Deep learning to discover and predict dynamics on an inertial manifold

Alec J. Linot and Michael D. Graham*

Department of Chemical and Biological Engineering, University of Wisconsin—Madison, Madison Wisconsin 53706, USA

(Received 16 December 2019; accepted 20 May 2020; published 18 June 2020)

A data-driven framework is developed to represent chaotic dynamics on an inertial manifold (IM) and applied to solutions of the Kuramoto-Sivashinsky equation. A hybrid method combining linear and nonlinear (neural-network) dimension reduction transforms between coordinates in the full state space and on the IM. Additional neural networks predict time evolution on the IM. The formalism accounts for translation invariance and energy conservation, and substantially outperforms linear dimension reduction, reproducing very well key dynamic and statistical features of the attractor.

DOI: 10.1103/PhysRevE.101.062209

I. INTRODUCTION

Partial differential equations are formally infinite-dimensional, but the presence of dissipation (through viscosity or diffusion, for example) leads to the expectation that the long-time dynamics collapse onto a finite-dimensional invariant manifold [1]. Specifically, for some systems, including the Kuramoto-Sivashinsky equation (KSE) that we consider here, it can be proven that all initial conditions exponentially approach an *inertial manifold* $\mathcal{M}$ of finite dimension $d_{\mathcal{M}}$ [2], on which the long time dynamics evolve. For states on the IM, $v \in \mathcal{M}$, one can in principle find a coordinate transformation $h = \chi(v)$ to coordinates h on the inertial manifold, a change of coordinates $v = \check{\chi}(h)$ back to the full space, and a dynamical system $h(t+\tau) = F(h(t))$ on $\mathcal{M}$. [Alternately, one could represent the dynamical system in differential form $dh/dt = G(h)$.] This dynamical system is an exact reduced-order model (ROM). Such a model can be practically useful, for computationally efficient simulations of a complex process, and may also be fundamentally important, since the coordinates h represent the key dynamical variables for the phenomenon of interest.

In the present work, we use "data" in the form of chaotic solutions to the KSE with periodic boundary conditions, to find neural-network (NN) representations of the functions χ, $\check{\chi}$, and F. Many prior studies of inertial manifolds, and approximations thereof, take the inertial manifold to be the graph of a function Φ such that $\mathcal{M} := \{v_+ + \Phi(v_+)\}$, where $v_+ = Pv$ is a projection onto the $d_{\mathcal{M}}$ leading eigenfunctions of the linear operator for the PDE [3–5]. The present work is not subject to this restriction. Furthermore, our formalism explicitly accounts for the important physical features of translation invariance and energy conservation found in this system, and reproduces, very well, with a minimal number of degrees of freedom, key dynamic, and statistical features of the attractor.

A standard machine learning method for nonlinear dimension reduction is the undercomplete autoencoder [6,7]. This is a pair of neural networks, one mapping from a high-dimensional space to a low-dimensional one, and the second doing the reverse. The networks take data u as input, compute an output $\tilde{u}$, and are trained to minimize a loss function $L = ||u - \tilde{u}||^2$ summed over a batch of data vectors u. Autoencoders have been used for nonlinear dimension reduction in many applications [6], including turbulent flow fields [8,9]. For dynamical systems, autoencoders have been used to explicitly yield coordinate transformations on which the dynamics are linear [10,11] (e.g., to determine eigenmodes of the Koopman operator) or have a sparse representation [12]. Gonzalez *et al.* [13] have combined autoencoders with nonlinear time-evolution models for reconstruction of dynamics of isotropic turbulence and lid-driven cavity flow. Lee and Carlberg [14] illustrate nonlinear model reduction via NNs and an application to the dynamics of Burgers' equation. The physical interpretation of NN representations of physical phenomena has been explored in Iten *et al.* [15].

Other studies have focused on developing NNs for evolving equations with chaotic dynamics without nonlinear dimension reduction. An early example of this is González-García *et al.* [16], who used a NN to predict the right-hand side of the discretized KSE. Specifically, they input the full state of the KSE and its derivatives into a NN to predict the parameters of a Runge-Kutta method for time integration. More recently, larger NN have been used for prediction. For example, Pathak *et al.* [17] showed that a reservoir network trained with time-evolution "data" from the KSE was capable of making excellent predictions of future time evolution. No explicit model reduction was performed. In Ref. [18], proper orthogonal decomposition (POD) and a spectral version (SPOD), both linear dimension reduction techniques, were used to reduce the dimension of fluid flow data that were then used to train a NN for time evolution. Vlachas *et al.* [19] combined various linear dimension reduction approaches with a long-short term memory NN and mean stochastic modeling to keep trajectories on the attractor. Similarly, in Ref. [20] a long-short term memory NN was used in a nonlinear Galerkin approach to estimate the nonlinearity, a task often achieved by assuming lower modes evolve slowly, and iteratively solving for the higher modes [5,21–24].

*mdgraham@wisc.edu

 ©2020 American Physical Society

Although methods exist for modeling dynamics on an IM, estimating $d_{\mathcal{M}}$ remains a difficult problem. In Ref. [25], an autoencoder is used to estimate $d_{\mathcal{M}}$ from data for the dynamics of the complex Ginzburg-Landau (CGL) equation; however, the dynamics on $\mathcal{M}$ are not modeled. A dynamical approach to determining $d_{\mathcal{M}}$ was taken in Ref. [26], where the covariant Lyapunov vectors of trajectories the KSE and the CGL were found to decompose into "physical" and "isolated" modes. Physical modes are entangled, in the sense that tangencies between them result in perturbations of a single mode effecting other modes, whereas isolated modes lack tangencies with physical or isolated modes. This suggests the number of physical modes corresponds to $d_{\mathcal{M}}$. Expanding on that work, Ding *et al.* [27] found the dimension for the KSE in similar ways using Floquet vectors from an ensemble of unstable periodic orbits that are close to the chaotic attractor. The present work combines data-driven dimension reduction and time evolution using an efficient autoencoder structure that incorporates translation symmetry and energy conservation.

II. FORMULATION

Our testbed for this approach is the KSE,

$$\partial_t v = -v\partial_x v - \partial_{xx} v - \partial_{xxxx} v, \tag{1}$$

with periodic boundary conditions in the domain $x \in [0, L]$. We select $L = 22, 44$, and 66 because these domain sizes yield increasingly chaotic dynamics, and $d_{\mathcal{M}}$ is known at $L = 22$ [27]. Solutions to this equation are only unique to within a translation that we will represent with a phase variable $\phi \in \mathbb{R}$. This equation has an energy conservation principle: when time-averaged, the energy production rate $\mathcal{P} = \langle \partial_x v \partial_x v \rangle$ balances the dissipation rate $\varepsilon = \langle \partial_{xx} v \partial_{xx} v \rangle$. Here $\langle \cdot \rangle$ represents averaging over x. These properties are incorporated into the dimension reduction formulation as detailed below. Trajectories of (1) were generated using a Fourier spectral method in space and a fourth-order time integration scheme [28] with the code available from Cvitanović *et al.* [29]. The solution $v(x)$ is represented on a uniformly spaced mesh of $d = 64$ points; we denote the solution on this mesh as $u \in \mathbb{R}^d$, so d is the dimension of the full state space in the present system.

III. METHODOLOGY AND RESULTS

Figure 1 illustrates our framework for finding the inertial manifold and the dynamical system on it. The first step of the process exploits translation invariance: The solution u at every time instant is transformed into a pattern $\hat{u} \in \mathbb{R}^d$ and a phase ϕ using an approach called the "method of slices" [30,31]. Factoring out the phase leads to a more compact representation of the data by eliminating the need of training redundant weights for translated signals (e.g., the representation will not need to separately represent $\sin 2\pi x/L$ and all of its translations). Furthermore, such "symmetry reduction" methods have been found to help elucidate the state space structure in fluid mechanics problems such as pipe flow [32].

The method of slices involves taking the discrete Fourier transform $\mathcal{F}$ of the data in x to yield $a(t) = \mathcal{F}\{u(t)\}$. With the data in Fourier space, the phase of the first Fourier mode

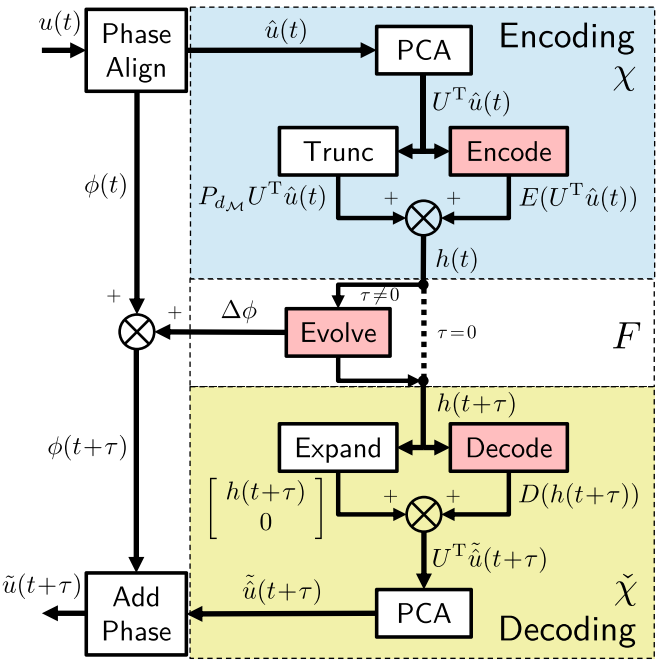

FIG. 1. Block diagram for the hybrid autoencoder and time-evolution scheme. NNs are pink (light gray).

$a_1(t)$ is found using $\phi(t) = \mathrm{atan2}\{\mathrm{Im}[a_1(t)], \mathrm{Re}[a_1(t)]\}$. Now we can construct a phase-aligned solution $\hat{u}(t)$ so that its first Fourier mode is a pure cosine: $\hat{u}(t) = \mathcal{F}^{-1}\{a(t)e^{-ik\phi(t)}\}$. Storage of ϕ for each time instant allows conversion of $\hat{u}$ back to u [i.e., $u(t)$ contains both $\phi(t)$ and $\hat{u}(t)$]. Times when $a_1(t)$ approaches zero require special treatment, as was recognized by Budanur *et al.* [30]. We use the solution they proposed, which is to stretch time according to $\Delta\hat{t} = \Delta t/|a_1(t)|$. The rescaled time $\hat{t}$ is called "in-slice" time. Given any $u(t)$ we can always find $\hat{u}(t)$, $\phi(t)$ and $a_1(t)$, so it is always possible to move back and forth between the original and phase-aligned solutions and between real and in-slice time. The data used for training the NNs were $\sim$4–5 $\times 10^5$ solutions u separated by $\Delta\hat{t} = 0.2$ "in-slice" time units, which corresponds to $\Delta t \approx 0.023$ time units on average for $L = 22$. Data were gathered after the dynamics had settled onto the attractor.

Given the phase-aligned pattern data $\hat{u}$, the first machine learning task is to find the manifold $\mathcal{M}$, of dimension $d_{\mathcal{M}}$, on which these data live, or equivalently the coordinate tranformations $h = \chi(\hat{u})$ and $\hat{u} = \check{\chi}(h)$. Because the phase of any given data vector u is arbitrary, the phase information is not needed for this step. The coordinates $h(t)$ and the phase $\phi(t)$ completely describe the state of the system so the dimension of the attractor in the unreduced state space will be $d_{\mathcal{M}} + 1$. (For further discussion of invariant manifolds in translation-symmetric systems, see Ref. [31].) Indeed, phase alignment allows more efficient representation of the data, because phase information need not be encoded—it is captured separately as noted above.

To find χ and $\check{\chi}$, we use a variant of a standard undercomplete autoencoder, shown as the "$\tau = 0$" branch of architecture shown in Fig. 1. This variant uses an NN to represent the *difference* between the data and their projection onto the basis arising from principal components analysis (PCA) of

the data set [33]. PCA is widely used for linear dimension reduction because it yields the projection of dimension d_h that minimizes the mean-squared deviation from the original data. Let U be a square orthogonal matrix whose columns are the PCA basis vectors, and $P_{d_h}U^T$ and $P_{d-d_h}U^T$ the projection onto the first d_h and last $d-d_h$ such vectors, respectively. The encoding step learns the function $E(U^T\hat{u}(t))$ such that

$$E(U^T\hat{u}(t)) = h(t) - P_{d_h}U^T\hat{u}(t). \tag{2}$$

This structure is shown inside the blue (upper) box in Fig. 1. It must be emphasized that there is no approximation in choosing this representation. Furthermore, U need not come from PCA; for example, U could be the discrete Fourier transform operator or simply the identity, the latter corresponding to using the solution values on the mesh points.

The decoding step takes the data $h(t) \in \mathbb{R}^{d_h}$ in the inertial manifold coordinates and transforms them back to the full space, as shown in the yellow (lower) box in Fig. 1. Again one can think of learning a difference: The decoder learns a function $D(h(t))$ such that

$$D(h(t)) = U^T\tilde{u}(t) - \begin{bmatrix} h(t) \\ 0 \end{bmatrix}. \tag{3}$$

Taking $E(U^T\hat{u}(t)) = 0$ recovers the original IM formulation but precludes the representation of curved manifolds that do not have a one-to-one mapping from a linear projection. An example of such a manifold is the Archimedean spiral, whose Cartesian representation is $(x, y) = (\phi\cos\phi, \phi\sin\phi)$. The autoencoder architecture used here is able to represent this manifold with $d_h = 1$.

Finally, inserting Eq. (2) into Eq. (3), solving for $\tilde{u}(t)$, and noting that this can be written $\tilde{u}(t) = U[P_{d_h}U^T\hat{u}(t), P_{d-d_h}U^T\hat{u}(t)]^T$, shows that the exact solution satisfies $E(U^T\hat{u}(t)) + D_{d_h}((h(t)) = 0$, where D_{d_h} contains the first d_h components of D. This constraint can be satisfied approximately by adding a penalty term to the autoencoder loss function so it becomes

$$L = ||\hat{u}(t) - \tilde{u}(t)||^2 + \alpha||E(\hat{u}(t)) + D_{d_h}(h(t))||^2. \tag{4}$$

With this structure, we can in principle achieve an exact representation (within the approximation error of the functions E and D) of data on a manifold of dimension $d_{\mathcal{M}}$ for all $d_h \geqslant d_{\mathcal{M}}$. In general, the functions E and D, or more generally χ and $\check{\chi}$, need not come from NNs. Other approaches to nonlinear dimension reduction and function approximation (e.g., tSNE, diffusion maps, kernel regression [34,35]) might be useful as well. The overall structure of our approach would be the same.

Autoencoders of the above structure, which we denote hybrid neural networks (HNN) were trained (i.e., the functions E and D were determined) using the phase-aligned data. At a given value of d_h, 20 HNNs (each initialized with different initial guesses for the weights), with $\alpha = 1$ were trained for 1000 epochs with an Adam optimizer using Keras [36]. This process was repeated for a range of d_h. Results are reported for the model with the lowest MSE at each value of d_h.

For comparison we trained three variations on the HNN to evaluate the effect of the linear projection (P_{d_h}), the PCA change of basis (U^T), and phase alignment steps. In the first variation, denoted PCANN, we built a NN without the "Trunc" and "Expand" blocks in Fig. 1, which corresponds to using the PCA basis, but using E and D to learn the whole nonlinear coordinate tranformation rather than just the difference from PCA. The next variation builds upon the previous and removes the "PCA" block in Fig. 1, which leaves it in the original basis, so we denote it ONN. Then, the last variation is to remove the phase shift (No Shift). Both the PCANN and the ONN are trained with the loss $L = ||\hat{u} - \tilde{u}||^2$, while the unsifted variation is trained with $L = ||u - \tilde{u}||^2$. Hyperparameter tuning of the NN architectures was performed manually by varying width, depth, and activation functions. All of these variations used the same architecture, shown in Table I, for functions E and D.

TABLE I. Architectures of the NNs. "Shape" indicates the dimension of each layer, and "activation" the corresponding activation functions (S is the sigmoid activation) [7]. Decoder1 refers to domains $L = 22$ and 44. Decoder2 refers to $L = 66$.

	Function	Shape	Activation
Encoder	E	$d : 500 : d_h$	S:tanh
Decoder1	D	$d_h : 500 : d$	S:linear
Decoder2	D	$d_h : 500 : 500 : d$	S:linear
Evolution	F_h	$d_h : 200 : 200 : d_h$	S:S:linear
Evolution	F_ϕ	$d_h : 500 : 50 : 500 : 1$	S:S:S:linear

Figure 2 shows the mean-squared error (MSE) on a separate test data set for the NN methods described above and PCA for $L = 22$. At low d_h, the HNN, PCANN, and ONN all perform similarly, and in all three cases the MSE drops significantly at $d_h = 7$. For the case of no shifting, the drop appears at $d_h = 8$ because the continuous translation symmetry has not been factored out. All NNs perform orders of magnitude better than PCA. On continuing to increase d_h, the MSE for the HNN continues to improve while the others stagnate, because the HNN only needs correct coefficients of the less relevant higher PCA modes, while the other methods modify all of them. Notably, the abrupt drop in MSE at $d_h = 7$ coincides with the true dimension $d_{\mathcal{M}}$ of the attractor as found in Ref. [27]. The remaining error for $d_h \geqslant 7$ for the HNN is small, at $O(10^{-7})$, which follows from the fact that at this

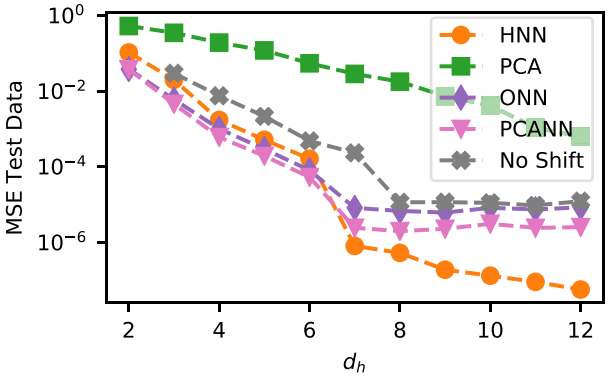

FIG. 2. MSE of test data for various d_h for $L = 22$. The legend is described in the text.

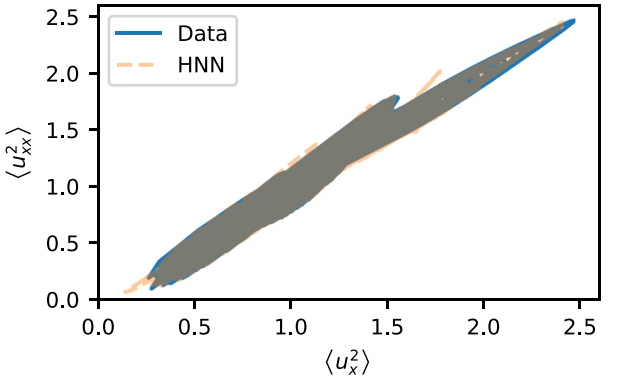

FIG. 3. $\mathcal{P}$ vs ε state-space projection for the data at $L = 22$, and HNN ROM prediction with $d_h = 7$.

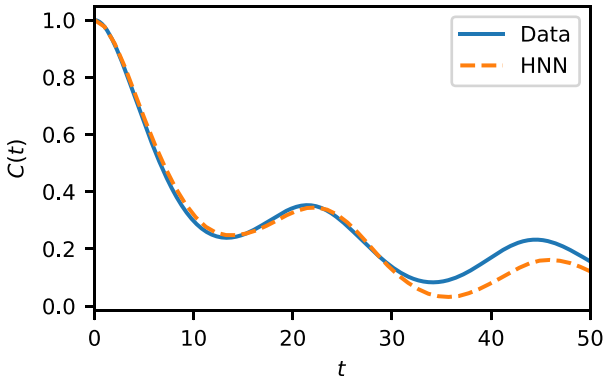

FIG. 4. Time-correlation function for the data at $L = 22$, and HNN ROM prediction with $d_h = 7$.

dimension an exact coordinate transformation exists, so the remaining error is approximation error.

Having in hand the coordinate representation for points on $\mathcal{M}$, we now use NNs to learn the dynamical system ("exact" reduced-order model) on the manifold, corresponding to the "$\tau \neq 0$" branch in Fig. 1. This approach will be denoted "HNN ROM." We construct discrete time mappings

$$h(t+\hat{\tau}) = F_h(h(t)), \quad \Delta\phi \equiv \phi(t+\hat{\tau}) - \phi(t) = F_\phi(h(t)), \tag{5}$$

where F_h and F_ϕ have the architectures shown in Table I. We use the symbol $\hat{\tau}$ instead of τ to emphasize that the discrete time mappings use in-slice time. We chose $\hat{\tau} = 2$, which reproduces trajectories well, by allowing for the signal to change an appreciable amount, but not too much, in one time interval. Setting $\hat{\tau}$ much smaller or larger results in poor model predictions.

Recall that the energy balance for the KSE requires that the production and dissipation rates $\mathcal{P}$ and ε must balance on average. We incorporate this fact in the training of the dynamic models as follows. We compute the projection of the data onto $\mathcal{P}$ and ε, as shown in Fig. 3. The relation between $\mathcal{P}$ and ε is narrowly distributed around the line $\mathcal{P} = \varepsilon$, with a sharp boundary, and we can find maximum and minimum dissipation rates $\varepsilon_{\max}$ and $\varepsilon_{\min}$ associated with a given value of $\mathcal{P}$. We then add a penalty for crossing this boundary to the loss function L_F for F_h and F_ϕ, as follows:

$$L_F = ||u(t+\hat{\tau}) - \tilde{u}(t+\hat{\tau})||^2 + \beta \max(\max(0, \tilde{\varepsilon} - \varepsilon_{\max}(\tilde{\mathcal{P}})), \varepsilon_{\min}(\tilde{\mathcal{P}}) - \tilde{\varepsilon}), \tag{6}$$

where $\tilde{\mathcal{P}}$ and $\tilde{\varepsilon}$ are calculated from $\tilde{u}$. We selected $\beta = 0.1$ so the second term contributed the same order of error to the loss as the first term. For each d_h, the best dimension reduction model was chosen, and 50 time-evolution models were trained for 200 epochs. Results are reported for the best models, as determined at a given d_h based on producing low errors in both short and long-time statistics.

To illustrate the performance of this approach, which we denote HNN ROM, on predicting dynamics, we first present short-time tracking results and then long-time statistics. All trajectories are evolved from a given initial condition $u(0)$ on the manifold, from which we find $\hat{u}(0)$ and $\phi(0)$ by phase alignment and then set $h(0) = \chi(\hat{u}(0))$. This initial condition in the manifold coordinates is evolved forward in in-slice time with Eq. (5). For validating the performance of the short-time tracking we need a timescale for comparison. Here we consider the integral timescale $T_I = \int_0^\infty C(t)\, dt \approx 19$, where

$$C(t) = \frac{\langle u(0)u(t)\rangle}{\langle u(0)^2\rangle}$$

is the temporal autocorrelation, and the Lyapunov time $T_L \approx 21$ [27]. Figure 4 shows the temporal autocorrelation of the data and the HNN ROM at $d_h = d_{\mathcal{M}}$ are in good agreement for $t \lesssim 30$. Likewise, typical trajectories show close tracking

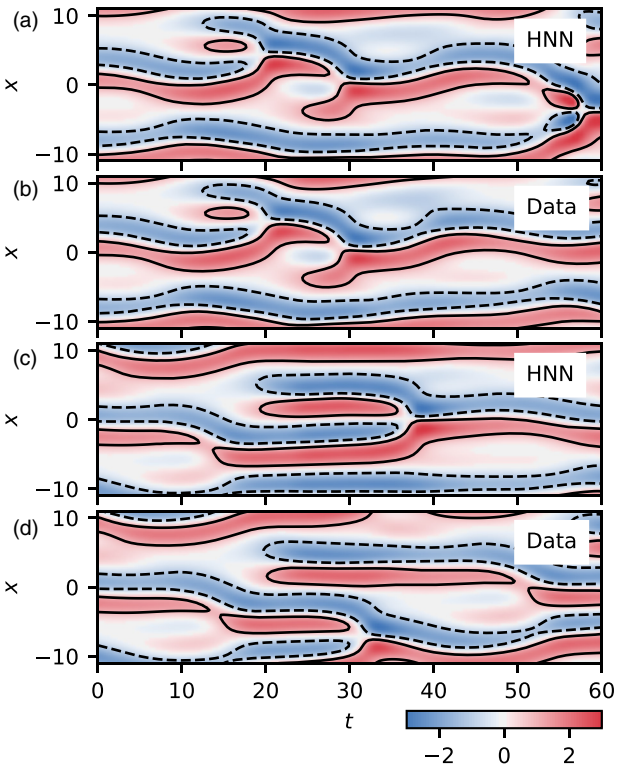

FIG. 5. [(a) and (c)] Trajectories [color contours of $u(x,t)$ with solid lines at $u = 1$ and dashed lines at $u = -1$] evolving from different initial conditions according to the HNN ROM; $L = 22$. [(b) and (d)] True trajectories corresponding to (a) and (c).

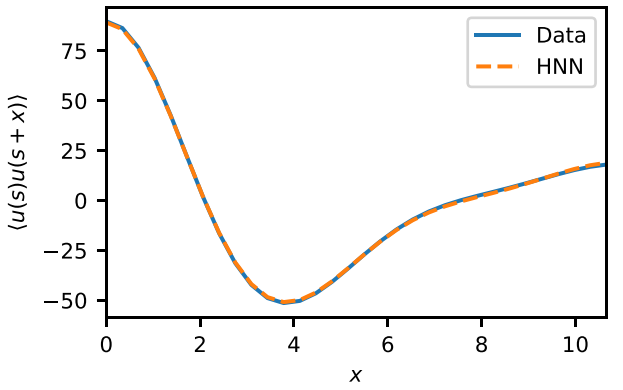

FIG. 6. Spatial correlation function for the data and HNN ROM prediction with $d_h = 7$, $L = 22$.

for 30 or more real time units. This comparison appears in Fig. 5, where Figs. 5(a) and 5(c) show the evolution of two initial conditions of test data using the dynamical system found with $d_h = d_{\mathcal{M}} = 7$, and the "exact" results are shown in Figs. 5(b) and 5(d) obtained from solving the KSE. These results indicate predictive capability for timescales longer than T_I or T_L, and thus represent very good performance for prediction of chaotic dynamics.

Next, we evaluate the ability of our dynamic model to reproduce key long-time statistical properties of the attractor, focusing on the quantities u_x and u_{xx} that determine the energy production and dissipation in the KSE. We examine predictions both for $d_h = d_{\mathcal{M}}$ and for values of d_h either larger or smaller than $d_{\mathcal{M}}$. The trajectories considered here cover approximately 9×10^3 real time units.

Figures 6 and 3, respectively, show the spatial autocorrelation function (averaged over space and time) and the energy balance ($\mathcal{P}$ vs ε) of the HNN ROM for $d_h = d_{\mathcal{M}}$ and for data, illustrating close agreement of these quantities. These statistics show that long-time trajectories do not diverge from the attractor and that the HNN ROM prediction stays within the envelope of the energy balance, which was the intent of the penalty in the loss for the time-evolution training, Eq. (6). These predictions deteriorate when $d_h < d_{\mathcal{M}}$.

A more detailed representation of the attractor is the joint probability density function (PDF) of the pointwise values of u_x and u_{xx}. Figure 7(a) shows this PDF, on a log scale, as determined from the data. At $d_h = d_{\mathcal{M}} = 7$, the HNN ROM prediction, Fig. 7(b), is very close to the exact PDF. To highlight the effect of the nonlinear autoencoder, we also consider predictions with linear dimension reduction from PCA (i.e., $E = D = 0$), and NNs for the dynamics; we denote this approach PCA ROM. At the same dimension, Fig. 7(d) shows that the PCA ROM prediction yields much poorer results. Figure 7(c) shows how the relative L_2 difference between the true PDF and the model predictions varies with d_h. For $d_h = 5$, the predictions are poor for both cases, but for $d_h \geqslant 7$, the error is

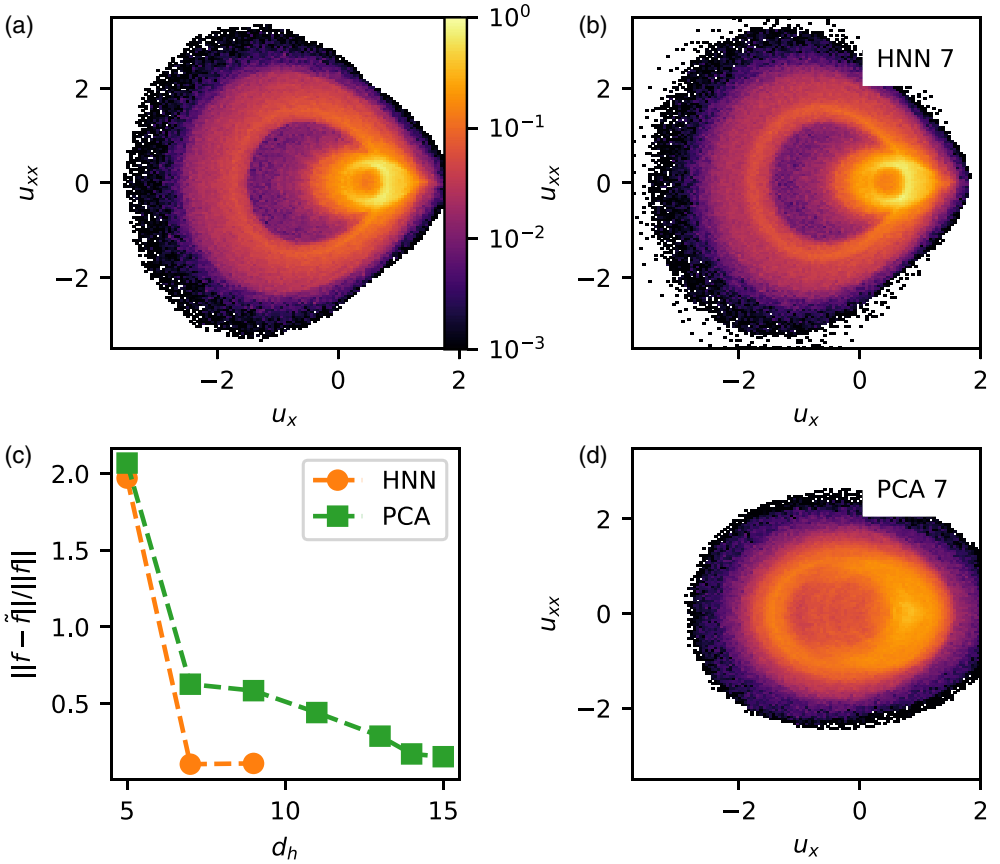

FIG. 7. (a) Joint PDF of data, $L = 22$, and (b) joint PDF of HNN ROM prediction, both plotted on a logarithmic scale. (c) Relative error in PDF vs dimension. The PDF from data is denoted f, and that from the model prediction is $\tilde{f}$.

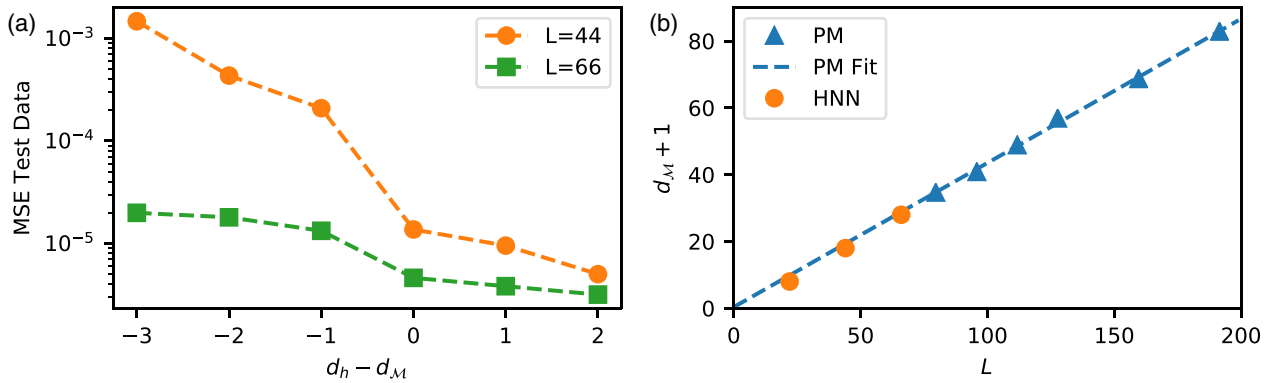

FIG. 8. (a) MSE vs the difference between d_h and the predicted $d_{\mathcal{M}}$. (b) Dimension $d_{\mathcal{M}} + 1$ of the unreduced IM for different domain sizes: "HNN" denotes the current results, "PM" the number of physical modes from Ref. [26], and "PM Fit" a linear fit of the PM curve.

small and nearly unchanging for the HNN ROM case, which is unsurprising since $d_{\mathcal{M}} = 7$. On the other hand, it takes $d_h = 14$ for the PCA ROM to yield a comparable model to the HNN ROM at $d_h = 7$. This result might be expected based on Whitney's embedding theorem, which states that any manifold of dimension k can be embedded in $\mathbb{R}^{2k}$ [37].

To investigate the generality of this method, we examine its performance for larger domains, $L = 44$ and $L = 66$. For $L = 22$ there is one positive Lyapunov exponent [27,38], whereas the dynamics at $L = 44$ and $L = 66$ are more chaotic; at these values, Edson *et al.* [38] report four and seven positive Lyapunov exponents, respectively. Data gathering and NN training were performed in the same way as for $L = 22$. Initial trials for the $L = 66$ showed poor results, so the capacity of the decoder was increased, as noted in Table I. Figure 8(a) shows the MSE on a test data set of HNNs with the lowest MSE at various d_h for $L = 44$ and 66. The horizontal axis is centered around the $d_{\mathcal{M}}$ of each domain size inferred from the drop in the MSE. This corresponds to $d_{\mathcal{M}} = 17$ and $d_{\mathcal{M}} = 27$ for $L = 44$ and $L = 66$. Increasing dimension still shows a distinct drop in MSE; however, it becomes less substantial with increased domain size. The $L = 66$ autoencoder has lower MSE than $L = 44$ because of the increased capacity.

The results at these domain sizes suggests that there is a linear scaling of $d_{\mathcal{M}}$ with L. This observation agrees well with

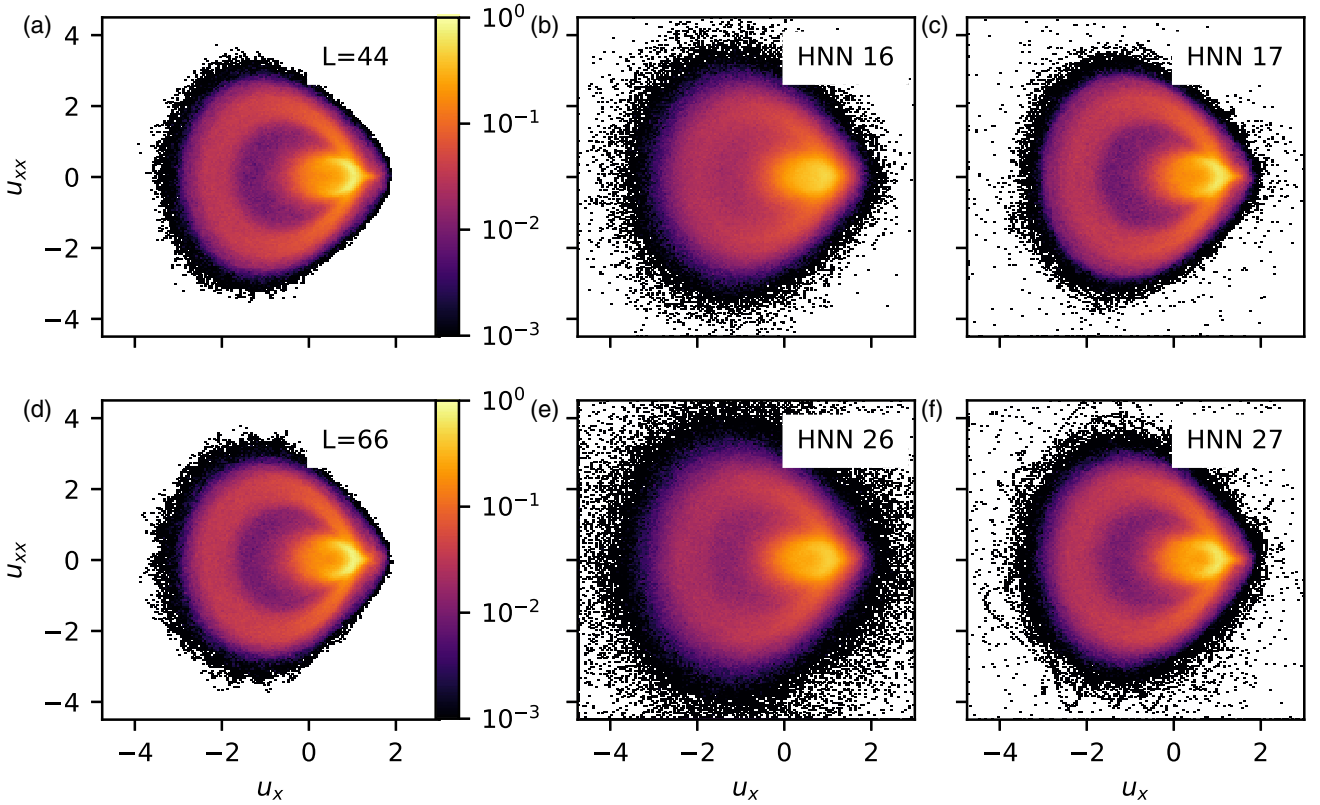

FIG. 9. [(a) and (d)] Joint PDF of data for $L = 44, 66$. [(b), (c), (e), and (f)] Joint PDF of HNN ROM prediction at $d_{\mathcal{M}} - 1$ and $d_{\mathcal{M}}$ for $L = 44, 66$.

results of Yang *et al.* [26] where they show the number of physical modes (PM) scales linearly with domain size.

Figure 8(b) shows $d_{\mathcal{M}} + 1$ (the dimension in the unreduced state space) against the domain size for both our results and their results, along with the extrapolation of their results to smaller domains. The excellent agreement between these results provides additional computational evidence that the dimension of the IM for the KSE scales linearly with L. Finally, we show the attractor recreation with the joint PDFs of u_x and u_{xx}. Figure 9(a) and 9(d) show these for $L = 44$ and $L = 66$ for trajectories of approximately 8×10^3 and 6×10^3 real time units, respectively. Model predictions for $d_h = d_{\mathcal{M}} - 1$, Figs. 9(b) and 9(e), and $d_{\mathcal{M}}$ degrees of freedom, Figs. 9(c) and 9(f), are shown for comparison. Unlike with $L = 22$, where models with too few dimensions tended to land on periodic orbits, here, with more degrees of freedom, models with too few dimensions maintain chaos. However, the PDFs for $d_{\mathcal{M}} - 1$ are more diffuse, less accurately recreating the data. In both cases, when the models retain $d_{\mathcal{M}}$ dimensions the joint PDF agrees well with the data, the main discrepancy being a broader tail of the model PDF, i.e., a higher, but still very low, probability of large excursions.

IV. CONCLUSION

We have shown here a framework for data-driven "exact" reduction of a dynamical system onto a low dimensional invariant manifold and time evolution on that manifold. Translation symmetry and energy conservation, two important features of many systems of interest, are incorporated naturally into the framework. By observing the model reduction error as a function of dimension, the dimension $d_{\mathcal{M}}$ of the invariant manifold can be determined, and once $d_{\mathcal{M}}$ is known, highly accurate model predictions can be obtained. In particular, key statistical quantities in a chaotic system can be well approximated, indicating that the model dynamics capture the shape of the attractor. In this work, the NNs can be trained on a single processor in a couple days, and time evolution over 10^4 time units takes only minutes. At present, it is difficult to predict how network size and training time will scale for more complex problems, especially given that it is not even known in general how $d_{\mathcal{M}}$ scales. Extensions to systems with higher-dimensional dynamics are underway. Systematizing this method could provide a straightforward, data-driven means of approximating the dimension of manifolds and constructing reduced order models, a difficult task for high-dimensional chaotic systems like turbulence. Code used in this work is available at [39].

ACKNOWLEDGMENTS

This work was supported by Air Force Office of Scientific Research (AFOSR) FA9550-18-1-0174 and Office of Naval Research (ONR) N00014-18-1-2865 (Vannevar Bush Faculty Fellowship).

[1] E. Hopf, Commun. Pure Appl. Math. **1**, 303 (1948).
[2] R. Temam and X. Wang, Dierential and Integral Equations **7**, 1095 (1994).
[3] S. Zelik, Proc. Roy. Soc. Edinb. A **144**, 1245 (2013).
[4] C. Foias, G. R. Sell, and R. Temam, J. Diff. Equ. **73**, 309 (1988).
[5] M. D. Graham, P. H. Steen, and E. S. Titi, J. Nonlin. Sci. **3**, 153 (1993).
[6] G. E. Hinton and R. R. Salakhutdinov, Science **313**, 504 (2006).
[7] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning* (MIT Press, Cambridge, MA, 2016).
[8] N. Omata and S. Shirayama, AIP Advances **9**, 015006 (2019).
[9] M. Milano and P. Koumoutsakos, J. Comput. Phys. **182**, 1 (2002).
[10] S. E. Otto and C. W. Rowley, SIAM J. Appl. Dynam. Syst. **18**, 558 (2019).
[11] B. Lusch, J. N. Kutz, and S. L. Brunton, Nat. Commun. **9**, 4950 (2018).
[12] K. Champion, B. Lusch, J. N. Kutz, and S. L. Brunton, Proc. Natl. Acad. Sci. **116**, 22445 (2019).
[13] F. J. Gonzalez and M. Balajewicz, arXiv:1808.01346v2.
[14] K. Lee and K. T. Carlberg, J. Comput. Phys. **404**, 108973 (2020).
[15] R. Iten, T. Metger, H. Wilming, L. del Rio, and R. Renner, Phys. Rev. Lett. **124**, 010508 (2020).
[16] R. González-García, R. Rico-Martínez, and I. G. Kevrekidis, Comput. Chem. Eng. **22**, S965 (1998).
[17] J. Pathak, B. Hunt, M. Girvan, Z. Lu, and E. Ott, Phys. Rev. Lett. **120**, 024102 (2018).
[18] H. F. Lui and W. R. Wolf, J. Fluid Mech. **872**, 963 (2019).
[19] P. R. Vlachas, W. Byeon, Z. Y. Wan, T. P. Sapsis, and P. Koumoutsakos, Proc. Roy. Soc. A: Math. Phys. Eng. Sci. **474**, 20170844 (2018).
[20] Z. Y. Wan, P. Vlachas, P. Koumoutsakos, and T. Sapsis, PLoS ONE **13**, e0197704 (2018).
[21] E. S. Titi, J. Math. Anal. Appl. **149**, 540 (1990).
[22] C. Foias, M. S. Jolly, I. G. Kevrekidis, G. R. Sell, and E. S. Titi, Phys. Lett. A **131**, 433 (1988).
[23] M. S. Jolly, I. G. Kevrekidis, and E. S. Titi, Physica D **44**, 38 (1990).
[24] H. G. Matthies and M. Meyer, Comput. Struct. **81**, 1277 (2003).
[25] P. V. Kuptsov and A. V. Kuptsova, in *Saratov Fall Meeting 2018: Computations and Data Analysis: From Nanoscale Tools to Brain Functions*, edited by D. E. Postnov (SPIE, 2019).
[26] H. L. Yang, K. A. Takeuchi, F. Ginelli, H. Chaté, and G. Radons, Phys. Rev. Lett. **102**, 074102 (2009).
[27] X. Ding, H. Chaté, P. Cvitanović, E. Siminos, and K. A. Takeuchi, Phys. Rev. Lett. **117**, 024101 (2016).
[28] A. K. Kassam and L. N. Trefethen, SIAM J. Sci. Comput. **26**, 1214 (2005).
[29] P. Cvitanović, R. Artuso, R. Mainieri, G. Tanner, and G. Vattay, *Chaos: Classical and Quantum* (Niels Bohr Institute, Copenhagen, 2016).
[30] N. B. Budanur, D. Borrero-Echeverry, and P. Cvitanović, Chaos **25**, 073112 (2015).

[31] N. B. Budanur, P. Cvitanović, R. L. Davidchack, and E. Siminos, Phys. Rev. Lett. **114**, 084102 (2015).
[32] A. P. Willis, P. Cvitanović, and M. Avila, J. Fluid Mech. **721**, 514 (2013).
[33] G. Strang, *Linear Algebra and Learning from Data* (Wellesley-Cambridge Press, London, 2019).
[34] L. van der Maaten and G. Hinton, J. Mach. Learn. Res. **9**, 2579 (2008).
[35] L. van der Maaten, E. Postma, and J. Van den Herik, J. Mach. Learn. Res. **10**, 66 (2009).
[36] F. Chollet *et al.*, Keras, 2015, https://keras.io.
[37] V. Guillemin and A. Pollack, *Differential Topology* (AMS Chelsea Publishing, Providence, RI, 2000).
[38] R. A. Edson, J. E. Bunder, T. W. Mattner, and A. J. Roberts, ANZIAM J. **61**, 270 (2019).
[39] A. J. Linot and M. D. Graham, KSNN, https://github.com/alinot5/KSNN, 2020.