbf{A_{neu}}$ for DMD and our proposed strategy respectively. For getting the initial observable, an initial step value is calculated using $\text{initstep} = a_3 + (b_3 - a_3)\text{rand}(0,1)$ with $a_3 = 0$ and $b_3 = 6000$ which is then used as an index to get the initial observable from the test data. Future observables can be predicted given the initial observable by using Eq. (20). Generated observable predictions are then compared with the corresponding ground truth through MSE; this process is repeated for 200 trials. Fig. 10 depicts the resulting MSE plot in panel A obtained by averaging over all the trials. Panels B-D compare pixel intensities from three different pixels for over a 100 timestep window for the proposed neural network approach and DMD; these pixels are taken from the regions highlighted in Panel B of Fig. 8. For the initial few timesteps, both DMD and the neural network estimates have approximately the same error. However, the difference becomes significant for subsequent predictions with the proposed approach having lower mean squared error than standard DMD. The initial jump in MSE error in the trends shown in Fig. 10 is due to the usage of POD basis which only captures 28 percent of the total energy of the

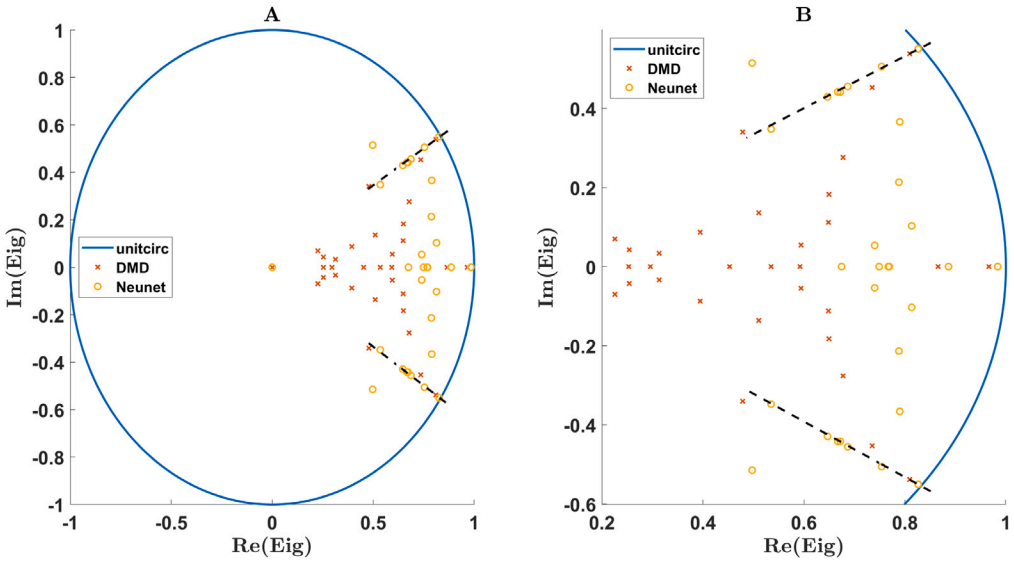

Fig. 9. Real and imaginary components of eigenvalues for both DMD and the proposed neural network approach are shown on the complex plane. The unit circle is shown in blue whereas red and yellow depict the eigenvalues for DMD and the neural network respectively. Panel A depicts how all the eigenvalues are placed on the complex plane while Panel B shows how the DMD and neural network eigenvalues differ in more detail. Black dashed lines in both panels show the target complex value argument.

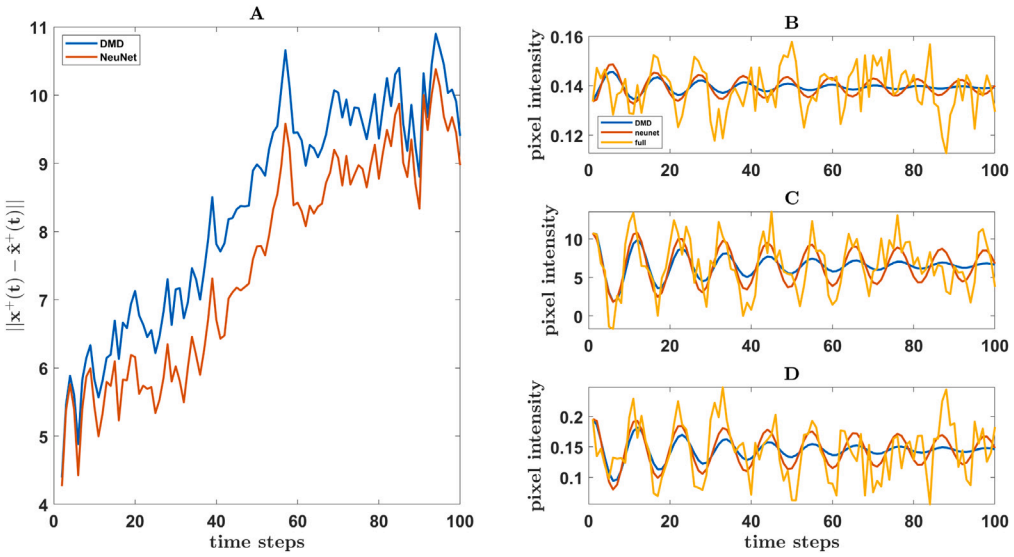

Fig. 10. Panel A shows the error difference between the estimated next observables and the ground truth for DMD (in blue) and the proposed approach (in red) given the initial observable. Pixel intensity values are plotted for three different pixels of a single frame from the original dataset (shown in yellow) and compared to the predicted pixel values from DMD (in blue) and our approach (in red) in Panels B to D. It should be noted that predicted pixel intensities from DMD decay quickly as compared to our proposed approach; this is due to the fast decaying eigenmodes learned by DMD that fail to capture the oscillations in the original dataset as accurately as the proposed approach.

measured data. Only 28 percent is captured through POD to minimize the interference of noise present in the raw data while learning the system approximation for the observable data. It is still impossible to filter out the noise completely due to the need to retain important information from the original data; for the dynamics that can be captured, the proposed neural network approach does reasonably well. These results are further considered by examining the pixel intensity plots illustrated in panels B-D which show that for each of the three pixels, neural network based system approximation does a better job in approximating the oscillatory dynamics of the full model pixels as compared to standard DMD. While the neural network approach is not able to perfectly match the pixel intensity amplitude, it is able to match the pixel's frequency for all time steps. The pixel intensities predicted by DMD converge to the average of the original pixel intensity values as time proceeds as many of its eigenvalues are fast decaying and hence, DMD generates an approximation that is unable to account for the full model dynamics as accurately as our proposed approach. Thus, by applying both the magnitude and frequency constraints through the proposed strategy, eigenvalues and corresponding eigenvectors are learned that have better long term observable predictions and match the ground truth profile better than DMD as seen in Fig. 10 as the slow decay of the eigenmodes account for the dominant characteristic frequency in the power spectrum of the original dataset better than DMD.

5. Discussion and conclusion

We propose an approach for inferring the Koopman eigenvalues and eigenfunctions from data in this work to predict future dynamics given the current observables of the system. A neural network based strategy is devised to learn these Koopman eigenfunctions as weights of the network shown in Eqs. (13)–(15) while explicitly considering and utilizing information about the Koopman eigenvalues during the learning process. This information is incorporated as a combination of domain knowledge based constraints which regularize the learned weights such that the system approximation obtained through these learned Koopman eigenfunctions outperform standard techniques like DMD in terms of long term observable predictions. The results are illustrated through a collection of datasets from nonlinear models.

Domain knowledge constraints are applied according to the system under consideration. For example, magnitude based constraints are primarily applied to systems like the spiking neuron population model in the first example which has sustained limit cycle oscillations. To ensure that learned system approximation is able to capture such periodicity accurately, the magnitude based constraints force the Koopman eigenvalues to be close to the unit circle. This means that the learned Koopman eigenvalues will be slow decaying and hence, will contribute more to the system dynamics alongside capturing the periodicity of the limit cycle as $t \to \infty$. Similarly, frequency based constraints are chosen to incorporate domain knowledge or specific frequency patterns observed from the full model data by regularizing the argument of the learned Koopman eigenvalues to be near a specific value. The value can be computed by analyzing the power spectrum of given data as done with the schlieren images data example. This allows our proposed approach to learn a system approximation that can capture the frequency profile of the underlying model efficiently by prioritizing the relevant frequency modes. Choosing the constraint, the argument and the combinations of neural network weights these constraints are applied to, requires a trial and error process. It might not be beneficial to apply different kinds of constraints for better long term observable predictions for some applications as demonstrated by the neuron population and the synthetic image dataset example.

Neural networks have been used in previous works to learn the dynamics of an underlying model from observable data. Authors in [38] present a model identification strategy utilizing isostable coordinates for systems with a fixed point. The neural network is structured based on an isostable coordinate reduced model that is reframed through a lifted basis into a set of unknown linear functions that are subsequently learned using the weights of the neural network in conjunction with known nonlinear functions. The neural network is then trained on observable data to learn these unknown coefficients and obtain a reduced model to approximate the full model dynamics. Work presented in [39] extends the neural network based model identification strategy for oscillatory dynamics where the weights of the neural network are the Fourier series coefficients of the unknown linear functions of the phase isostable coordinate-based reduced model. By training the network, one can learn an approximation to the underlying oscillatory dynamics by obtaining a phase isostable coordinate-based reduced model.

The proposed strategy in this work follows a similar idea by utilizing a neural network to learn an approximation of the Koopman eigenfunctions for approximating the underlying system dynamics through observable data. Comparing our proposed work with the strategies from [38,39], a supervised learning framework is used to implement our approach where both the input and output data is already available, unlike the other two strategies which generate data as the training proceeds. Our proposed approach also makes use of constraints based on prior domain knowledge to learn better approximation of the underlying system matrix while the neural networks presented in [38, 39] do not rely on domain knowledge. Note that the neural network framework in our proposed approach does not include nonlinear activation functions to preserve the linearity in the eigendecomposition structure from (16). The neural network framework is primarily used as an optimization tool in our approach to learn the approximation of eigenvalues and eigenvectors of the underlying dynamical system through the backpropagation algorithm. Moreover, the neural network framework allows for both scalability in the number of parameters to be learned and to adjust the learned eigendecomposition according to the domain knowledge-based constraints through the loss function which is one of the key innovations of our proposed method.

The effectiveness of the proposed strategy can be improved in a number of ways from a deep learning perspective. For example, in order to consider complex eigenvalues and eigenvectors, the neural network is restructured according to forms shown in Eqs. (13)–(15) separating the real and imaginary components for complex values to allow for a real-valued neural network to be used. It might be worthwhile to employ complex-valued neural networks (CVNN) [31,40] for our strategy. These networks have their own specific architectures, learning schemes and optimization algorithms which might be advantageous in learning the complex eigenvalues and eigenvectors of the underlying dynamical system more efficiently. Also, the current MSE based loss function in Eq. (17) incorporates the orthogonality constraint and other regularization constraints in an additive fashion which makes this approach a multi-objective optimization problem. Even though this implementation gives adequate performance in presented examples, it might result in sub-optimal solutions. Therefore, it would be useful to investigate a custom training functions that can incorporate the constraints within itself more efficiently rather than in an additive fashion.

Also, it might be useful to replace the MSE error in the loss function with a more suitable metric as the MSE is unable to fully account for discrepancies related to the frequency of the full model output. This ultimately results in mismatches between the predicted and the ground truth output profiles. Because one of the main goals of our proposed strategy is to provide more accurate long term observable predictions than DMD, it might be advantageous to modify the loss function in a way that we are able to compute errors for training based on multi-step predictions. This might potentially lead to solutions obtained through our approach performing much better than presented against DMD that only relies on minimizing error for one-step ahead predictions. Another potential approach is to use batch normalization [41,42] for our proposed strategy which makes the network convergence during training more stable and efficient. However, the implementation is non-trivial and a workaround is needed to avoid altering the eigendecomposition based network structure.

CRediT authorship contribution statement

Talha Ahmed: Writing – review & editing, Writing – original draft, Visualization, Validation, Software, Resources, Methodology, Investigation, Formal analysis, Data curation, Conceptualization. **Dan Wilson:** Writing – review & editing, Writing – original draft, Visualization, Validation, Supervision, Software, Resources, Project administration, Methodology, Investigation, Funding acquisition, Formal analysis, Data curation, Conceptualization.

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.

Acknowledgments

This material is based upon the work supported by the National Science Foundation (NSF), USA under Grant No. CMMI-2140527.

Data availability

Data will be made available on request.

References

[1] A. Ghadami, B.I. Epureanu, Data-driven prediction in dynamical systems: recent developments, Phil. Trans. R. Soc. A 380 (2229) (2022) 20210213.

[2] S.J. Schiff, Towards model-based control of Parkinson's disease, Philos. Trans. R. Soc. A: Math., Phys. Eng. Sci. 368 (1918) (2010) 2269–2308.

[3] W.L. Jin, A dynamical system model of the traffic assignment problem, Transp. Res. Part B: Methodol. 41 (1) (2007) 32–48.

[4] U. Ledzewicz, H. Moore, Dynamical systems properties of a mathematical model for the treatment of CML, Appl. Sci. 6 (10) (2016) 291.

[5] A. Chatterjee, An introduction to the proper orthogonal decomposition, Current Sci. (2000) 808–817.

[6] J. Weiss, A tutorial on the proper orthogonal decomposition, in: AIAA Aviation 2019 Forum, 2019, p. 3333.

[7] M. Sieber, C.O. Paschereit, K. Oberleithner, Spectral proper orthogonal decomposition, J. Fluid Mech. 792 (2016) 798–828.

[8] T. Bui-Thanh, M. Damodaran, K. Willcox, Proper orthogonal decomposition extensions for parametric applications in compressible aerodynamics, in: 21st AIAA Applied Aerodynamics Conference, 2003, p. 4213.

[9] C.W. Rowley, T. Colonius, R.M. Murray, Model reduction for compressible flows using POD and Galerkin projection, Phys. D: Nonlinear Phenom. 189 (1–2) (2004) 115–129.

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

[11] I.V. Gosea, S. Gugercin, C. Beattie, Data-driven balancing of linear dynamical systems, SIAM J. Sci. Comput. 44 (1) (2022) A554–A582.

[12] P.J. Schmid, Dynamic mode decomposition and its variants, Annu. Rev. Fluid Mech. 54 (2022) 225–254.

[13] Jonathan H. Tu, Dynamic Mode Decomposition: Theory and Applications (Ph.D. thesis), Princeton University, 2013.

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

[15] J.L. Proctor, S.L. Brunton, J.N. Kutz, Dynamic mode decomposition with control, SIAM J. Appl. Dyn. Syst. 15 (1) (2016) 142–161.

[16] J.N. Kutz, X. Fu, S.L. Brunton, Multiresolution dynamic mode decomposition, SIAM J. Appl. Dyn. Syst. 15 (2) (2016) 713–735.

[17] H. Arbabi, I. Mezic, Ergodic theory, dynamic mode decomposition, and computation of spectral properties of the Koopman operator, SIAM J. Appl. Dyn. Syst. 16 (4) (2017) 2096–2126.

[18] G. Snyder, Z. Song, Koopman operator theory for nonlinear dynamic modeling using dynamic mode decomposition, 2021, arXiv preprint arXiv:2110.08442.

[19] C. Folkestad, D. Pastor, I. Mezic, R. Mohr, M. Fonoberova, J. Burdick, Extended dynamic mode decomposition with learned koopman eigenfunctions for prediction and control, in: 2020 American Control Conference (Acc), IEEE, 2020, pp. 3906–3913.

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

[21] T. Krake, D. Klötzl, B. Eberhardt, Daniel Weiskopf, Constrained dynamic mode decomposition, IEEE Trans. Vis. Comput. Graphics 29 (1) (2022) 182–192.

[22] A. Cichocki, R. Unbehauen, Neural networks for computing eigenvalues and eigenvectors, Biol. Cybernet. 68 (1992) 155–164.

[23] D. Finol, Y. Lu, V. Mahadevan, A. Srivastava, Deep convolutional neural networks for eigenvalue problems in mechanics, Internat. J. Numer. Methods Engrg. 118 (5) (2019) 258–275.

[24] M.O. Williams, I.G. Kevrekidis, C.W. Rowley, A data–driven approximation of the Koopman operator: Extending dynamic mode decomposition, J. Nonlinear Sci. 25 (6) (2015) 1307–1346.

[25] H. Kantz, T. Schreiber, Nonlinear Time Series Analysis, vol. 7, Cambridge University Press, 2004.

[26] I. Mezić, Analysis of fluid flows via spectral properties of the Koopman operator, Annu. Rev. Fluid Mech. 45 (2013) 357–378.

[27] M. Budišić, R. Mohr, I. Mezić, Applied Koopmanism, Chaos: Interdiscip. J. Nonlinear Sci. 22 (4) (2012) 047510.

[28] I. Mezić, Spectrum of the Koopman operator, spectral expansions in functional spaces, and state-space geometry, J. Nonlinear Sci. (2019) 1–55.

[29] K. Taira, S.L. Brunton, S.T.M. Dawson, C.W. Rowley, T. Colonius, B.J. McKeon, O.T. Schmidt, S. Gordeyev, V. Theofilis, L.S. Ukeiley, Modal analysis of fluid flows: An overview, AIAA J. 55 (12) (2017) 4013–4041.

[30] Jose Agustin Barrachina, Chengfang Ren, Gilles Vieillard, Christele Morisseau, Jean-Philippe Ovarlez, Theory and implementation of complex-valued neural networks, 2023, arXiv preprint arXiv:2302.08286.

[31] C. Lee, H. Hasegawa, S. Gao, Complex-valued neural networks: A comprehensive survey, IEEE/CAA J. Autom. Sin. 9 (8) (2022) 1406–1426.

[32] B. Pang, E. Nijkamp, Y.N. Wu, Deep learning with tensorflow: A review, J. Educ. Behav. Stat. 45 (2) (2020) 227–248.

[33] J.E. Rubin, D. Terman, High frequency stimulation of the subthalamic nucleus eliminates pathological thalamic rhythmicity in a computational model, J. Comput. Neurosci. 16 (3) (2004) 211–235.

[34] J.N. Kutz, S.L. Brunton, B.W. Brunton, J.L. Proctor, Dynamic Mode Decomposition: Data-Driven Modeling of Complex Systems, SIAM, 2016.

[35] N.M. Kalimullah, A. Shelke, A. Habib, Multiresolution dynamic mode decomposition (mrDMD) of elastic waves for damage localisation in piezoelectric ceramic, Ieee Access 9 (2021) 120512–120524.

[36] M. Bilal, M. Rizwan, S. Saleem, M.M. Khan, M.S. Alkatheir, M. Alqarni, Automatic seizure detection using multi-resolution dynamic mode decomposition, IEEE Access 7 (2019) 61180–61194.

[37] D. Wilson, S. Sahyoun, P. Kreth, S.M. Djouadi, Investigating the underlying dynamical structure of supersonic flows using effective model reduction, in: 2020 American Control Conference, ACC, IEEE, 2020, pp. 4527–4532.

[38] T. Ahmed, A. Sadovnik, D. Wilson, Data-driven inference of low-order isostable-coordinate-based dynamical models using neural networks, Nonlinear Dynam. (2022) 1–19.

[39] T. Ahmed, D. Wilson, Phase-amplitude coordinate-based neural networks for inferring oscillatory dynamics, J. Nonlinear Sci. 34 (1) (2024) 15.

[40] J. Bassey, L. Qian, X. Li, A survey of complex-valued neural networks, 2021, arXiv preprint arXiv:2101.12249.

[41] S. Santurkar, D. Tsipras, A. Ilyas, A. Madry, How does batch normalization help optimization? Adv. Neural Inf. Process. Syst. 31 (2018).

[42] S. Ioffe, C. Szegedy, Batch normalization: Accelerating deep network training by reducing internal covariate shift, in: International Conference on Machine Learning, PMLR, 2015, pp. 448–456.