Nonlinear Dyn
https://doi.org/10.1007/s11071-022-07954-x

ORIGINAL PAPER

Data-driven inference of low-order isostable-coordinate-based dynamical models using neural networks

Talha Ahmed · Amir Sadovnik · Dan Wilson

Received: 24 February 2022 / Accepted: 27 September 2022

Abstract The isostable coordinate system, which encodes for level sets of the slowest decaying eigenmodes of the Koopman operator, provides an effective framework with which to represent the dynamics of a general nonlinear system using a low-order basis. When the underlying model equations are known, transformation to an isostable-coordinate-based reduced order model is relatively straightforward. In a data-driven setting, where only time series measurements of observables are available, robust, accurate, and general strategies for inference of these reduced order models have yet to be developed. To this end, in this work we reframe the reduced order isostable coordinate dynamics of a general nonlinear dynamical system in the basin of attraction of a stable fixed point in terms of the composition of a set of known nonlinear functions and unknown linear functions. This framing allows for the use of an artificial neural network to identify the weights of the unknown linear functions without any need of prior estimation of the isostable coordinates. Once learning is completed, these weights can be extracted to yield a nonlinear reduced order model that is independent of the artificial neural network. The proposed technique is illustrated in a collection of models including one that considers the dynamics of a synaptically coupled population of tonically firing conductance-based neurons.

Keywords Data-driven · Model identification · Isostable Coordinates · Model order reduction · Artificial neural networks · Nonlinear systems

1 Introduction

Recent years have seen a surge of interest in the development of data-driven strategies for inference of low-dimensional, predictive dynamical models [7,19]. Such techniques attempt to determine an appropriate set of equations that can accurately reflect model output in response to general inputs without prespecifying any underlying dynamical structure. This data-driven model identification paradigm represents a significant departure from more conventional strategies that build models from the bottom up, first deciding on the underlying dynamical mechanisms and subsequently fitting model parameters to experimental data, for example, as was the approach of Hodgkin and Huxley [14]. In contrast to bottom-up model identification approaches, implementation of data-driven model identification techniques generally does not require any domain-specific knowledge. Indeed, data-driven strategies are particularly well-suited for use in applications where the underlying dynamical mechanisms that give rise to observed model behaviors are not well under-

T. Ahmed · A. Sadovnik · D. Wilson (✉)
Department of Electrical Engineering and Computer Science, University of Tennessee, Knoxville, TN 37996, USA
e-mail: dwilso81@utk.edu

stood and in situations where there is not a clear mapping between state variables and model observables.

Many well-established reduced order model identification strategies employ modal decomposition techniques which attempt to extract a subset of dynamically important features from an available dataset. Proper orthogonal decomposition (POD) [5,15,45] is one such algorithm that yields an optimal set (in an L^2 energy sense) of orthogonal reduced order modes. Alternatively, dynamic mode decomposition (DMD) [19,36,41] identifies modes with associated growth, decay, or oscillation rates from temporally ordered pairs of snapshot data. Both POD and DMD are well suited to high-dimensional datasets where the less important modes can be truncated. If the underlying equations are known, they can be projected onto a representative set of modes in order to yield a reduced order model in a process commonly referred to as Galerkin projection [15,27]. If the underlying equations are unknown, it is generally not straightforward to identify an associated dynamical model, especially for systems with fundamentally nonlinear behavior.

A variety of other data-driven model identification strategies have been developed with the ability to explicitly account for non-negligible nonlinearities. Among these are techniques that attempt to learn the associated model equations themselves, automatically selecting terms from a prespecified function library that best reproduce the training data [8,22,31,40]. These typically promote sparsity to avoid overfitting and can be particularly useful when the model behavior can be accurately represented with a relatively simple set of dynamical equations. Alternatively, the theory of spectral submanifolds [10,12], which represent an extension of linear modal subspaces to nonlinear systems, has shown promise in the analysis of nonlinear vibrations [33,43]. Generally, when using the spectral submanifold approach, asymptotic dynamics can be accurately captured by considering a subspace spanned by a collection of the slowest modes allowing for analysis in terms of a reduced order model. Additionally, reduced order methods that incorporate adaptive parameter sets [53–55,57] attempt to capture salient features of the nonlinear system by including a set of nonstatic parameters with an associated parameter update rule designed to limit truncation errors.

Methods based on Koopman analysis have also gained traction for model identification purposes. Koopman operator theory allows for the representation of a generally nonlinear dynamical system in terms of a linear, but possibly infinite dimensional operator [9,25,26]. The critical challenge of implementing Koopman analysis is in finding a finite dimensional representation for this infinite dimensional operator. In a data-driven setting, this can be accomplished by finding representative eigenmodes of the Koopman operator; learning-based approaches have shown promise for this task [11,17,21,61]. Koopman operator theory has a close connection to DMD [36] which can also be used to approximate Koopman eigenmodes. Such algorithms typically work best when high-dimensional data is available but can also be implemented in conjunction with lifting functions [46], for instance using delay embedding of time series data [3,4,6,49]. How to best accommodate external inputs when using Koopman-based approaches is still an open question. Authors of [18] and [34] consider least-squares fitting approaches to obtain linear predictors for the behavior in response to input. Alternatively, [32] considers the identification of a set of Koopman reduced order models that correspond to a finite set of constant controls, ultimately yielding a switching time optimization problem.

In this work, we focus on model identification techniques that leverage isostable coordinates [24, 58], which represent level sets of the slowest decaying eigenmodes of the Koopman operator. Because isostable coordinates can be defined in the basin of attraction of a fixed point, this coordinate framework can be used to accurately capture fully nonlinear behaviors in the corresponding dynamical system. If the underlying dynamical equations are known, it is possible to compute the necessary terms of the isostable-based reduced order models to high orders of accuracy using strategies described in [50] and [52]. In a data-driven setting (i.e., when the model equations are unknown) it is not as clear how to proceed. References [56] and [52] propose a least-squares model fitting approach using the steady state response to periodic forcing. However, this strategy specifically requires sinusoidal inputs to be used for model fitting, limiting its practical utility in many settings. Here we suggest a different approach: by lifting to a higher dimensional state space, we reframe the isostable-coordinate-based input–output dynamics in terms of the composition of a set of known nonlinear functions and unknown linear functions. We design an artificial neural network that can learn the weights associated with the unknown linear functions from training data. Once learning is completed, the weights can be extracted to yield a nonlinear reduced order model that is independent of the artificial neural network.

Various approaches highlighted in different research works also employ novel variations of the artificial neural networks in order to infer dynamics of nonlinear systems. For instance, reference [29] investigates the use of modular neural networks in obtaining chaotic time-delay system dynamics and compares results to those obtained using feed-forward neural networks. Additionally, reference [28] introduces the idea of orthogonal neural networks to predict the solution of numerous time-dependent and independent systems following singular Emden-Fowler dynamics whose behavior can be described by either an ordinary or partial differential equation. To identify the dynamic model of two specific processes in an automotive engine, [44] uses external recurrent neural networks for model identification.

Other previous works focus on utilizing feed forward artificial neural networks in order to tackle model identification of nonlinear dynamics. Authors in [23] use the notion of dynamic neurons in feed-forward neural networks in order to capture system nonlinearities. Also, [30] focuses on using feed-forward neural networks for long-time predictive modelling of nonlinear dynamical systems by augmenting Jacobian regularization in the network's loss function. Both of these approaches differ from our work through the fact that our approach structures the artificial neural network according to our reduced isostable coordinate dynamics rather than just utilizing the network as a black-box. Another approach, presented in [60], is based upon the equivalency of feed-forward neural networks to Volterra series; the authors use this idea to compute Volterra kernel representations for dynamical systems through the internal parameters of the network.

A new class of neural networks, physics informed neural networks (PINNs), have been gaining prominence in inferring dynamics of systems governed by nonlinear partial differential equations (PDEs) [35]. Prior knowledge based on these PDE's is embedded into the neural network training process in order to allow the training algorithm to capture the right solution more efficiently without needing large amounts of training data. Given a set of noisy measurements from the dynamical system, PINNs can be modified to solve both the forward and inverse problems for PDEs. Although our proposed approach also utilizes output measurements in order to infer model dynamics, there are two important distinctions between PINNs and our approach. Unlike PINNs, our approach does not rely on any prior domain knowledge of any physical laws pertaining to the set of equations governing the dynamical system in consideration. Moreover, our approach leverages the isostable based reduced order model dynamics for structuring the neural network whereas PINN's structure is typically based on the type of system at hand.

The organization of this paper is as follows: Sect. 2 provides necessary background on the isostable coordinate framework and previously developed isostable-coordinate-based model order reduction frameworks. Section 3 describes the mathematical formulation that allows for the implementation of the model identification strategy using artificial neural networks and discusses practical matters related to implementation. Results are given in Sect. 4 where we illustrate the proposed technique in two simple dynamical models along with a more complicated illustration in a model that captures neural spiking behavior. Sect. 5 provides a discussion of the proposed strategy in the context of the results and Sect. 6 gives concluding remarks.

2 Background

2.1 Isostable coordinates

Consider a general ordinary differential equation of the form

$$\begin{aligned} \dot{x} &= F(x, U(t)), \\ y &= H(x), \end{aligned} \tag{1}$$

where $x \in \mathbb{R}^N$ is the state, F sets the dynamics, $U(t) \in \mathbb{R}^P$ is a general control input, $y \in \mathbb{R}$ is an observable defined by H. Suppose the system admits a stable fixed point $F(x_{ss}, U_{ss}) = 0$ with $y_{ss} = H(x_{ss})$. Let $A = \frac{\partial F}{\partial x}$ (i.e., the Jacobian) evaluated at the fixed point and let λ_k be an eigenvalue of A with associated left and right eigenvectors w_k and v_k, respectively. Eigenvalues will be ordered so that $\text{Re}(\lambda_k) \geq \text{Re}(\lambda_{k+1})$. Recall that x_{ss} is stable so that λ_1 corresponds to the slowest decaying eigenmode of the linearized system. Provided λ_1 is not defective, an associated isostable coordinate can be defined in the basin of attraction of the fixed point as

$$\psi_1(x) = \lim_{t\to\infty} \left(w_1^T (\phi(t,x) - x_{\rm ss}) \exp(-\lambda_1 t) \right), \tag{2}$$

where $\psi_1 \in \mathbb{C}$. Here, $\phi(t,x)$ denotes the flow of Eq. (1) under the constant application of $U_{\rm ss}$ and ^{T} denotes the vector transpose. In the definition from Eq. (2), the term $w_1^T(\phi(t,x) - x_{\rm ss})$ captures the slow decay to the fixed point and the term $\exp(-\lambda_1 t)$ grows at a corresponding rate in the limit as time approaches infinity (note that $-\lambda_1 > 0$) so that the right hand side of (2) converges to the isostable coordinate. As discussed in [24], isostable coordinates represent level sets of the slowest decaying eigenmodes of the Koopman operator [9,25]. Equation (2) provides an explicit definition for the slowest decaying isostable coordinate. Such an explicit definition is not guaranteed for the faster decaying isostable coordinates [20], which can instead be defined implicitly as level sets of Koopman eigenfunctions with decay rates that are governed by an associated λ_k.

2.2 Model order reduction using isostable coordinates

When taking $U(t) = U_{\rm ss}$, each isostable coordinate evolves in time according to

$$\dot\psi_k = \lambda_k \psi_k, \tag{3}$$

resulting in a simple exponential decay. This property, as described below, allows for the use of an isostable-based reduced order coordinate system that captures input–output relationships for general nonlinear dynamical systems. For simplicity, in this work it will be assumed that $U(t)$ is a rank-1 input, i.e., $U(t) = Bu(t)$ where $B \in \mathbb{R}^P$ and $u(t) \in \mathbb{R}$. With this in mind, let $U_{\rm ss} = Bu_{\rm ss}$. It will also be assumed that $u(t) - u_{\rm ss} = O(\epsilon)$ where $0 < \epsilon \ll 1$. Asymptotic expansion of Eq. (1) about the fixed point, neglecting higher-order terms, yields

$$\dot x = F(x, Bu_{\rm ss}) + \frac{\partial F}{\partial U} B(u(t) - u_{\rm ss}), \tag{4}$$

where the partial derivatives are evaluated at $x(t)$ and $U_{\rm ss}$. Transforming Eq. (4) to work in a basis of isostable coordinates $\psi_1, \dots, \psi_\beta$, from the chain rule, one finds

$$\begin{aligned}\dot\psi_k &= \frac{\partial \psi_k}{\partial x} \cdot \frac{dx}{dt} \\ &= \frac{\partial \psi_k}{\partial x} \cdot F(x, Bu_{\rm ss}) + \frac{\partial \psi_k}{\partial x} \cdot \left(\frac{\partial F}{\partial U} B(u(t) - u_{\rm ss}) \right) \\ &= \lambda_k \psi_k + \frac{\partial \psi_k}{\partial x} \cdot \left(\frac{\partial F}{\partial U} B \right)(u(t) - u_{\rm ss}),\end{aligned} \tag{5}$$

for $k = 1, \dots, \beta$. Above, the dot denotes the dot product and all partial derivatives are once again evaluated at the fixed point. Additionally, in the third line we use the fact that $\dot\psi_k = \lambda \psi_k$ when $U = U_{\rm ss}$.

Equation (5) is still a function of the state precluding its direct use as a reduced order representation of Eq. (1). Instead, as suggested in [50] and [52], one can assume that the truncated fast decaying isostable coordinates are well approximated by zero. Adopting this strategy and following the formulation from [52] yields a reduced order model of the form

$$\begin{aligned}\dot\psi_k &= \lambda_k \psi_k + I_k(\psi_1, \dots, \psi_\beta)(u(t) - u_{\rm ss}), \\ & \quad k = 1, \dots, \beta, \\ y(t) - y_{\rm ss} &= G(\psi_1, \dots, \psi_\beta),\end{aligned} \tag{6}$$

where $I_k(\psi_1, \dots, \psi_\beta) \in \mathbb{C}$ provides a good approximation for $\frac{\partial \psi_k}{\partial x} \cdot (\frac{\partial F}{\partial U} B)$ and $G(\psi_1, \dots, \psi_\beta) \in \mathbb{R}$ provides a good approximation for $H(x) - y_{\rm ss}$. Taylor expanding each I_k and G in a basis of the nontruncated isostable coordinates centered at the fixed point (i.e., for which $\psi_1 = \psi_2 = \dots = \psi_\beta = 0$) yields

$$\begin{aligned}I_n(\psi_1, \dots, \psi_\beta) \approx\ & I_n^0 + \sum_{k=1}^{\beta} \left[\psi_k I_n^k \right] \\ & + \sum_{j=1}^{\beta} \sum_{k=1}^{j} \left[\psi_j \psi_k I_n^{jk} \right] \\ & + \sum_{i=1}^{\beta} \sum_{j=1}^{i} \sum_{k=1}^{j} \left[\psi_i \psi_j \psi_k I_n^{ijk} \right] + \dots,\end{aligned} \tag{7}$$

$$\begin{aligned}G(\psi_1, \dots, \psi_\beta) \approx\ & \sum_{k=1}^{\beta} \left[\psi_k g^k \right] + \sum_{j=1}^{\beta} \sum_{k=1}^{j} \left[\psi_j \psi_k g^{jk} \right] \\ & + \sum_{i=1}^{\beta} \sum_{j=1}^{i} \sum_{k=1}^{j} \left[\psi_i \psi_j \psi_k g^{ijk} \right] + \dots,\end{aligned} \tag{8}$$

for $n = 1, \dots, \beta$. By computing the terms $g^{ijk\dots}$ and $I_n^{ijk\dots}$ to a desired order of accuracy in the isostable coordinate expansion, a reduced order model can be obtained that accurately replicates nonlinear behaviors of the underlying system (1). This general isostable-based approach for model order reduction has been used previously to characterize dynamics associated with memory in entrained oscillations [1], to investigate the emergence of coupling-induced oscillations in nominally non-oscillatory systems [51], and to develop nonfeedback control strategies to stabilize chaotic dynamics [48]. Note that in the reduced order model (6) given above, it is often assumed that both $u(t)-u_{\text{ss}}$ and each isostable coordinate $\psi_1, \dots, \psi_\beta$ are order $O(\epsilon)$ terms. As such, it is conventional to refer to I_n^0, I_n^k, I_n^{jk} from the expansion (7) as first-, second-, third-order terms, respectively, of the reduced order model. Likewise g^k, g^{jk}, g^{ijk} from the expansion (8) will be referred to as the first-, second-, and third-order terms, respectively, with this pattern continuing for higher orders.

3 Problem formulation and general approach

3.1 Problem description

If the underlying model equations from (1) are known, it is possible to directly solve for each $g^{ijk\dots}$ and $I_n^{ijk\dots}$ as described in Appendix A of [52]. If instead the model equations are unknown, the necessary terms must be identified using data-driven methods. Reference [56] suggests a strategy for accomplishing this task by measuring the steady-state output in response to sinusoidal inputs applied over a range of frequencies and using this information to infer linear approximations for each I_k and G from Eq. (6). This strategy was expanded in [52] to infer nonlinear terms in the expansions from (8) and (7). This previous approach requires a substantial amount of data, must use purely sinusoidal inputs for training, and requires measurements of the steady-state response; each of these requirements represents a potential limitation that may preclude implementation in an experimental setting.

In this work we employ an alternative approach. Specifically, for a general system of the form of Eq. (1), given a collection of output measurements $y_1(t), y_2(t), \dots, y_n(t)$ that result from the application of an arbitrary collection of applied inputs $u_1(t), u_2(t), \dots, u_n(t)$, we seek to accurately infer the unknown constants from Eqs. (7) and (8) to arbitrary orders of accuracy in the expansion of isostable coordinates. Our proposed strategy accomplishes this task using gradient descent methods on an artificial neural network with implementation details described below.

3.2 Reframing the nonlinear dynamics

To reframe the problem in a manner that is amenable to a solution using artificial neural networks, we consider an isostable-coordinate-based reduction of the form (6) with initial conditions $\psi_1(t_0), \dots, \psi_\beta(t_0)$ corresponding to output $y(t_0) = G(\psi_1(t_0), \dots, \psi_\beta(t_0))$. For simplicity of exposition, we will take $u_{\text{ss}} = 0$. Letting $\Psi = [\psi_1, \dots, \psi_\beta]^T$ and considering the dynamics mandated by Eq. (5), using a forward Euler method of solution with a timestep of Δt, the isostable coordinate and corresponding outputs at $t = t_0 + \Delta t$ are

$$\begin{aligned} \psi_k(t_0+\Delta t) &= f_k(\Psi(t_0), u(t_0)) \\ &= (1+\lambda_k \Delta t)\psi_k(t_0) \\ &\quad + I_k(\Psi(t_0))u(t)\Delta t, \\ k &= 1, \dots, \beta, \\ y(t_0+\Delta t) - y_{\text{ss}} &= G(\Psi(t_0+\Delta t)). \end{aligned} \tag{9}$$

Above, the functions $f_1, \dots, f_\beta$ take the current isostable coordinates and input and map to the isostable coordinates Δt time units later. To proceed, notice that each f_k with associated I_k, defined according to the Taylor expansion in (7), is linear in a basis of lifted coordinates comprised of the state and input. For instance, when using only one isostable coordinate (i.e., when $\beta = 1$), one can write

$$\begin{aligned} \psi_1(t_0+\Delta t) = (1+\lambda_1\Delta t)[\psi_1(t_0)] \\ + [u(t_0)]I_1^0\Delta t + [\psi_1(t_0)u(t_0)]I_1^1\Delta t \\ + [\psi_1^2(t_0)u(t_0)]I_1^{11}\Delta t \\ + [\psi_1^3(t_0)u(t_0)]I_1^{111}\Delta t + \dots, \end{aligned} \tag{10}$$

where the brackets are used to denote elements of a lifted basis. With this in mind, we will consider each f_k as the composition of two functions

$$f_k = w_k \circ n_I, \tag{11}$$

for $k = 1, \dots, \beta$, where $n_I : \mathbb{C}^\beta \times \mathbb{R} \to \mathbb{C}^\gamma$, $w_k : \mathbb{C}^\gamma \to \mathbb{C}$, and γ is the size of the lifted basis. For example, when $\beta = 1$ and taking the asymptotic expansion to third order of accuracy,

$$n_I(\psi_1, u) = \begin{bmatrix} \psi_1 \\ u \\ \psi_1 u \\ \psi_1^2 u \end{bmatrix}, \quad w_1 = n_I(\psi_1, u)^T \begin{bmatrix} 1+\lambda_1\Delta t \\ I_1^0 \Delta t \\ I_1^1 \Delta t \\ I_1^{11}\Delta t \end{bmatrix}. \tag{12}$$

Note here that u and ψ_1 are both $O(\epsilon)$ so that the terms $\psi_1^3 u$ and I_1^{111} and above are truncated at third order of accuracy. Likewise, we consider G as the composition of two additional functions

$$G = w_G \circ n_G, \tag{13}$$

where $n_G : \mathbb{C}^\beta \to \mathbb{C}^\zeta$ and $w_G : \mathbb{C}^\zeta \to \mathbb{R}$ and ζ is the size of the lifted basis. For example, when $\beta = 1$ and taking the asymptotic expansion to third order of accuracy,

$$n_G(\psi_1) = \begin{bmatrix} \psi_1 \\ \psi_1^2 \\ \psi_1^3 \end{bmatrix}, \quad w_g = n_G(\psi_1)^T \begin{bmatrix} g^1 \\ g^{11} \\ g^{111} \end{bmatrix}. \tag{14}$$

In the context of the data-driven model identification strategies considered in this work, for a general system of the form (1), the functions n_I and n_G are nonlinear but are known once the number of isostable coordinates and order of accuracy are specified. The functions $w_1, \ldots, w_\beta$ and w_G are linear and contain unknown coefficients (i.e., the terms $g^{ijk\cdots}$ and $I_n^{ijk\cdots}$ from Eqs. (7) and (8)). Identification of these unknown coefficients specifies the isostable coordinate update rule from Eq. (9).

3.3 A data-driven approach for model identification using artificial neural networks

Using the formulation described in the previous section, for a general dynamical system, our goal is to learn the unknown terms of the asymptotic expansions from Eqs. (7) and (8) that comprise the output update rule from Eq. (9). These functions themselves are split into compositions given by Eqs. (11) and (13) where terms of the form n_X are known nonlinear functions which lift the state to a higher dimension and terms of the form w_X are linear functions of the lifted state with undetermined coefficients. Using discrete sets of recorded observable measurements $y_j = [y_j(t_0), y_j(t_0+\Delta t), \ldots, y_j(t_0+\eta\Delta t)]$ that result when applying inputs $u_j = [u_j(t_0), u_j(t_0+\Delta t), \ldots, u_j(t_0+(\eta-1)\Delta t)]$ to the system for $j = 1\ldots, \nu$, the goal is to infer the unknown coefficients from Eqs. (7) and (8), along with the decay rates $\lambda_1, \ldots, \lambda_\beta$, ultimately yielding a reduced order model of the form in Eq. (6).

The reduced order isostable coordinate based representation is implemented using a multi-layer simple feed forward network architecture, devised for model learning purposes. The network structure is based on the mathematical formulation from Sect. 3.2 with a representation shown in Fig. 1. The input layer accepts a concatenation of the isostable coordinates $\Psi(t_0)$ and the applied input $u(t_0)$. Next, a nontrainable function layer lifts the input to a higher dimension implementing the (known) function n_I. This lifted state is fed through a hidden layer that implements the linear functions $w_1, \ldots, w_\beta$ to yield an updated isostable coordinate $\Psi(t_0+\Delta t)$. This output is then fed through another nontrainable function layer which implements the lifting described by n_G, and the lifted state is fed through a fully connected trainable output layer that implements w_G.

We denote the estimate of the observable at a given time as $\hat{y}_j(t)$ and compare with a system measurement $y_j(t)$ that results from the application of the input $u_j(t)$. For a given set of input–output measurements y_j and u_j, we use the current approximation for the update function from Eq. (9) to yield a set of approximations $\hat{y}_D = [\hat{y}_j(t_0), \hat{y}_j(t_0+\Delta t), \ldots, \hat{y}_j(t_0+\eta\Delta t)]$. A loss function is defined as the mean squared error (MSE)

$$\text{MSE} = \frac{1}{\nu\eta}\sum_{j=1}^{\nu}\sum_{i=1}^{\eta}(y_j(t_0+i\Delta t) - \hat{y}_j(t_0+i\Delta t))^2. \tag{15}$$

Recall that ν is the number of inputs and η is the number of timesteps for each input. The weights of the trainable layers that implement $w_1, \ldots, w_\beta$ and w_G are updated during backpropagation. For all sets of inputs and corresponding outputs, we take the initial condition to correspond to x_{ss} so that $\psi_1(t_0) = \psi_2(t_0) = \cdots = \psi_\beta(t_0) = 0$ and $y(t_0) = y_{\text{ss}}$. This is done so that the initial condition is known and can be accomplished by allowing the system to approach the stable steady-state solution before applying the test input. Training the neural network using mini-batch gradient descent allows us to train on multiple inputs simultaneously, thus allowing the network to learn a solution that accurately captures the response for general inputs.

The unknown functions in the network described above are linear. As such, we use a linear activation function in the artificial neural network. After training is finished, a nonlinear reduced order model of the form (6) can be recovered from the learned weights of the network. Importantly, the resulting model is independent of the artificial neural network allowing for subsequent application of standard nonlinear control and analysis techniques. As a final point of emphasis, in Eq. (9) I_n is multiplied by Δt for $n = 1, \ldots, \beta$; as such, the learned coefficients $I_n^0, I_n^k, \ldots$ will be proportional to

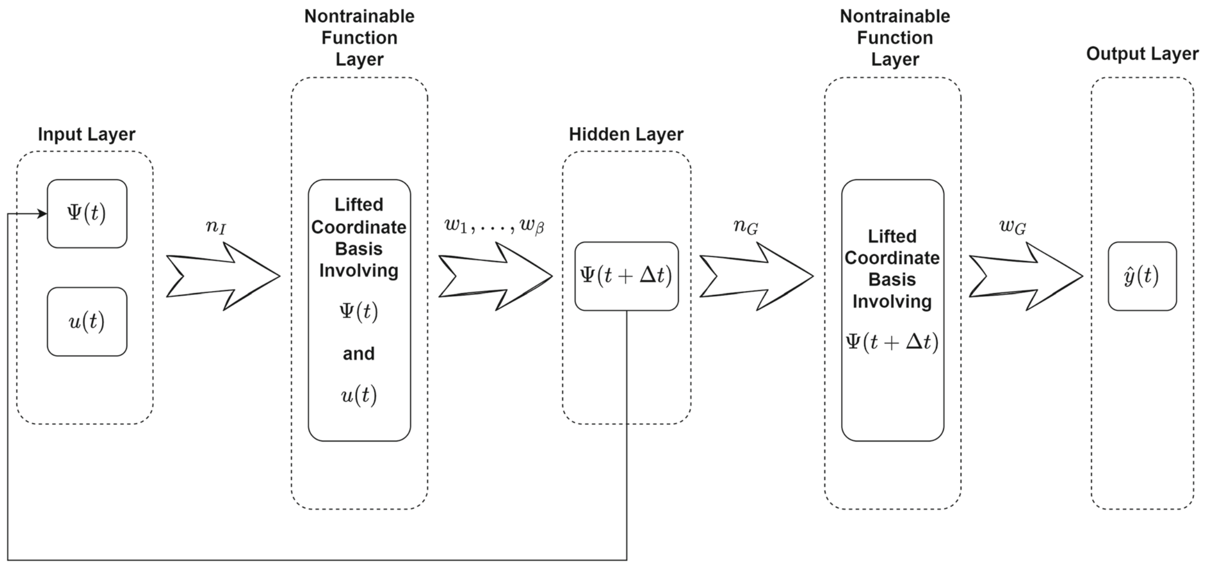

Fig. 1 General architecture for the proposed neural network based strategy

Δt. Conversion between the discrete time update rule (9) and the continuous reduced order model (6) will require an appropriate scaling of the learned weights.

The layer level diagram given in Fig. 1 represents the general architecture comprising the neural network employed in this work. The input layer consists of the value of the isostable coordinates at the current timestep, i.e., $\Psi(t)$, as well as the set of inputs, $u(t)$. Both parts of the input layer are then passed through a nontrainable function layer, n_I, that transforms these components to the lifted coordinate basis. The isostable coordinate update rule from Eq. (9) is then emulated by multiplying the lifted coordinate basis with the coefficients comprising $w_1, \ldots, w_\beta$; these coefficients are the weights associated with the hidden layer in the neural network and are learned through training. The resulting update to the isostable coordinates, $\Psi(t+\Delta t)$, is implemented by a hidden layer in the neural network architecture. The isostable updates are recursively fed back into the input layer of the network to generate further updates at successive timesteps. Once all the essential isostable coordinate values have been generated, these are passed through n_G, to transform $\Psi(t+\Delta t)$ to a second set of lifted coordinates and implemented with a second nontrainable function layer. The mapping from the isostable coordinates to the state from Eq. (9) corresponds to the output layer. Here w_G contains the associated weights. The predicted outputs $[\hat{y}(t_0), \hat{y}(t_0 + \Delta t), \hat{y}(t_0 + 2\Delta t) \ldots]$ are identified recursively and compared with the full model output $[y(t_0), y(t_0 + \Delta t), y(t_0 + 2\Delta t) \ldots]$ to compute the prediction error. This error is used for training through backpropagation. Specific steps required for the training process and the network implementation are described below.

3.4 Implementation of the model-identification strategy using a neural network

The implementation of the artificial neural network based approach for model identification is described in Sect. 3.3; the network structure is feedforward with no recurrent connections. This neural network is implemented and built using Keras, a deep learning application programming interface (API) written in python and built on top of Tensorflow, a machine learning platform within python. Additionally, unlike conventional supervised learning methods which have the training data readily available, the training data has to be generated online as the learning proceeds. This model identification strategy can be performed using the procedure outlined below. This procedure assumes that a representative collection of outputs $y_1(t), \ldots, y_n(t)$ generated by inputs $u_1(t), \ldots, u_n(t)$ has been measured. Each of the measured outputs are assumed to start at the steady-state solution y_{ss} so that the initial isostable coordinates are equal to zero.

Step (1) Specify β, the number of isostable coordinates to use. It is generally better to use the fewest isostable coordinates possible to avoid overfitting. Define functions for the non-trainable function layers that implement the lifting functions n_I and n_G from (11) and (13).

Step (2) Define the structure of the artificial neural network based on the order of accuracy (in the expansion in isostable coordinates) of the functions $f_1, \ldots, f_\beta$ and G from Eq. (9). Linear activation functions are used so that the learned weights correspond directly to the coefficients in the Taylor expansions from Eqs. (7) and (8)

Step (3) Define an auxiliary function that implements the forward Euler step from Eq. (9) using the current network weights. This function is used in order to generate the output approximations $\hat{y}_D(t)$ used for training.

Step (4) Initialize network weights, specify an optimizer, a loss function, and a learning rate. In this work, we used the mean square error loss from (15).

Step (5) Each epoch is comprised of two loops. For a given input $u(t)$, the first loop implements the forward Euler update rule from Eq. (9) to calculate the isostable coordinate update. The second loop updates the weights of the trainable layers by training using the data generated by the first loop. As discussed earlier, it is assumed that each $\psi_k(t_0) = 0$ for $k = 1, \ldots, \beta$, i.e., the system starts at its steady-state solution; all subsequent isostable coordinate updates are generated while training without the need of direct measurements. An outer loop is implemented to iterate over the epochs until the training loss converges.

Step (6) The weights in the trained artificial neural network correspond to coefficients of the Taylor expansions from Eqs. (7) and (8) along with decay rates. These weights can be extracted to yield a reduced order model of the form (6) that can be analyzed independent of the artificial neural network.

Below, we provide a few general notes about the implementation of the training procedure specified above. First, the proposed model identification strategy tends to perform poorly if the initialization of the weights from Step 4 yields a model that does not provide a good representation of the training data. To circumvent this issue, it can be useful to initially train using small magnitude inputs so that the influence of the higher-order terms is suppressed and obtain a model that is valid to linear order by finding coefficients of the form I_n^0 and g^n from (7) and (8), as well as the decay rates λ_n for $n = 1, \ldots, \beta$. Subsequently, one can train higher-order models using additional training data obtained from larger magnitude inputs and also using the coefficients of the first-order model in the initialization from Step 4 in the procedure above. Since these weights have already been fit to the lower-order model, regularization can be used to ensure that they do not drift too far from the initialized values during learning for the higher-order weights.

To first-order accuracy, $\partial \psi_n / \partial x$ is given by the left eigenvector of the Jacobian, A, as defined below Eq. (1). As such, in the definition of isostable coordinates from Eq. (2), the scaling for the left eigenvector w_1 can be chosen arbitrarily. A practical consequence of this fact is that each term $I_1^0, \ldots, I_\beta^0$ can each be scaled by an arbitrary constant. Some scaling must be chosen to implement the proposed model identification algorithm; in this work, we scale so that each of these terms is penalized for being far from 1.

In order to train the network, the total number of timesteps for each input–output pair are treated as a single batch in the mini-batch stochastic gradient descent algorithm. The complete dataset is thus comprised of multiple of these batches corresponding to the different inputs. Training on all inputs in each epoch is important for the network to find a solution which can generalize to novel inputs. Since the inputs for subsequent timesteps depend on the network itself, periodically updating the dataset is necessary. Here it is done after each epoch using the current learned weights. There is no need in reaching convergence before updating since the inputs themselves are based on incorrect weights. However, waiting for an entire epoch allows the training to occur on all input–output pairs, thus providing a more generalizable solution.

3.5 Handling complex-valued isostable coordinates

Artificial neural networks perform best when using real valued weights while training. Nonetheless, it is generally possible for isostable coordinates to come in complex conjugate pairs [47], necessitating the identification of complex-valued weights in the expansions from (7) and (8). Some extensions of the traditional feed forward neural network structure have been proposed to handle complex-valued weights [13,39,42]. Here, we take an alternative approach by transforming a network that requires complex-valued weights to an equivalent network that only requires real-valued weights.

To begin, as discussed in [47], isostable coordinates that take complex-values always come in complex-conjugate pairs. As such, consider a reduced order model with two complex-conjugate isostable coordinates, i.e.,

$$\psi_2 = \psi_1^*, \tag{16}$$

where * denotes the complex-conjugate. Considering the isostable coordinate update rule from (10) taken to second-order accuracy, one can write

$$\begin{aligned}\psi_1(t_0+\Delta t) = (1+\lambda_1\Delta t)[\psi_1(t_0)] + [u(t_0)]I_1^0\Delta t \\ +[\psi_1(t_0)u(t_0)]I_1^1\Delta t \\ +[\psi_1^*(t_0)u(t_0)]I_1^2\Delta t.\end{aligned} \tag{17}$$

Separating the real and imaginary components from (17) the following relations are obtained

$$\begin{aligned}\mathrm{Re}(\psi_1^+) &= \mathrm{Re}(\psi_1)(1+\mathrm{Re}(\lambda_1)\Delta t)) \\ &\quad - \mathrm{Im}(\lambda_1)\mathrm{Im}(\psi_1)\Delta t + \mathrm{Re}(I_1^0)u\Delta t \\ &\quad + [\mathrm{Re}(I_1^1)\mathrm{Re}(\psi_1) - \mathrm{Im}(I_1^1)\mathrm{Im}(\psi_1)]\Delta t u \\ &\quad + [\mathrm{Re}(I_1^2)\mathrm{Re}(\psi_1) + \mathrm{Im}(I_1^2)\mathrm{Im}(\psi_1)]\Delta t u, \\ \mathrm{Im}(\psi_1^+) &= \mathrm{Im}(\psi_1)(1+\mathrm{Re}(\lambda_1)\Delta t)) \\ &\quad + \mathrm{Im}(\lambda_1)\mathrm{Re}(\psi_1)\Delta t + \mathrm{Im}(I_1^0)u\Delta t \\ &\quad + [\mathrm{Im}(I_1^1)\mathrm{Re}(\psi_1) + \mathrm{Re}(I_1^1)\mathrm{Im}(\psi_1)]\Delta t u \\ &\quad + [\mathrm{Im}(I_1^2)\mathrm{Re}(\psi_1) - \mathrm{Re}(I_1^2)\mathrm{Im}(\psi_1)]\Delta t u,\end{aligned} \tag{18}$$

where the $^+$ is used to denote the value at the next timestep, for instance, $\mathrm{Re}(\psi_1^+) = \mathrm{Re}(\psi_1(t_0+\Delta t))$ and the explicit time dependence on each of the isostable coordinates has been suppressed for convenience of notation. Notice that in the above equations, all terms are real-valued. Additionally, the state of the second isostable coordinate does not need to be explicitly considered since it is simply the conjugate of the first isostable coordinate.

A similar transformation can be used for the map from the isostable coordinates to the output. Once again, considering a two isostable coordinate model where ψ_1 and ψ_2 are complex-conjugate, the Taylor expansion of the output equation from (8) can be expressed to second-order accuracy as

$$\begin{aligned}y = y_{ss} + \psi_1 g^1 + \psi_1^* g^2 + \psi_1^2 g^{11} \\ +\psi_1\psi_1^* g^{12} + (\psi_1^*)^2 g^{22}.\end{aligned} \tag{19}$$

Noting that the output is real-valued so the expansion at all orders of accuracy must also be real-valued, $g^1 = g^{2^*}$ and $g^{11} = g^{22^*}$. Writing (19) in terms of the real-valued components yields

$$\begin{aligned}y(t) &= 2\mathrm{Re}(\psi_1)\mathrm{Re}(g^1) - 2\mathrm{Im}(\psi_1)\mathrm{Im}(g^1) \\ &\quad + 2\mathrm{Re}(\psi_1)^2\mathrm{Re}(g^{11}) - 2\mathrm{Im}(\psi_1)^2\mathrm{Re}(g^{11}) \\ &\quad - 4\mathrm{Re}(\psi_1)\mathrm{Im}(\psi_1)\mathrm{Im}(g^{11}) \\ &\quad + \mathrm{Re}(\psi_1)^2\mathrm{Re}(g^{12}) + \mathrm{Im}(\psi_1)^2\mathrm{Re}(g^{12}).\end{aligned} \tag{20}$$

Additionally, considering the imaginary components of (19), one finds $\mathrm{Im}(g^{12}) = 0$. Equations (18) and (20) can be decomposed in a manner similar to the decomposition suggested by Eqs. (11) and (13). For instance, considering Eq. (18), the lifted state would contain the elements $\mathrm{Re}(\psi_1)$, $\mathrm{Im}(\psi_1)$, u, $\mathrm{Re}(\psi_1)u$, $\mathrm{Im}(\psi_1)u$ with the remaining coefficients incorporated in the trainable weights of the artificial neural network.

From Eq. (18), one can see that $\mathrm{Re}(\psi_1^+)$ and $\mathrm{Im}(\psi_1^+)$ have shared coefficients. For instance, $(1+\mathrm{Re}(\lambda_1)\Delta t))$ is common to both of the update rules. Hence, for complex-conjugate isostable coordinates, two separate sets of weights are considered: shared and distinct. In order to implement these two sets of weights, we use an additional dense layer in the neural network; two instances of this dense layer are created in the network for the corresponding input terms from both the real and imaginary isostable coordinate updates. The update rules from Eq. (18) also contain distinct coefficients. For instance, $\mathrm{Re}(I_1^0)$ and $\mathrm{Im}(I_1^0)$ are present only in real and imaginary isostable coordinate updates, respectively. Distinct coefficients are implemented with their own separate layer in the neural network. The output from both the shared and distinct layers are summed to yield the update for both $\mathrm{Re}(\psi_1^+)$ and $\mathrm{Im}(\psi_1^+)$. The resulting network structure is shown in Fig. 2 where w_{sh} represents the shared weights, and $w_{dt,Re}$ (resp., $w_{dt,Im}$) is used to denote the distinct weights that comprise the update rule for the real (resp., imaginary) components of the isostable coordinates.

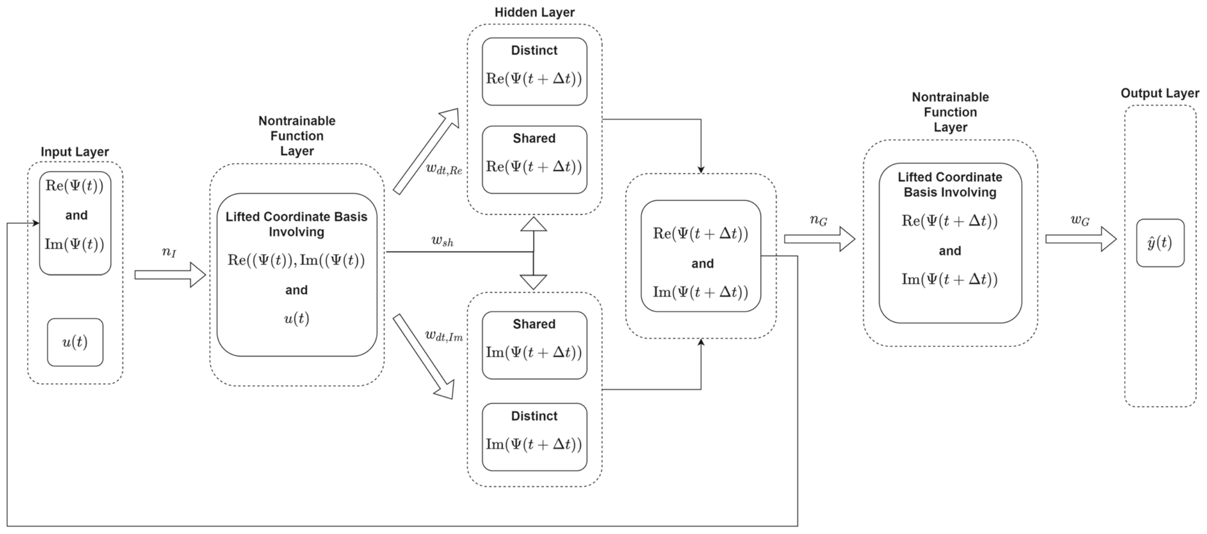
Hidden Layer
Distinct
$\mathrm{Re}(\Psi(t+\Delta t))$
Shared
$\mathrm{Re}(\Psi(t+\Delta t))$
Shared
$\mathrm{Im}(\Psi(t+\Delta t))$
Distinct
$\mathrm{Im}(\Psi(t+\Delta t))$
Input Layer
$\mathrm{Re}(\Psi(t))$
and
$\mathrm{Im}(\Psi(t))$
$u(t)$
n_I
Nontrainable Function Layer
Lifted Coordinate Basis Involving
$\mathrm{Re}((\Psi(t)), \mathrm{Im}((\Psi(t))$
and
$u(t)$
$w_{dt,Re}$
w_{sh}
$w_{dt,Im}$
$\mathrm{Re}(\Psi(t+\Delta t))$
and
$\mathrm{Im}(\Psi(t+\Delta t))$
n_G
Nontrainable Function Layer
Lifted Coordinate Basis Involving
$\mathrm{Re}(\Psi(t+\Delta t))$
and
$\mathrm{Im}(\Psi(t+\Delta t))$
w_G
Output Layer
$\hat{y}(t)$

Fig. 2 A modified architecture is necessary when considering complex-valued isostable coordinates. Separate layers are required to implement the shared and distinct weights that comprise the isostable coordinate update rule. Compared with the structure illustrated in Fig. 1, the primary difference is that shared weights, w_{sh} are used to update both the imaginary and real components of the isostable coordinates at the next timestep. Distinct weights $w_{dt,Re}$ and $w_{dt,Im}$ are used to update the real and imaginary components of the isostable coordinates, respectively

The above equations provide a transformation that can be used to consider a system with two complex-valued isostable coordinates in terms of their real and imaginary components; this transformation eliminates the need for complex-valued weights in the neural network. Similar computations can be performed for higher-order accuracy models. Additionally, this strategy can be straightforwardly implemented with different combinations of complex-valued and real-valued isostable coordinates. Because of the increasing complexity at higher orders of accuracy, it is preferable to compute the required relations using a symbolic computational package.

4 Results

4.1 Illustration in a simple two-dimensional model

As a preliminary illustration of the proposed model identification strategy, we consider the simple 2-dimensional system

$$\begin{aligned} \dot{x}_1 &= \mu x_1 + u(t), \\ \dot{x}_2 &= \sigma(-x_1 + x_2 + x_1^2 + x_1^3), \\ y &= x_2(t), \end{aligned} \tag{21}$$

where the state is $x = [x_1, x_2]$, the output is given by y, $u(t)$ is the input, and constants are taken to be $\sigma = -1$, $\mu = -0.05$. This system was also considered in [52] using a different model identification strategy. When taking $u(t) = 0$, the system settles to a stable equilibrium at $x_1 = x_2 = 0$ and we use the model identification strategy detailed in Sect. 3 to infer a reduced order model with a single isostable coordinate from output measurements in response to a variety of inputs. Starting with the first order of accuracy, the neural network model has three weights corresponding to the λ_1, I_1^0 and g^1. Additional terms from the asymptotic expansions from (7) and (8) are estimated as the order of accuracy considered increases.

For training the model, adaptive moment estimation (ADAM) is used as an optimizer using the MSE loss from (15) with a learning rate of 0.01. The general architecture of the model is shown in Fig. 1. The training is implemented as described in Sect. 3.4. The timestep for the update rule from (9) is taken to be $\Delta t = 0.05$. Ten inputs are used for training, of the form $u(t) = \epsilon \sin(2\epsilon t)$, taking $\epsilon = \{0.001, 0.002, 0.003, ..., 0.010\}$ for $t \in [0, 50]$. Note that on the timescales used for training, these training inputs are similar to ramp functions. The corresponding outputs are used for training. As discussed in Sect. 3.4, all trials start from the stable fixed point so that the initial value of ψ_1 is 0. The number of outer loop iterations is set to ensure that the training loss converges; for first-order accuracy, they are set to 300.

For the first-order accuracy approximation, the artificial neural network identifies coefficient values of $\lambda_1 = -0.073$, $I_1^0 = 1$ and $g^1 = 1.09$. For comparison, the actual coefficient values can also be obtained by utilizing the underlying model equations and computing the partial derivatives as described in [52]; these are $\lambda_1 = -0.05$, $I_1^0 = 1$ and $g^1 = 1.05$. Both the actual and the learned coefficients are nearly identical. These learned first-order coefficients are then used for initialization and regularized for subsequent training of models up to third order of accuracy. For each of the higher-order models, both the optimizer and loss functions are the same as first order, however, the learning rate is varied to ensure convergence. Also, in order to drive the model past the linear regime, a larger magnitude input set of the form $u(t) = (0.20 + \epsilon)\sin((0.05 + \epsilon/2)t)$ is used for $t \in [0, 50]$ where $\epsilon \in \{0.01, 0.02, \ldots, 0.10\}$.

For convergence during training, outer loop iterations and the learning rate are set to 300 and 0.005 respectively for the second-order model. The third-order accuracy is trained with a 0.001 learning rate for 600 iterations. Training the higher-order models results in the following set of learned coefficients: $\lambda_1 = -0.073$, $I_1^0 = 1$, $g^1 = 1.09$, $I_1^1 = -0.001$, $I_1^2 = 0.0003$, $g^2 = -0.95$ and $g^3 = -0.95$. The actual coefficient values are $\lambda = -0.05$, $I_1^0 = 1$, $g^1 = 1.05$, $I_1^1 = 0$, $I_1^2 = 0$, $g^2 = -1.11$ and $g^3 = -1.17$. Training the first-, second-, and third-order reduced model took 157, 283, and 513 s, respectively, using a desktop computer with a midgrade processor.

For validation, two Gaussian pulses of the form

$$u(t) = \frac{5}{\sqrt{200\pi}} \exp\frac{-(t-30)^2}{200} + \frac{5}{\sqrt{200\pi}} \exp\frac{-(t-100)^2}{200} \tag{22}$$

are applied. Note that this test input is different from the training inputs. Results are illustrated in Fig. 3. Panel A compares the full model simulations to the isostable-based model taken to first, second, and third orders of accuracy. Panel B shows the corresponding error, and panel C shows the test input. The accuracy of the reduced order isostable-based model improves substantially as more orders of accuracy are considered. The first-order accuracy model fails to emulate the two dips characteristic of the full model output shown in Fig 3. Moving on to second-order accuracy, the reduced model output improves significantly but it is still unable to fully capture the behavior of the full model. Finally, the third-order reduced model predicts an output almost identical to the full model as illustrated by both panel A and B in Fig 3.

4.2 Illustration in a model with dynamics near a hopf bifurcation

Next, we illustrate the proposed model identification strategy on a modified version of the radial isochron clock [59].

$$\begin{aligned}
\dot{x}_1 &= \sigma x_1(\mu - x_1^2 - x_2^2)) \\
&\quad - x_2((1 + \rho(x_1^2 + x_2^2 - \mu))) + u(t), \\
\dot{x}_2 &= \sigma x_2(\mu - x_1^2 - x_2^2)) \\
&\quad + x_1((1 + \rho(x_1^2 + x_2^2 - \mu))), \\
y(t) &= x_1(t),
\end{aligned} \tag{23}$$

where $u(t)$ is an external input applied directly to the x_1 variable. Above, we choose constants $\sigma = 0.1$, $\rho = 0.1$, and $\mu = -0.2$. Here μ is a bifurcation parameter with a stable limit cycle emerging as a result of a Hopf bifurcation when $\mu > 0$. Because $\mu < 0$ in this example, the stable fixed point has two complex conjugate eigenvalues. As such, the isostable coordinates will also be complex-valued necessitating the use of the transformation described in Sect. 3.5.

We train a model using two complex-conjugate isostable coordinates; the general architecture for the model is shown in Fig. 2. The ADAM optimizer is used with an MSE loss of the form (15) and a learning rate of 0.005. The training is implemented as described in Sect. 3.4. The timestep for the update rule from (9) is taken to be $\Delta t = 0.05$. Inputs of the form $u(t) = \epsilon(1-\theta(t-t_s))$ for $t \in [0, 50]$ are used for training where $\theta(t)$ is the Heaviside step function, ϵ sets the magnitude of the input, and t_s controls the duration of the input application. Ten training inputs are used taking $\epsilon = 0.1 + 0.01n$ and $t_s = 0.5n$ for $n = 1, \ldots, 10$.

After learning is completed, the first order of accuracy coefficients are: $\lambda_{1,2} = -0.042 \pm 1.027i$, $I_{1,2}^0 = 0.9998$ and $g^{1,2} = 0.4972 \pm 0.0245i$. Ground truth

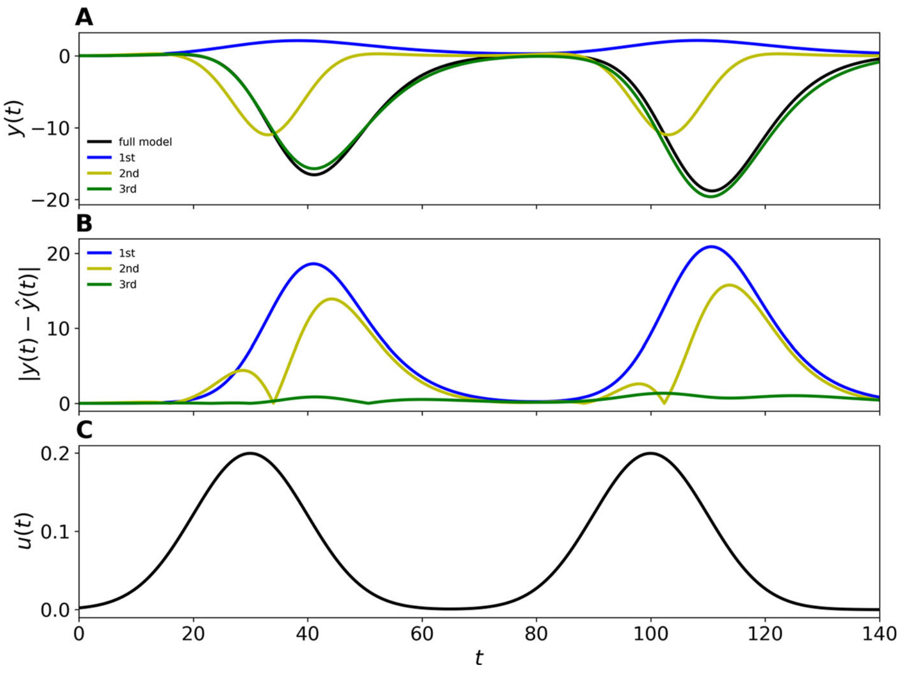

Fig. 3 The proposed model identification algorithm is applied to the simple two-dimensional model from (21). A single isostable coordinate is considered and models of the form (6) are obtained that are valid to first-, second-, and third-order accuracy in the expansion of the isostable coordinates. After training is completed, panel A shows the response to the two Gaussian pulses described by Eq. (22) and shown in Panel C. Panel B shows the corresponding error between the full and reduced order models. Qualitatively similar results are observed for other test inputs (not shown)

values for these coefficients obtained using methods discussed in [52] are $\lambda_{1,2} = -0.02 \pm 1.02i$, $I^0_{1,2} = 1$ and $g^{1,2} = 0.5$. For subsequent training of higher-order models up to fifth order of accuracy, the first-order coefficients are used for initialization and fixed through regularization. For training the higher-order models, the learning rate is varied to ensure convergence.

The second-order model is trained with the same set of inputs as first order. For other models up to fifth order, a step input set of ten trials of the form $u(t) = \epsilon(1 - \theta(t - t_s))$ for $t \in [0, 50]$ is used where $\epsilon = 0.2 + 0.01n$ with $t_s = 0.5n$ taking $n = 1, \ldots, 10$. These larger magnitude inputs are used here to drive the state farther from the fixed point to better capture the contribution from nonlinear terms. These learned higher-order coefficients are close to their actual values obtained directly from the model equations. Training the first-, second-, third-, fourth-, and fifth-order reduced model takes 123, 149, 210, 424, and 647 s, respectively, to achieve convergence using a desktop computer with a midgrade processor. The increased learning time for the higher-order accuracy models can be attributed to the growing number of coefficients at higher orders of accuracy, i.e., that comprise the Taylor expansions from Eqs. (7) and (8).

Learned models are validated using test stimuli $u(t) = \alpha \sin(0.5t)$ for $\alpha = 0.15$ and 0.3. Note that these are different than the training stimuli. Panel A of Fig. 4 shows output in response to the inputs $u(t) = 0.15 \sin(0.5t)$ and $u(t) = 0.30 \sin(0.5t)$. Panel B shows the associated magnitude of the error and Panel C shows the applied input for reference. For the lower magnitude inputs, third-, fourth-, and fifth-order models have substantially smaller errors than the first-order accurate models. Larger errors are observed for larger magnitude inputs. Additionally, there is little difference between the third-, fourth-, and fifth-order accurate models indicating diminishing returns at higher orders of accuracy. Further elaborating on the results depicted in Fig 4, one can see that for the smaller magnitude test input results shown in Panel A, the predicted outputs for all orders of accuracy are nearly identical to the full model output. However, the difference between 1st-, 2nd- and higher-order accuracy predicted outputs is more apparent in Panel B; error for first-order accuracy is the highest followed by second order. The same pattern is observed in Panel E for the larger test input with identical errors observed for higher-order accuracy predicted outputs even though the magnitude of errors in Panel E is higher.

Additional validation for the learned models is done by applying step function inputs of the form $u(t) = 0.01(1 - \theta(t - 10))$. Results are shown in Fig. 5. Panels A and Panel B follow the same pattern as observed

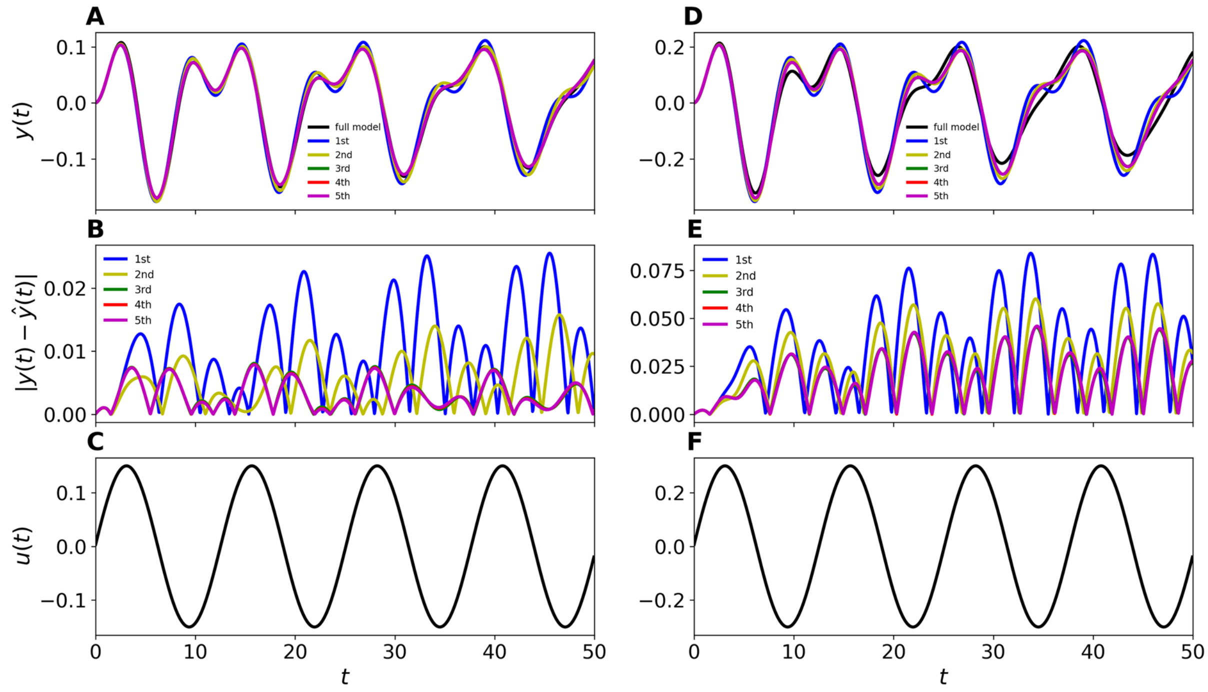

Fig. 4 The proposed model identification algorithm is applied to Eq. (23). Two complex-conjugate isostable coordinates are considered and models of the form (6) are obtained that are valid to first through fifth orders of accuracy in the expansion of the isostable coordinates. Panels A and B compare outputs from full and isostable-coordinate-based models when applying the low amplitude sinusoidal input from panel C. Panels D and E compare results from the full and isostable-coordinate-based models when applying a larger magnitude sinusoidal input shown in panel F. Note that these test inputs are different than the step function inputs used for training

in Fig 4. First-order accuracy model is able to replicate the characteristics of the full model output to a certain extent albeit with a large difference in magnitude when input from Panel C is applied to the system. The second-order accuracy model is more accurate. Even higher-order accuracy models generate outputs almost identical to the full model with little difference between third, fourth and fifth order.

4.3 Spike rates of neural populations

As a final example, we consider the proposed model identification strategy on a more complicated system that captures neural spiking rates in a large, coupled population of neurons in response to external inputs. Model equations are based on a model for thalamic neurons from [37]:

$$\begin{aligned}
C\dot{V}_i &= -I_{\mathrm{L}}(V_i) - I_{\mathrm{Na}}(V_i,h_i) - I_{\mathrm{K}}(V_i,h_i) - I_{\mathrm{T}}(V_i,r_i) \\
&\quad + I_i^b - \frac{g_{syn}}{N}\sum_{j=1}^{N} s_j(V_i - E_{syn}) + \sqrt{2D}\eta_i(t) + u(t), \\
\dot{h_i} &= (h_\infty(V_i) - h_i)/\tau_h(V_i), \\
\dot{r_i} &= (r_\infty(V_i) - r_i)/\tau_r(V_i), \\
\dot{s_i} &= \frac{a(1-s)}{1+\exp(-(V_i - V_T)/\sigma_T)} - bs_i. \qquad (24)
\end{aligned}$$

Here, we consider $N = 1000$ total neurons in the population, V_i, h_i, and r_i represent the transmembrane voltage and two gating variables, respectively, that determine the ionic currents for neuron i, s_i is a variable that governs the synaptic coupling with the conductance $g_{syn} = 0.4\text{mS/cm}^2$ determining the coupling strength, $E_{syn} = -100$ mV is the reversal potential of the neurotransmitter so that coupling is inhibitory, and $C = 1\mu F/\text{cm}^2$ is the membrane capacitance. Parameters $a = 3$, $V_t = -20$ mV, $\sigma_T = 0.8$ mV, and $b = 1$ determine the synaptic current. Noise is incorporated with independent and identically distributed zero-mean

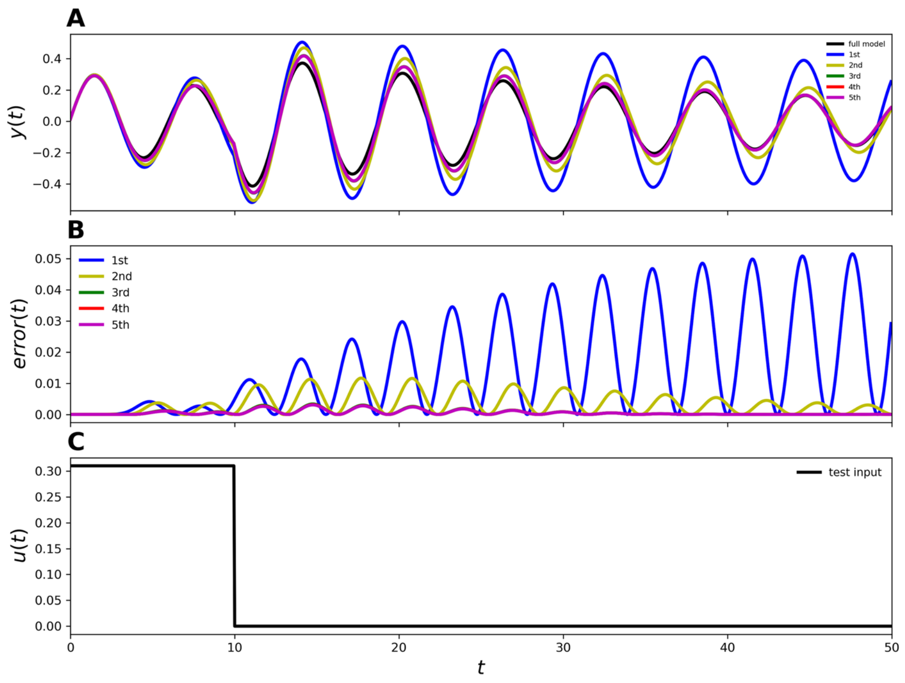

Fig. 5 Panels A and B compare outputs from full and isostable-coordinate-based models when applying a step function input as shown in panel C. As shown in Fig 4, results from first to fifth order of accuracy are presented in the figure with results following the same trend as observed before

white noise $\sqrt{2D}\eta_i(t)$ added to the voltage variable of each neuron taking $D = 1$ to be the noise intensity. The baseline current of the i^{th} neuron, I_i^b, is drawn from a normal distribution with a mean of 5 and a variance of $1\mu\text{A}/\text{cm}^2$. $u(t)$ represents a transmembrane current applied identically to each neuron. Reference [37] contains a full description of the remaining ionic currents and gating variables. A different isostable-coordinate-based model identification technique was also considered in [52]. For the neural model (24), we consider using the firing rate as the observable defined according to

$$y(t) = \frac{\text{Number of action potentials on the interval } [t - W, t]}{W}, \tag{25}$$

where $W = 5$ ms is the width of the window and an action potential is defined to occur the moment that the transmembrane voltage crosses -25 mV with a positive slope.

In the absence of coupling, noise, and input, each neuron from (24) is in the tonically firing regime with action potentials that occur with an interspike interval of 8.4 milliseconds. When considering the aggregate behavior of (24) when $u(t) = 0$, coupling is not strong enough to overcome noise so that the distribution of neural phases approaches a steady state and the firing rate settles to a steady value of 91.6 action potentials per millisecond. The firing rate can be modulated through the application of input with behavior qualitatively similar to a stable sink, i.e., with two complex-conjugate eigenvalues. As such, we apply our proposed model identification strategy using two complex-valued isostable coordinates necessitating the use of the transformation described in Sect. 3.5. The form of the inferred reduced order model is identical to the one obtained in Sect. 4.2; the important difference here is that the data used for training comes from a system with substantially higher dimension.

We train a model using two complex-conjugate isostable coordinates accurate to various orders of accuracy. The network is based upon the architecture represented in Fig. 2. Inputs of the form $u(t) = \epsilon(1 - \theta(t - t_s))$ for $t \in [0, 50]$ are used for training where $\theta(t)$ is the Heaviside step function, ϵ sets the magnitude of the input, and t_s controls the duration of the input application. Ten trials are considered for training taking $\epsilon = 2 + 0.1n$ and $t_s = 10 + 2n$ for $n = 1, \ldots, 10$. The presence of Gaussian noise in the dynamical model (24) interferes with the training leading to slower convergence. In order to mitigate the effects of noise, the stimuli described above are applied for ten identical trials and the model output $y(t)$ is taken to be the average over the set of trials. Once these representative outputs have been obtained, the ADAM optimizer is used with an MSE loss of the form (15) taking the learning rate to be 0.005. The timestep for the update rule from Eq. (9) is taken to be $\Delta t = 0.05$ ms. The training is implemented as described in Sect. 3.4.

Due to the complexity of the underlying equations that comprise the model (24), it is not possible to compute the ground truth for the approximation of the coefficients for the Taylor expansions from (7) and (8). Instead, we must compare the predicted output between the full order model and the inferred reduced order model in order to gauge accuracy using inputs that are distinct from those that were used for training. As done for the previous two models, the first-order coefficients are obtained first and subsequently used for initialization when training models to second- and third-order accuracy. As described in Sect. 3.5, for training the learned weights are categorized into shared and distinct weights. Training the first-, second-, and third-order reduced model took 82, 105, and 1139 s, respectively, to achieve convergence using a desktop computer with a midgrade processor. The jump between the second and third order of accuracy models is related to the use of a smaller learning rate for the third-order model in order to achieve convergence.

Learned models are validated using a test stimulus $u(t) = 0.3(\sin(0.1t) + \cos(0.23t) + \sin(-0.49t))$. Results are shown in Fig. 6. Panel A shows the first-, second-, and third-order reduced order model output compared to the full order model output in response to the input from panel C. Note that the reduce order models do not account for noise. Panel B shows the associated error. The first-order accuracy model has a MSE value of 9.2; 2nd-order accuracy subsequently reduces the MSE to 6.1 with 3rd-order reduced model giving the lowest error value of 5.5.

5 Discussion

As evidenced by the results from previous sections, the proposed framework performs well in emulating the full model dynamics and requires relatively little data. For all the three examples shown, the models were trained on a set of ten inputs highlighting the method's ability to learn from limited data in an efficient manner. We emphasize that for each model, qualitatively different inputs were used for testing and training. Moreover, the isostable coordinates, as functions on state space incorporated within the neural network structure, do not need any prior estimates; instead, the initial isostable coordinate is assumed to be zero for the steady-state solution, i.e., at the fixed point. Subsequent isostable coordinate estimates are generated and optimized iteratively using the isostable-based neural network structure as the training proceeds.

There are a number of extensions that would be worthwhile to consider in future work for improving the accuracy and effectiveness of the presented method. Foremost, here we only consider MSE for the loss function as defined in Eq. (15) for training the artificial neural network. While this loss function provides adequate performance in the examples considered here, it does have a few notable shortcomings. For instance, predicted outputs that are qualitatively similar but slightly time shifted with respect to the true outputs are heavily penalized. This presents an issue in situations where the input causes oscillations in the full model. When the reduced order model is unable to perfectly match the oscillation frequencies, a slight timing shift can yield a significant increase in the MSE even though the qualitative behavior is nearly identical. It may be possible to mitigate this issue by investigating other loss functions that directly consider the frequency or amplitude of these forced oscillations. Additionally, during implementation of the inner loop during training, i.e. where relation (9) is evaluated recursively to obtain $\hat{y}(t)$, initial errors at early timesteps will be propagated and potentially amplified at later timesteps. It would likely be worthwhile to modify the cost function with an additional weighting term, for instance, using $\sum_{i=1}^{\eta} \upsilon(i)(y(t_0 + i\Delta t) - \hat{y}(t_0 + i\Delta t))^2$ where $\upsilon(i)$ confers additional importance to initial solutions in order to limit this subsequent error propagation.

There are also improvements which are worth considering from a deep learning perspective. For example, for the three dynamical systems considered in this work, there was no normalization applied to either the inputs nor the outputs of the neural network. It has been shown in various works that batch normalization [16,38] not only makes the neural network's convergence faster but also more stable during training. This could be useful for more complex dynamical systems with additive noise (such as the neural population example considered in this work). The existence of additive noise alongside the inherent model complexities, potentially makes it more difficult for the neural network optimizer used in training to learn an opti-

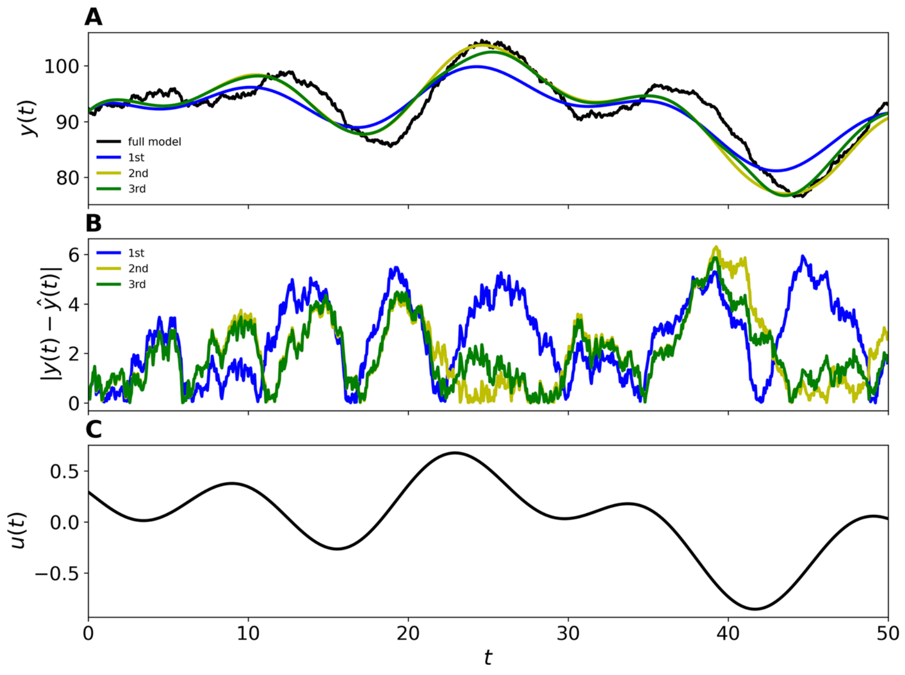

Fig. 6 The proposed model identification algorithm is applied to the neural model from Eq. (24) with output (25). Two complex-conjugate isostable coordinates are considered and models of the form (6) are obtained that are valid to first, second, and third orders of accuracy in the expansion of the isostable coordinates. Panel A shows the response to the applied input from panel C. Panel B shows the corresponding error between the full and reduced order models. Note that the applied input here is different from the step function inputs used for training

mal set of coefficients for minimizing the loss. Thus, a workaround could be devised to incorporate normalization into the presented framework without altering the isostable based network structure helping to tackle the convergence problem. In addition, even though the method is able to give accurate results with a limited amount of training data for the examples considered in this work, it would certainly be worthwhile to extend the training data-set to include more samples in future work, especially when more complicated models are considered.

The proposed model identification strategy uses an asymptotic approximation in a basis of isostable coordinates to capture the nonlinear dynamics of the reduced order model (6). While this strategy can be useful to obtain models that are more accurate than their linear counterparts, diminishing returns at higher orders of accuracy and the growing combinatorial complexity of this expansion at higher orders limit the ability to consider the behavior in response to exceedingly large inputs. It may be of interest to employ adaptive isostable-coordinate-based strategies [57] which consider the dynamical behavior in reference to a continuous family of fixed points in conjunction with the proposed model identification algorithm to circumvent the need for taking the expansion to particularly large orders of accuracy. Furthermore, as mentioned earlier, the current model identification algorithm is only valid for systems with fixed point dynamics. It may be possible to adapt the phase-isostable-based model order reduction strategies discussed in [50] for use with the model identification algorithms proposed in this work with an ultimate goal of considering systems with more complicated dynamics. Finally, the efficacy of the presented strategy might further improve if the isostable coordinates themselves could be inferred from the output (i.e., the observable y) and then used in conjunction with the neural network for learning the coefficients. A related strategy has been used for extracting the isostable coordinate components in [1,49] with time delay embeddings of observables and it may be worthwhile to incorporate these approaches into the proposed model identification strategy.

6 Conclusion

In this work, we propose a strategy for inferring nonlinear dynamical models for general nonlinear dynamical system in the basin of attraction of a stable fixed point. Representing the dynamics using a transformed basis of isostable coordinates according to Eq. (6), the rapidly decaying isostable coordinates are truncated allowing for a low-order representation. The gradient of the remaining isostable coordinates and the isostable-to-output relationships are represented according to the asymptotic expansions from Eqs. (7) and (8), respectively. We subsequently reframe the state update rules for this reduced order model in terms of the composition of known nonlinear functions and unknown linear functions. This reframing allows for the use of an artificial neural network to identify an accurate approximation for the unknown weights of the linear relationships from training data. We illustrate the model identification strategy in a collection of nonlinear models including a synaptically coupled population of conductance based neurons.

We emphasize that while the approach used here considers an asymptotic expansion in the basis of isostable coordinates centered at a fixed point, the underlying model dynamics themselves do not necessarily need to approach a fixed point. This subtle point is illustrated in the example from Section (4.3). When synaptic coupling is small, because of noise, the distribution of neural phases tends to approach a stable stationary solution that is reflected in the aggregate behavior captured by the firing rate, i.e., the output from Eq. (25). As such, the proposed model identification algorithm learns a representation for this aggregate behavior and not the behavior of the individual oscillators.

In contrast to methods that learn sparse representations for governing model equations [8,22,31,40], the proposed method assumes a universal underlying model structure of the form (6) which is valid for systems with stable fixed points and does not require the specification of a library of candidate functions. In connection to other Koopman-based model identification algorithms such as Koopman model predictive control [2,18], extended DMD [46], and strategies that use delay embeddings [3,49], the proposed strategy explicitly considers a subset of the slowest decaying Koopman eigenmodes in the fitting procedure helping to mitigate issues associated with overfitting. Indeed, the resulting reduced order models accurately predict response to inputs that were not used for training, for instance, the coupled neuron model was trained using a collection of step function inputs but the learned model accurately reflects the behavior in response to a continuous input.

A related model identification technique was considered in previous work [52], where relationships between the coefficients of the expansions from (7) and (8) and the steady-state output in response to sinusoidal forcing were identified and subsequently used to infer nonlinear isostable-based models using least-squares fitting techniques. The strategy proposed in this work improves on the strategy from [52] in two important ways: first, the proposed learning-based strategy evaluates the performance of the learned models by comparing the output of the learned model directly to the output obtained from training data. By contrast, the method from [52] considers the steady-state amplitude of various Fourier modes in response to purely sinusoidal forcing. The amplitudes of these modes can be quite sensitive to noise or other uncertainties, especially when considering high accuracy expansions from Eqs. (7) and (8); small errors in the inference of these higher-order terms can have profound impacts on the overall accuracy of the resulting models. Second, the proposed model identification strategy can be implemented using inputs of any form for training. Conversely, the strategy from [52] requires the steady-state response to sinusoidal inputs which would limit practical utility in situations where sinusoidal inputs cannot feasibly be applied and in situations where convergence to steady state is slow.

This material is based on work supported by National Science Foundation Grant No. CMMI-1933583. The datasets generated during and/or analysed during the current study are available from the corresponding author on reasonable request.

References

1. Ahmed, T., Wilson, D.: Exploiting circadian memory to hasten recovery from circadian misalignment. Chaos Interdiscipl. J. Nonlinear Sci. **31**(7), 073130 (2021)
2. Arbabi, H., Korda, M., Mezic, I.: A data-driven Koopman model predictive control framework for nonlinear flows (2018). arxiv:1804.05291
3. Arbabi, H., Mezic, I.: Ergodic theory, dynamic mode decomposition, and computation of spectral properties of the Koopman operator. SIAM J. Appl. Dyn. Syst. **16**(4), 2096–2126 (2017)
4. Avila, A.M., Mezić, I.: Data-driven analysis and forecasting of highway traffic dynamics. Nat. Commun. **11**(1), 1–16 (2020)
5. Berkooz, G., Holmes, P., Lumley, J.L.: The proper orthogonal decomposition in the analysis of turbulent flows. Annu. Rev. Fluid Mech. **25**(1), 539–575 (1993)
6. Brunton, S.L., Brunton, B.W., Proctor, J.L., Kaiser, E., Kutz, J.N.: Chaos as an intermittently forced linear system. Nat. Commun. **8**(1), 1–9 (2017)

7. Brunton, S.L., Kutz, J.N.: Data-Driven Science and Engineering: Machine Learning, Dynamical Systems, and Control. Cambridge University Press, New York (2019)
8. Brunton, S.L., Proctor, J.L., Kutz, J.N.: Discovering governing equations from data by sparse identification of nonlinear dynamical systems. Proc. Natl. Acad. Sci. **113**(15), 3932–3937 (2016)
9. Budišić, M., Mohr, R., Mezić, I.: Applied Koopmanism. Chaos Interdiscipl. J. Nonlinear Sci. **22**(4), 047510 (2012)
10. Cenedese, M., Axås, J., Yang, H., Eriten, M., Haller, G.: Data-driven nonlinear model reduction to spectral submanifolds in mechanical systems (2021). arXiv:2110.01929
11. Geneva, N., Zabaras, N.: Transformers for modeling physical systems (2020). arXiv:2010.03957
12. Haller, G., Ponsioen, S.: Nonlinear normal modes and spectral submanifolds: existence, uniqueness and use in model reduction. Nonlinear Dyn. **86**(3), 1493–1534 (2016)
13. Hirose, A.: Complex-Valued Neural Networks: Advances and Applications. John Wiley & Sons, New York (2013)
14. Hodgkin, A.L., Huxley, A.F.: A quantitative description of membrane current and its application to conduction and excitation in nerve. J. Physiol. **117**(4), 500–544 (1952)
15. Holmes, P., Lumley, J.L., Berkooz, G., Rowley, C.W.: Turbulence, Coherent Structures, Dynamical Systems and Symmetry. Cambridge University Press, New York (1996)
16. Ioffe, S., Szegedy, C.: Batch normalization: Accelerating deep network training by reducing internal covariate shift. In: *International Conference on Machine Learning*, pp. 448–456. PMLR (2015)
17. Kaiser, E., Kutz, J.N., Brunton, S.: Data-driven discovery of Koopman eigenfunctions for control. Mach. Learn. Sci. Technol. **2**, 035023 (2021)
18. Korda, M., Mezić, I.: Linear predictors for nonlinear dynamical systems: Koopman operator meets model predictive control. Automatica **93**, 149–160 (2018)
19. Kutz, J.N., Brunton, S.L., Brunton, B.W., Proctor, J.L.: Dynamic Mode Decomposition: Data-Driven Modeling of Complex Systems. Society for Industrial and Applied Mathematics, Philadelphia, PA (2016)
20. Kvalheim, M.D., Revzen, S.: Existence and uniqueness of global Koopman eigenfunctions for stable fixed points and periodic orbits. Phys. D Nonlinear Phenom. **425**, 132959 (2021)
21. Lusch, B., Kutz, J.N., Brunton, S.L.: Deep learning for universal linear embeddings of nonlinear dynamics. Nat. Commun. **9**(1), 1–10 (2018)
22. Mangan, N.M., Askham, T., Brunton, S.L., Kutz, J.N., Proctor, J.L.: Model selection for hybrid dynamical systems via sparse regression. Proc. Royal Soc. A **475**(2223), 20180534 (2019)
23. Masri, S.F., Chassiakos, A.G., Caughey, T.K.: Structure-unknown non-linear dynamic systems: identification through neural networks. Smart Mater. Struct. **1**(1), 45 (1992)
24. Mauroy, A., Mezić, I., Moehlis, J.: Isostables, isochrons, and Koopman spectrum for the action-angle representation of stable fixed point dynamics. Phys. D **261**, 19–30 (2013)
25. Mezić, I.: Analysis of fluid flows via spectral properties of the Koopman operator. Annu. Rev. Fluid Mech. **45**, 357–378 (2013)
26. Mezić, I.: Spectrum of the Koopman operator, spectral expansions in functional spaces, and state-space geometry. J. Nonlinear Sci. **30**, 1–55 (2019)
27. Noack, B.R., Afanasiev, K., Morzynski, M., Tadmor, G., Thiele, F.: A hierarchy of low-dimensional models for the transient and post-transient cylinder wake. J. Fluid Mech. **497**, 335–363 (2003)
28. Omidi, M., Arab, B., Rasanan, A.H., Rad, J.A., Parand, K.: Learning nonlinear dynamics with behavior ordinary/partial/system of the differential equations: looking through the lens of orthogonal neural networks. Eng. Comput. **38**(2), 1635–1654 (2022)
29. Ortin, S., Gutierrez, J.M., Pesquera, L., Vasquez, H.: Nonlinear dynamics extraction for time-delay systems using modular neural networks synchronization and prediction. Phys. A **351**(1), 133–141 (2005)
30. Pan, S., Duraisamy, K.: Long-time predictive modeling of nonlinear dynamical systems using neural networks. Complexity (2018). https://doi.org/10.1155/2018/4801012
31. Pantazis, Y., Tsamardinos, I.: A unified approach for sparse dynamical system inference from temporal measurements. Bioinformatics **35**(18), 3387–3396 (2019)
32. Peitz, S., Klus, S.: Koopman operator-based model reduction for switched-system control of PDEs. Automatica **106**, 184–191 (2019)
33. Ponsioen, S., Jain, S., Haller, G.: Model reduction to spectral submanifolds and forced-response calculation in high-dimensional mechanical systems. J. Sound Vib. **488**, 115640 (2020)
34. Proctor, J.L., Brunton, S.L., Kutz, J.N.: Dynamic mode decomposition with control. SIAM J. Appl. Dyn. Syst. **15**(1), 142–161 (2016)
35. Raissi, M., Perdikaris, P., Karniadakis, G.E.: Physics-informed neural networks: a deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. J. Comput. Phys. **378**, 686–707 (2019)
36. Rowley, C.W., Mezić, I., Bagheri, S., Schlatter, P., Henningson, D.S.: Spectral analysis of nonlinear flows. J. Fluid Mech. **641**, 115–127 (2009)
37. Rubin, J.E., Terman, D.: High frequency stimulation of the subthalamic nucleus eliminates pathological thalamic rhythmicity in a computational model. J. Comput. Neurosci. **16**(3), 211–235 (2004)
38. Santurkar, S., Tsipras, D., Ilyas, A., Madry, A.: How does batch normalization help optimization? Adv. Neural Inf. Process. Syst. 31 (2018)
39. Scardapane, S., Van Vaerenbergh, S., Hussain, A., Uncini, A.: Complex-valued neural networks with nonparametric activation functions. IEEE Trans. Emerg. Topics Comput. Intell. **4**(2), 140–150 (2018)
40. Schaeffer, H.: Learning partial differential equations via data discovery and sparse optimization. Proc. Royal Soc. A: Math. Phys. Eng. Sci. **473**(2197), 20160446 (2017)
41. Schmid, P.J.: Dynamic mode decomposition of numerical and experimental data. J. Fluid Mech. **656**, 5–28 (2010)
42. Suresh, S., Sundararajan, N., Savitha, R.: Supervised Learning with Complex-Valued Neural Networks. Springer, Berlin (2013)
43. Szalai, E., Ehrhardt, D., Haller, G.: Nonlinear model identification and spectral submanifolds for multi-degree-

of-freedom mechanical vibrations. Proc. Royal Soc. A: Math. Phys. Eng. Sci. **473**(2202), 20160759 (2017)
44. Tan, Yonghong, Saif, Mehrdad: Neural-networks-based nonlinear dynamic modeling for automotive engines. Neurocomputing **30**(1–4), 129–142 (2000)
45. Towne, A., Schmidt, O.T., Colonius, T.: Spectral proper orthogonal decomposition and its relationship to dynamic mode decomposition and resolvent analysis. J. Fluid Mech. **847**, 821–867 (2018)
46. Williams, M.O., Kevrekidis, I.G., Rowley, C.W.: A data-driven approximation of the Koopman operator: Extending dynamic mode decomposition. J. Nonlinear Sci. **25**(6), 1307–1346 (2015)
47. Wilson, D.: Isostable reduction of oscillators with piecewise smooth dynamics and complex Floquet multipliers. Phys. Rev. E **99**(2), 022210 (2019)
48. Wilson, D.: An optimal framework for nonfeedback stability control of chaos. SIAM J. Appl. Dyn. Syst. **18**(4), 1982–1999 (2019)
49. Wilson, D.: A data-driven phase and isostable reduced modeling framework for oscillatory dynamical systems. Chaos Interdiscipl. J. Nonlinear Sci. **30**(1), 013121 (2020)
50. Wilson, D.: Phase-amplitude reduction far beyond the weakly perturbed paradigm. Phys. Rev. E **101**(2), 022220 (2020)
51. Wilson, D.: Analysis of input-induced oscillations using the isostable coordinate framework. Chaos Interdiscipl. J. Nonlinear Sci. **31**(2), 023131 (2021)
52. Wilson, D.: Data-driven inference of high-accuracy isostable-based dynamical models in response to external inputs. Chaos Interdiscipl. J. Nonlinear Sci. **31**(6), 063137 (2021)
53. Wilson, D.: Optimal control of oscillation timing and entrainment using large magnitude inputs: an adaptive phase-amplitude-coordinate-based approach. SIAM J. Appl. Dyn. Syst. **20**(4), 1814–1843 (2021)
54. Wilson, D.: An adaptive phase-amplitude reduction framework without $\mathcal{O}(\epsilon)$ constraints on inputs. SIAM J. Appl. Dyn. Syst. **21**(1), 204–230 (2022)
55. Wilson, D.: Data-driven identification of dynamical models using adaptive parameter sets. Chaos Interdiscipl. J. Nonlinear Sci. **32**(2), 023118 (2022)
56. Wilson, D., Djouadi, S.: Isostable reduction and boundary feedback control for nonlinear convective flows. In *2019 IEEE 58th Conference on Decision and Control*, pages 2138–2143. IEEE (2019)
57. Wilson, D., Djouadi, S.M.: Adaptive isostable reduction of nonlinear PDEs with time varying parameters. IEEE Control Syst. Lett. **5**(1), 187–192 (2020)
58. Wilson, D., Moehlis, J.: Extending phase reduction to excitable media: theory and applications. SIAM Rev. **57**, 201–222 (2015)
59. Winfree, A.: The Geometry of Biological Time, 2nd edn. Springer, New York (2001)
60. Wray, Jonathan, Green, Gary GR.: Calculation of the volterra kernels of non-linear dynamic systems using an artificial neural network. Biol. Cybern. **71**(3), 187–195 (1994)
61. Yeung, E., Kundu, S., Hodas, N.: Learning deep neural network representations for Koopman operators of nonlinear dynamical systems. In: *2019 American Control Conference*, pp. 4832–4839. IEEE, (2019)