Machine Learning the Tangent Space of Dynamical Instabilities from Data

Antoine Blanchard\textsuperscript{a)}\textsuperscript{a)} and Themistoklis P. Sapsis\textsuperscript{b)\textsuperscript{b)}}

Department of Mechanical Engineering, Massachusetts Institute of Technology, Cambridge, MA 02139

For a large class of dynamical systems, the optimally time-dependent (OTD) modes, a set of deformable orthonormal tangent vectors that track directions of instabilities along any trajectory, are known to depend “pointwise” on the state of the system on the attractor, and not on the history of the trajectory. We leverage the power of neural networks to learn this “pointwise” mapping from phase space to OTD space directly from data. The result of the learning process is a cartography of directions associated with strongest instabilities in phase space. Implications for data-driven prediction and control of dynamical instabilities are discussed.

The optimally time-dependent (OTD) modes are a set of deformable orthonormal tangent vectors that track directions of instabilities along any trajectory of a dynamical system. Traditionally, these modes are computed by a time-marching approach that involves solving multiple initial-boundary-value problems concurrently with the state equations. However, for a large class of dynamical systems, the OTD modes are known to depend “pointwise” on the state of the system on the attractor, and not on the history of the trajectory. So we propose a neural-network algorithm to learn this “pointwise” mapping from phase space to OTD space directly from data. The neural network produces a cartography of directions of strongest instability in phase space, as well as accurate estimates for the leading Lyapunov exponents.

I. INTRODUCTION

The theory of dynamical systems has a long history of trying to elucidate one of the most important concepts in science and engineering: instability. Dynamical instabilities are important because they can give rise to a variety of phenomena with unexpected, and even disastrous, consequences. They occur in fluid mechanics\textsuperscript{1,2}, climate dynamics\textsuperscript{3}, optics\textsuperscript{4}, and thermoacoustics\textsuperscript{5}, and come in many shapes and forms. Perhaps the simplest of all are instabilities arising from a linear mechanism, whose investigation traditionally involves linearizing the governing equations around a fixed point, and looking for unstable eigenvalues of the linearized problem\textsuperscript{6}. Generalization of this concept to periodic orbits led to the well-known Floquet theory, in which stability of periodic trajectories is ascertained by computing the spectrum of the monodromy matrix\textsuperscript{5}. The realization that episodes of transient growth cannot be predicted by linear or Floquet analyses came much later, and in turn gave rise to the theory of non-normal and transient instabilities\textsuperscript{7}. This culminated with the introduction of the Lyapunov exponents and Lyapunov vectors, which provide a rigorous description of instability mechanisms in chaotic systems\textsuperscript{8,9}. Since then, considerable effort has been devoted to the development of algorithms for computation of Lyapunov exponents and Lyapunov vectors from data. Great strides have been made by the likes of Eckmann et al.\textsuperscript{10}, Sano & Sawada\textsuperscript{11}, Rosenstein et al.\textsuperscript{12}, Kantz\textsuperscript{13}, and Wolfe et al.\textsuperscript{14}, who proposed new methods to compute Lyapunov exponents and Lyapunov vectors from experimental measurements, or improved on existing ones.

The key feature shared by these algorithms is that they monitor the fate of perturbations around a reference orbit for sufficiently long times, sometimes resorting to orthonormalization in order to prevent blow-up of the perturbation vectors. But tracking the evolution of perturbations along a trajectory is nothing more than a time-marching approach in disguise. More importantly, it does not utilize the fact that for a large class of dynamical systems, the \( i \)th Lyapunov vector \( u_i \) depends pointwise on the state of the system (with the mapping \( x \mapsto u_i(x) \) being uniquely determined by the phase-space point \( x \)), and consequently the \( i \)th Lyapunov exponent—a quadratic form over \( u_i(x) \)—can be recovered by measure-averaging over many measurement points, rather than time-averaging over a long trajectory. This was pointed out by Ershov and Potapov\textsuperscript{15}, who recognized the benefits of using measure-averaging in lieu of time-averaging, as the former allows for the measurement points to be arranged in any arbitrary order, and even for them to belong to different trajectories. These results show that if one were able to compute the pointwise function \( u_i(x) \) from data, then one would immediately have access to a complete picture of the directions of instabilities at any point in the phase space, as well as accurate estimates for the Lyapunov exponents.

As far as we know, the only attempt to compute the map \( u_i(x) \) was made in Ref. 16, where we used the theory of slow invariant manifolds by Fenichel\textsuperscript{17} to derive analytical expressions for \( u_i(x) \) in situations exhibiting slow-fast dynamics. However, application of the method is limited because a) the dimensionality of the system cannot be too large for the manifold analysis to be tractable; and b) the system must have a well-defined separation of time scales, short of which Fenichel’s theory becomes moot. Of course, the method being analytical, there is no data component to it. This is precisely what we set out to rectify in the present paper, with the introduction of a machine-learning algorithm that infers the map \( u_i(x) \) from a large collection of state measurements.

Machine-learning (ML) algorithms have permeated virtually every area of science and engineering because of two reasons. First, the amount of data available from experiments,
field measurements, or numerical simulations has reached unprecedented levels. Second, ML algorithms have proven to be extremely versatile and powerful from the standpoint of extracting patterns and information from tremendously complex systems that would otherwise remain unaccessible. Applications of ML to dynamical systems include dimensionality reduction and flow-feature extraction\(^{18-21}\), discovery of governing equations\(^{22-24}\), design of optimal control strategies\(^{25}\), turbulence closure modeling\(^{26}\), integration of partial differential equations\(^{27-29}\), and more\(^{30}\). However, the literature on applications of ML to dynamical instabilities is remarkably scarce. The only related investigation of which we are aware was conducted by Wan et al.\(^{31}\), who used long-short-term-memory neural networks to predict occurrences of extreme events in chaotic systems.

The learning algorithm that we propose in this paper produces a cartography of directions associated with strongest instabilities in phase space, from which the leading Lyapunov exponents can be extracted. These directions coincide with the backward Lyapunov vectors of Legras & Vautard\(^9\) and the optimally-time-dependent (OTD) modes of Babaee & Sapsis\(^{32}\). (The equivalence between the two was established in Ref. 16.) The potential of the learning algorithm in problems related to prediction and control of transient instabilities and extreme events is considerable, because the proposed method is fully data-driven, has no restricting assumptions other than invertibility, autonomy, ergodicity, and measure-invariance of the underlying dynamical system, and only requires state measurements as inputs.

The paper is organized as follows. We formulate the problem in §II, introduce the learning algorithm in §III, followed by results in §IV, a discussion in §V, and a conclusion in §VI.

II. FORMULATION OF THE PROBLEM

A. Preliminaries

We consider the autonomous dynamical system

\[ \dot{x} = F(x), \quad x(t_0) = x_0, \]

where \( x \) belongs to a compact Riemannian manifold \( \mathcal{X} \) endowed with the Borel \( \sigma \)-algebra and a measure \( \mu \), the map \( F: \mathcal{X} \rightarrow \mathcal{X} \) is sufficiently smooth to ensure existence and uniqueness of solutions, and overdot denotes differentiation with respect to the time variable \( t \). We assume that the transformation

\[ S_t : \mathcal{X} \rightarrow \mathcal{X}, \quad x_0 \mapsto x(t) = S_t(x_0), \]

sometimes referred to as the “flow map”, is invertible, measure-preserving, and ergodic. Measure-invariance is important from the standpoint of defining time-averages of scalar functions. (This is the well-known Birkhoff ergodic theorem\(^{33}\).) Measure-invariance and ergodicity are important to guarantee that time-averages and measure-averages coincide:

\[ \lim_{t \to \infty} \frac{1}{T} \int_0^T f(S_t(x_0)) dt = \int_{\mathcal{X}} f(x) d\mu(x), \quad (3) \]

for all \( f \in L^2_\mu(\mathcal{X}) \). In words, these assumptions ensure that trajectories asymptotically settle into an attractor \( A \subset \mathcal{X} \) (which may be steady, time-periodic, quasiperiodic, or chaotic), and remain on that attractor (i.e., there is no “switching” between attractors).

Infinitesimal perturbations around a given trajectory obey the variational equation

\[ \dot{v} = L(x; v), \quad v(t_0) = v_0, \quad (4) \]

where \( v \) belongs to the tangent space of the manifold \( \mathcal{X} \) at point \( x \), denoted by \( T_x \mathcal{X} \), and \( L(x; v) \triangleq dF(x; v) \) is the Gâteaux derivative of \( F \) evaluated at \( x \) along the direction \( v \). In principle, the variational equation could be used to track directions of instabilities around trajectories. In practice, however, this is impossible, because any collection of perturbations \( \{v_i\}_{i=1}^r \) evolved with (4) for a sufficiently long time would see the magnitude of its individual members grow or decay exponentially fast, and the angle between them rapidly vanish.

To compute a set of meaningful directions (or “modes”) from the variational equation, Babaee & Sapsis\(^{32}\) proposed to enforce orthonormality of the \( v_i \)’s at all times. One way to achieve this is to continuously apply the Gram–Schmidt algorithm to the collection \( \{v_i\}_{i=1}^r \), starting with \( v_1 \) and moving down. Blanchard & Sapsis\(^{15}\) showed that the resulting vectors obey

\[ \dot{u}_i = L(x; u_i) - \langle u_i, L(x; u_1) \rangle u_1 - \sum_{k=1}^{i-1} \langle [u_i, L(x; u_k)] u_k \rangle u_k \quad (5) \]

for \( i \in \{1, \ldots, r\} \), where the angle brackets denote a suitable inner product on \( T_x \mathcal{X} \). In the above, we recognize the variational equation (the left-hand side and the first term on the right-hand side), appended with terms enforcing the orthonormality constraint (the last two terms on the right-hand side). We also note that the summation index goes to \( i-1 \) rather than \( r \), so that (5) assumes a lower-triangular form. (This reflects the very structure of the Gram–Schmidt algorithm.) The \( u_i \)’s have been referred to as the OTD modes, and the collection \( \{u_i\}_{i=1}^r \) as the OTD subspace\(^{32}\). The terms “subspace” and “modes” are appropriate because the collection \( \{u_i\}_{i=1}^r \) forms a real vector space, for which the vectors \( u_i \) are orthonormal basis.

A key property of the OTD modes is that they and the \( v_i \)’s span the same subspace. The first OTD mode aligns with the most unstable direction, just like \( v_1 \) does. The second OTD mode is not free to align with the second-most unstable direction, because it must remain orthogonal to \( u_1 \). But the subspace spanned by \( u_1 \) and \( u_2 \) coincides with the two-dimensional subspace that is most rapidly growing (this is also the subspace spanned by \( v_1 \) and \( v_2 \)). For hyperbolic fixed points, the OTD subspace aligns with the most unstable
eigenspace of the associated linearized operator. For time
dependent trajectories, the OTD subspace aligns with the left
eigenspace of the Cauchy–Green tensor, which characterizes transient instabilities. As a result, the \(i\)th Lyapunov exponent \(\lambda_i\) can be recovered from the \(i\)th OTD mode:

\[
\lambda_i = \lim_{T \to \infty} \frac{1}{T} \int_0^T \langle u_i(t), L(x(t); u_i(t))u_i(t) \rangle \, dt.
\] (6)

The fact that the OTD modes track directions of instabilities along any trajectory has been utilized on multiple occasions, including in the context of prediction of extreme events in chaotic systems, and design of low-dimensional controllers for stabilization of unsteady flows.

The OTD system (5) is a set of time-dependent differential equations which must be solved concurrently with the dynamical system (1). The standard approach for infinite-dimensional systems is to discretize (1) and (5) in space, and advance each using a time-stepping scheme. The dimension \(d\) of the phase space after discretization may be quite large, with the discretized state typically having thousands of millions of degrees of freedom. In such cases, computation of the first \(r\) OTD modes involves solving \(r\)-dimensional differential equations (the OTD equations), plus a \(d\)-dimensional differential equation for the state itself. For very large \(d\), this procedure is computationally onerous, and alternative approaches might be desirable.

B. The learning problem

For dynamical systems satisfying the assumptions made earlier (autonomy, invertibility, ergodicity, and measure-invariance), the OTD modes asymptotically converge to a set of vectors defined at every point on the attractor \(\mathcal{A}\) for the state. Each \(x_n\) must belong to the attractor, but not necessarily to the same trajectory, a consequence of the use of measure-averaging. Second, we assume that we have a mechanism to compute the vector field \(F(x_n)\) and the action of the linearized operator \(L\) at \(x_n\) in any direction \(v\). Third, we need to eliminate the explicit dependence of the OTD system (5) on time. This is done by applying the chain rule to the left-hand side of (5), resulting in

\[
\frac{d}{dt} u_i(x; F(x)) = L(x; u_i) - \langle u_i, L(x; u_i) \rangle u_i - \sum_{k=1}^{i-1} \left[ \langle u_i, L(x; u_k) \rangle + \langle u_k, L(x; u_i) \rangle \right] u_k,
\] (9)

where \(d/dt u_i(x; F(x))\) is the Gâteaux derivative of \(u_i\) evaluated at \(x\) along the direction \(F(x)\). There is no explicit temporal dependence in (9), so that the variable \(x\) should no longer be thought of as a point on a particular time-dependent trajectory, but rather as any point on the attractor. System (9) may also be thought of as a partial differential equation on \(\mathcal{X}\).

For computational purposes, it is useful to consider the discretized counterpart of (9):

\[
\nabla_x u_i F(x) = L(x) u_i - \langle u_i, L(x) u_i \rangle - \sum_{k=1}^{i-1} \left[ \langle u_i, L(x) u_k \rangle + \langle u_k, L(x) u_i \rangle \right] u_k,
\] (10)

where \(\nabla_x\) and \(u_i\) belong to \(\mathbb{R}^d\), \(L(x) \triangleq \nabla_x F(x)\) is the Jacobian of the vector field \(F: \mathbb{R}^d \to \mathbb{R}^d\), and \(\nabla_x u_i\) is the Jacobian of \(u_i\) with respect to \(x\). Although not explicitly shown in (10), the vector \(u_i\) should be understood as \(u_i(x)\). We are now in a position to state the learning problem:

Learning problem. Given a dataset \(\{x_n\}_{n=1}^N\) of snapshots belonging to the attractor \(\mathcal{A}\), and a mechanism to compute \(F(x_n)\) and the action of \(L(x_n)\), find the collection of graphs \(x \mapsto u_i(x), i \in \{1, \ldots, r\}\), that best satisfies (10) at every \(x_n\).

In what follows, we solve the learning problem by a deep-learning approach based on neural networks.

III. LEARNING THE OTD MODES FROM DATA

We will find it convenient to operate in the “big-data” regime, so we assume that the dataset used in the learning algorithm contains a very large number of snapshots. Before
we proceed, we reiterate the fundamental assumptions that are made about the data. The underlying dynamical system from which data is collected should be autonomous, invertible, measure-preserving and ergodic, and should have a non-quasidegenerate Lyapunov spectrum. As discussed in §II B, these assumptions are key to ensure existence, unicity, and continuity of the SLB in phase space.

A. Network architecture

To learn the graphs \( \{ u_i \}_{i=1} \) from the collection of snapshots, we employ a neural-network approach. This is appropriate, because each \( u_i \) is a continuous function of \( x \). This allows us to leverage the universal approximation theorem \(^{40}\), which states that any function may be approximated by a sufficiently large and deep neural network. We will refer to the learned OTD modes as the “deep OTD (dOTD) modes”.

1. Overview of the network architecture

We find it natural to assign to each OTD mode its own neural network \( u_i(x; \theta_i) \), where \( \theta_i \) denotes the parameters (weights and biases) of the \( i \)th network. We use the same fully-connected feed-forward architecture with \( L \) hidden layers for all OTD modes (figure 1a). (The input and output layers are numbered 0 and \( L + 1 \), respectively.) The loss function for the \( i \)th network is specified as

\[
\ell_{i}^{\text{pde}}(\theta_i) = \frac{1}{N} \sum_{n=1}^{N} \left\| \nabla_x u_i(x_n; \theta_i) F(x_n) - L(x_n) u_i(x_n; \theta_i) + \langle u_i(x_n; \theta_i), L(x_n) u_i(x_n; \theta_i) \rangle \right\|^2,
\]

which states that any function may be approximated by a sufficiently large and deep neural network. We will refer to the learned OTD modes as the “deep OTD (dOTD) modes”.

Equation (11) shows that the loss function for the \( i \)th dOTD mode depends on the first \( i - 1 \) dOTD modes. This raises the question of the order in which the dOTD modes ought to be learned. In what follows, we opt for a “sequential” approach (figure 1b), whereby the parameters \( \theta_i \) are optimized sequentially (starting with \( \theta_1 \) and moving down), and the outputs of the first \( i - 1 \) networks are fed into the \( i \)th network as dummy inputs. We find this architecture to be intuitive because it mimics the triangular structure of the Gram–Schmidt algorithm. Our numerical experiments suggest that this approach is more stable (compared to other approaches described below), in the sense that it facilitates convergence of the optimization algorithm. The only issue has to do with error accumulation, arising as a result of using the outcomes of the first \( i - 1 \) optimizations to compute the \( i \)th dOTD mode. However, this is easily fixed by tightening the tolerance of the optimization algorithm, or doing multiple passes of training with decreasing tolerance.

Of course, the “sequential” approach is not the only option. Instead, one could solve the \( r \) optimization problems concurrently, whereby the optimization algorithm performs one iteration for each neural network before updating the parameters.

FIG. 1. (a) Simplified sketch of the neural-network architecture for the \( i \)th dOTD mode; and (b) schematic of the sequential approach, where each unit labeled “NN\( i \)” is of the type shown in (a), and gray arrows denote that the outputs of the first \( i - 1 \) networks are passed to the \( i \)th network as dummy inputs.

Each iteration would need to be done sequentially (i.e., start-
ing with $\theta_1$, then $\theta_2$, etc), and “coupling” between the dOTD modes would be done by passing the first $i-1$ parameter vectors at the current iteration to the $i$th neural network. Alternately, one could combine the $r$ loss functions (11), and solve for all of the dOTD modes in a single neural network. This combined loss function. However, these two approaches appear to cause difficulty for the optimization algorithm, both in terms of execution speed and its ability to converge.

Use of the loss function (11) alone, which is solely based on the residual of the OTD system, might be problematic for two reasons. First, as discussed in §II B, any SLB is a global minimizer of (11), but only one of them is stable (this is the SLB to which the OTD modes converge when computed with a time-marching approach). So we need a mechanism to ensure that the optimization algorithm converges to the stable SLB, and not to any of the unstable ones. Second, there may be other (possibly local) minimizers of (11) besides SLBs, so we also need a mechanism that prevents the optimization routine from getting trapped in an irrelevant minimum.

2. Ensuring orthonormality of the dOTD modes

We begin with the question of how to ensure that the optimization algorithm converges to an SLB, with no consideration yet for whether that SLB is stable or not. We first note that minimization of the loss function (11) does not guarantee orthonormality of the resulting dOTD modes. (For example, the trivial solution $u = 0$ is a global minimizer of (11).) The reason is that the terms responsible for orthonormalizing the OTD modes in the time-dependent problem (5) are no longer sufficient to enforce this constraint. So orthonormality must be enforced explicitly in the neural network. This is important because our numerical experiments suggest that the SLBs are the only orthonormal minimizers of (11). In other words, explicitly enforcing orthonormality of the dOTD modes helps the optimization algorithm systematically converge to an SLB, rather than some other irrelevant minimum.

Ensuring orthonormality of the dOTD modes can be realized in two ways. One approach is to embed Gram–Schmidt orthonormalization immediately after the last layer of the network, so that the dOTD modes are orthonormal by construction. Another approach is to append to the loss function (11) a regularization term,

$$\ell_i^{\text{reg}}(\theta_i) = \frac{1}{N} \sum_{n=1}^{N} \left\{ \alpha \left( \langle u_i(x_n; \theta_i), u_i(x_n; \theta_i) \rangle - 1 \right)^2 + \sum_{k=1}^{i-1} \beta_k \langle u_i(x_n; \theta_i), u_k(x_n; \theta_i) \rangle \right\} .$$

The first term in the curly brackets enforces normality of $u_i$, and the second term enforces orthogonality of $u_i$ and $u_k$ ($k < i$). The regularization parameters $\alpha$ and $\beta_k$ should be chosen on a case-by-case basis, which offers less flexibility than the Gram–Schmidt approach. We note that the regularization approach and the Gram–Schmidt approach have no consequence for the estimation error, because their respective loss functions are exactly zero for any SLB.

In practice, we have found that the Gram–Schmidt approach is more robust than the regularization approach, in the sense that the former requires fewer iterations for the optimization algorithm to converge to an SLB. We note that each iteration in the Gram–Schmidt approach requires computing the gradients of the Gram–Schmidt layer with respect to the network parameters by back-propagation, which is significantly more expensive than computing the gradients of the regularizing terms in (12). However, this is a burden worth carrying, given that we have found cases in which the regularization approach failed to converge while the Gram–Schmidt approach succeeded, and no cases suggesting otherwise.

3. Ensuring uniqueness of the dOTD modes

Now that we have designed a mechanism ensuring that the optimization algorithm converges to an SLB, we must address the question of how to isolate the stable SLB from all of the unstable ones. We begin with a simple example that provides insight into this issue. Consider a case in which the trajectory of interest is a hyperbolic fixed point, denoted by $x_\ast$. Theorem 2.3 in Babaee & Sapsis23 states that any $r$-dimensional eigenspace of the operator $L(x_\ast)$ is an SLB of $x_\ast$, and that the only stable SLB is that associated with the $r$ most unstable eigenvalues of $L(x_\ast)$. In that work, “stability” was determined by examining the time-dependent problem governing the evolution of a perturbed SLB. In the learning problem, however, we have eliminated any notion of temporality from the OTD system. Hence, for the case of a hyperbolic fixed point, the neural network, in its current manifestation, may converge to any of the $d$-choose-$r$ eigenspaces of $L(x_\ast)$, and not necessarily to the most unstable one. This is problematic because the OTD modes draw their power from their ability to track directions of greatest instabilities. Naturally, we wish our learning algorithm to also have this feature.

To make sure that the learning algorithm returns the SLB associated with directions of strongest instabilities, we use a criterion based on Lyapunov exponents. As discussed in §II B, the $i$th Lyapunov exponent $\lambda_i$ can be computed as a measure-average of the Lagrange multiplier $\langle u_i(x), L(x)u_i(x) \rangle$. With a finite-size dataset, however, we can do no better than approximating $\lambda_i$ by a finite sum over the data points. If the dataset is composed of multiple long trajectories initialized on the attractor according to some probability distribution (in general, we want the initial conditions to be independent and identically distributed), we have that

$$\lim_{N \to \infty} \frac{1}{N} \sum_{n=1}^{N} \langle u_i(x_n), L(x_n)u_i(x_n) \rangle = \lambda_i,$$

where $x_n$ is the state of the $n$th trajectory after some long time $T$. The above limiting statement holds only when $T \to \infty$, but in practice we merely require that $T$ be large enough so as to ensure convergence of the distribution of initial conditions to an invariant one. Equation (13) also holds when the dataset is composed of uniformly-sampled snapshots collected in the asymptotic regime of a single long trajectory, in which case
of the state-of-the-art methods could be applied to the present problem, each introducing its own level of complexity. The key issue is that reconstruction of $F(x)$ can be done offline, regardless of the dimensionality of the system. In other words, computation of $F(x_n)$ for each $x_n$ may be viewed as a preprocessing step, and therefore does not add to the computational burden related to optimizing the parameters of the neural network. In what follows, we opt for perhaps the simplest of all approaches. We assume that the snapshots are sampled along a single long trajectory with a uniform and sufficiently small sampling time-step $\Delta s$, so that we may approximate $F(x_n)$ as a standard Euler-forward finite difference:

$$F(x_n) = \frac{x_{n+1} - x_n}{\Delta s} + O(\Delta s).$$

Higher-order finite-difference formulas (e.g., Adams–Bashforth, Adams–Moulton, or backward differentiation formulas) may be used if higher accuracy is desired. Finite differences have the advantage of being extremely cheap to compute, even for high-dimensional systems. Implementation is straightforward, and the requirement that $\Delta s$ be small is far from drastic.

To compute the action of the Jacobian matrix $L(x)$ from data, we employ the classical algorithm proposed independently by Eckmann et al.\textsuperscript{10} and Sano & Sawada\textsuperscript{11}. First, we scan the dataset to identify the $K$ nearest neighbors of each datapoint $x_n$. The nearest neighbors of $x_n$ are defined as those points $x_k$ of the orbit (past or future) that are contained in a ball of radius $\gamma$ centered at $x_n$:

$$||x_n - x_k|| \leq \gamma, \quad k \in \{1, \ldots, K\}. $$

If $\gamma$ is sufficiently small, then each vector $v_k^* = x_k - x_n$ may be viewed as a perturbation vector from the reference orbit. We therefore have

$$\frac{v_{n+1}^* - v_k^*}{\Delta s} = L(x_n)v_k^* + O(\Delta s),$$

which allows us to compute the action of the Jacobian matrix $L(x_n)$ on the vectors $\{v_k^*\}_{k=1}^{K}$. Now, the critical step is to note that the vectors $\{v_k^*\}_{k=1}^{K}$ belong to the tangent space at point $x_n$, and so do the OTD modes. (In fact, the OTD modes form a basis of that space when $r = d$.) So if we stack up the vectors $\{v_k\}_{k=1}^{K}$ into a matrix $V_n \in \mathbb{R}^{d \times K}$, then the least-square fit

$$u_i(x_n; \theta_i) \approx V_n^+ V_n u_i(x_n; \theta_i)$$

should be a reasonably good approximation for the dOTD modes. Here, $V_n^+$ is the pseudo-inverse of $V_n$. (We note that the least-square approach allows for $K$ exceeding the dimension of the phase space $d$.) From (19), we can compute the action of the linearized operator on the dOTD modes as

$$L(x_n)u_i(x_n; \theta_i) \approx \Delta V_n V_n^+ u_i(x_n; \theta_i),$$

where $\Delta V_n$ is a $d$ by $K$ matrix with columns $(v_{n+1}^* - v_k^*)/\Delta s$.

Equation (20) requires no information other than the snapshot data, and can be used to evaluate the loss function (11).
For low-dimensional systems, it is possible to form the matrix \( \Delta V \|_{x} \in \mathbb{R}^{d \times d} \) explicitly, and store it in memory. This computation can be done offline for every datapoint \( x_{n} \), and consequently the computational burden associated with reconstruction is nil for the neural network. For high-dimensional systems, however, forming and storing \( \Delta V \|_{x} \) is not possible, so (20) must be computed online (i.e., every time the loss function (11) is evaluated), which introduces an additional cost. Another aggravating issue is that the complexity of most nearest-neighbor-search algorithms grows exponentially with the dimension of the state, making the above approach intractable for \( d \) greater than about 25 (Ref. 42). The curse of dimensionality thus requires that we pursue a different strategy.

C. Learning in a high-dimensional phase space

For high-dimensional systems, the neural-network approach is intractable for two reasons. First, there is the issue of reconstructing the Jacobian matrix (or, equivalently, its action on the dOTD modes), which was discussed in §III B. Second, to evaluate the term \( \nabla_{\theta} U(x_{n}, \theta) \) appearing in (11), we must compute the gradient of the dOTD modes with respect to the state vector, resulting in a \( d \)-by-\( d \) matrix. For large \( d \), this computation is virtually hopeless. So to make the neural-network approach applicable to high-dimensional systems, we proceed to an order-reduction of the phase space.

If \( x \) arises from discretizing a partial differential equation (PDE) defined in a domain \( \Omega \), then one approach is to randomly select \( M \) points in that domain according to some probability distribution. Each snapshot \( x_{n} \) then has dimension \( M \), with \( M \) presumably much smaller than \( d \). When \( x \) contains nodal values of the state, this approach amounts to randomly excising \( M \) entries from each \( x_{n} \). (The excised entries need not be the same for all the snapshots.) Random sampling has been used successfully in a number of problems related to deep learning of PDEs \(^{28,43} \), largely because it has the advantage of being a “mesh-free” approach. However, applicability of this method to the present problem is limited, because the algorithm for attractor reconstruction requires that the sampled points be the same for all snapshots. As a result, we may need a large number of sampled points (with \( M \) possibly on the order of \( d \)) to faithfully capture the dynamics. If \( M \) is reasonably small, however, the dOTD modes can be learned at the sampled points, and subsequently reconstructed over the entire domain using any standard interpolation algorithm.

To avoid these difficulties, we use an approach based on Galerkin projection. We assume that any state on the attractor can be represented as a superposition of proper-orthogonal-decomposition (POD) modes:

\[
\mathbf{x} - \bar{x} = \Phi \xi, \tag{21}
\]

where \( \bar{x} \) is the mean flow, \( \Phi \in \mathbb{R}^{d \times n_{s}} \) contains the first \( n_{s} \) POD modes, and \( \xi \in \mathbb{R}^{n_{s}} \) contains the corresponding POD coefficients. Measure-invariance and ergodicity of the attractor allow use to view \( \xi \) as a function of time or as a function of the state, so that we may use \( \xi(t) \) and \( \xi(x) \) interchangeably. Each POD mode can be computed directly from data by the method of snapshots \(^{18} \), and the \( i \) th POD coefficient can be obtained by projecting \( \mathbf{x} - \bar{x} \) onto the \( i \) th POD mode. The number of retained POD modes is determined by examining the cumulative energy of the POD eigenvalues. In general, \( n_{s} \) is selected so as to account for at least 95% of the total energy. We also note that implicit in (21) is a one-to-one correspondence between snapshots in \( x \)-space (\( x_{n} \)) and snapshots in \( \xi \)-space (\( \xi_{n} \)). This allows us to view any function of \( x_{n} \) as a function of \( \xi_{n} \), and vice versa.

In the POD subspace, the dynamics obeys

\[
\dot{\xi} = \Phi^{T} F(\bar{x} + \Phi \xi) \approx \mathbf{G}(\xi), \tag{22}
\]

so in principle we could use the neural-network approach to learn the OTD modes in the \( \xi \)-subspace. We would simply apply the method proposed in §III A and §III B with \( \xi \) in place of \( x \), \( G \) in place of \( F \), and \( \nabla \xi \mathbf{G} \) in place of \( L \); then project the dOTD modes learned in the \( \xi \)-subspace back to the full space, resulting in \( \Phi \mathbf{u}_{\xi}(\xi) \). However, this approach is flawed, because by construction it assumes that the OTD modes live in the same subspace as the state itself (that is, the POD subspace spanned by the columns of \( \Phi \)). In reality, the OTD modes belong the tangent space at point \( x \), whose principal directions have no reason to coincide with that of the state on the attractor. This inconsistency persists regardless of the number of POD modes used in (21), so that resorting to very large \( n_{s} \) does not solve the problem.

All is not lost, though, since we can use different projection subspaces for the state and the OTD modes:

\[
\mathbf{x} - \bar{x} = \Phi \xi(x), \tag{23a}
\]

\[
\mathbf{u}_{\xi}(x) = \Psi(x) \mu_{\xi}(x), \tag{23b}
\]

where \( \Psi(x) \in \mathbb{R}^{d \times n_{s}} \) is a reduced orthonormal basis of the tangent space at \( x \), and \( \mu_{\xi}(x) \in \mathbb{R}^{n_{s}} \) contains the basis coefficients. We note that the number of POD modes (\( n_{s} \)) and columns of \( \Psi \) (\( n_{s} \)) need not be the same, which allows for the possibility of learning the OTD modes in a subspace bigger than the POD subspace. If \( \Psi(x_{n}) \) is a good approximation of the tangent space at \( x_{n} \), then the \( K \) nearest neighbors of \( x_{n} \) satisfy

\[
x_{k} \approx x_{n} + \Psi(x_{n}) \mathbf{a}_{\mathbf{x}_{k}}, \quad k \in \{1, \ldots, K\}, \tag{24}
\]

where \( \mathbf{a}_{\mathbf{x}_{k}} \) is a matrix of coefficients. To compute \( \Psi(x_{n}) \) from data, we solve the minimization problem

\[
\Psi(x_{n}) = \underset{\Psi_{\mathbf{a}}}{\operatorname{argmin}} \| x_{k} - (x_{n} + \Psi_{\mathbf{a}}) \|^{2}, \tag{25}
\]

which is tantamount to a principal component analysis (PCA) of the set of nearest neighbors \( \{x_{k}\}_{k=1}^{K} \). In other words, the columns of \( \Psi(x_{n}) \) are the leading \( n_{s} \) POD modes of the \( K \) nearest neighbors of \( x_{n} \); we refer to them as the “tangent POD (tPOD) modes”. We note that this approach has been used as the basis for a number of manifold learning algorithms that involve reconstruction of tangent space from data \(^{44,45} \).

Now that we have available two low-dimensional representations for the state and the OTD modes, the learning problem
reduces to finding the collection of graphs

\[ \mu_j : \mathbb{R}^{n_x} \rightarrow \mathbb{R}^{n_y}, \]

\[ \xi \rightarrow \mu_j(\xi), \]  

(26)
given a dataset of reduced states \( \{ \xi_n \}_{n=1}^N \) and tangent bases \( \{ \Psi(\xi_n) \}_{n=1}^N \). To derive an equation for \( \mu_j \), we substitute (23a,b) into (10) and arrive at

\[ \nabla_\xi \mu_j G(\xi) = L_\Psi \mu_j - \langle \mu_j, L_\Psi \mu_j \rangle \]

\[ - \sum_{k=1}^{-(i-1)} \left[ \langle \mu_j, L_\Psi \mu_k \rangle + \langle \mu_k, L_\Psi \mu_j \rangle \right] \mu_k \]

\[ - \Psi^T \Psi \mu_j, \]  

(27)

Three remarks are in order. First, we note that (27) is a system of \( n_x \)-dimensional differential equations, much less expensive to solve than the original system of \( d \)-dimensional differential equations. Second, we note that the presence of an additional term on the right-hand side of (27) arising from the dependence of \( d \) to solve that the original system of \( d \)-dimensional basis \( \Pi \), which is the projection of the high-dimensional operator \( \Psi \) onto the reduced basis \( \Phi \), leading to \( \Psi(\xi) = \Phi \) for every point \( \xi \) on the attractor, leading to \( L_\Phi = \Phi^T L(\xi) \Phi \). But this is a poor approximation for \( L(\xi) \), because \( \Phi \) is a reduced basis of the phase space, not the tangent space.

To compute the reduced operator \( L_\Psi(x_n) \) from data, we use an approach similar to that proposed in §III B. We first recall that (18) provides a mechanism to compute the action of \( L(x_n) \) on a collection of perturbation vectors \( \{ v_k \}_{k=1}^K \) at point \( x_n \). But the columns of \( \Psi(x_n) \) are linear combinations of these perturbation vectors, as per the POD construction. Therefore, we may write \( \Psi(x_n) \approx \sum_n \kappa_n v_n \), where \( \kappa_n \in \mathbb{R}^{K \times n_x} \) is a matrix of coefficients. This leads to

\[ L_\Psi(x_n) \approx \kappa_n^T \nabla \kappa_n \sum v_n. \]  

(29)
The operator \( L_\Psi(x_n) \) has the advantage of being low-dimensional, so it can be computed offline and stored in memory, along with the POD-reduced vector field \( G(\xi_n) \).

The last term on the right-hand side of (27), however, is problematic because it involves the temporal derivative of the reduced tangent basis \( \Psi \). We could use the chain rule and write \( \Psi = \nabla_\xi \Psi G(\xi) \), but the gradient \( \nabla_\xi \Psi \) is expensive to compute. Another option is to use a finite-difference formula in the spirit of (16):

\[ \Psi(x_n) = \frac{\Psi(x_{n+1}) - \Psi(x_n)}{\Delta t} + O(\Delta t). \]

(30)

If \( \Delta t \) is sufficiently small, then tPOD modes smoothly deform from \( x_n \) to \( x_{n+1} \), so (30) is a good approximation. (We note in passing that continuity of \( u(x) \) requires continuity of both \( \Psi(x) \) and \( \mu_j(x) \).) With this approach, the term \( \Psi^T \Psi \) can be computed offline, and passed to the neural network as a dummy input.

Finally, we address the question of whether the “local” tangent bases \( \Psi(x) \) could be combined into a larger “global” subspace that does not depend on \( x \). Mathematically, this is equivalent to seeking a reduced basis of the tangent bundle

\[ T \mathcal{X} = \bigcup_{\xi \in \mathcal{X}} T_\xi \mathcal{X}, \]  

(31)

where \( \bigcup \) is the disjoint union operator. Such a construct would have the benefit of eliminating the last term on the right-hand side of (27). Also, it would be aesthetically more appealing, because the OTD modes would be learned in a common subspace, regardless of the point at which they are computed. To construct a reduced basis of \( T \mathcal{X} \), one can perform POD on the set of tangent bases \( \{ \Psi(x_n) \}_{n=1}^N \). This results in a set of “bundle modes”, denoted by \( \Pi \in \mathbb{R}^{n_x \times n_b} \), whose span is the best \( n_b \)-dimensional approximation of the tangent bundle \( T \mathcal{X} \). The expectation is that the number of bundle modes \( n_b \), although generally greater than the number of local tPOD modes \( n_e \), will still be much smaller than \( d \). The DOTD modes are sought as vectors in the bundle space, that is, \( u(x) = \Pi \rho(x) \), where the coefficients \( \rho \in \mathbb{R}^{n_b} \) satisfy

\[ \nabla_\xi \rho \Psi G(\xi) = L_\Pi \rho - \langle \rho, L_\Pi \rho \rangle \]

\[ - \sum_{k=1}^{n_b} \left[ \langle \rho, L_\Pi \rho_k \rangle + \langle \rho_k, L_\Pi \rho \rangle \right] \rho_k, \]

(32)

and \( L_\Pi \) is defined as

\[ L_\Pi = \Pi^T L(\xi) \Pi \in \mathbb{R}^{n_b \times n_b}. \]

(33)

For \( n_b \) not too large, the reduced operator \( L_\Pi \) may be computed offline and stored in memory.

Applicability of the tangent-bundle method is limited to cases in which the governing equations of the dynamical system are available. This is because \( L_\Pi \) cannot be computed solely from snapshot data, except when the linearized operator is known \( L(x) \) explicitly. To see this, we first recognize that by construction, each bundle mode is a linear combination of the columns of \( Y = [V_1, \ldots, V_m] \). So evaluation of (33) from data is conditioned on our ability to compute \( L(x_m)Y \) or, equivalently, each member of the set \( \{ L(x_m)Y_m \}_{m=1}^N \). But there is no mechanism to compute \( L(x_m)Y_m \) from data when \( m \neq n \). (Equation (18) holds only for \( m = n \).) Thus, evaluation of (33) requires explicit knowledge of the linearized operator and, in turn, the governing equations.

D. Implementation

We conclude this section with a few words on implementation. The neural network is built from scratch in Python. We use Autograd for automatic differentiation. The activation function between hidden layers is the hyperbolic tangent, although other choices (e.g., sigmoid function or swish
function) are possible. The activation function for the output layer is a linear function. As discussed in §III A 2, the last layer of the neural network is followed by a Gram–Schmidt layer. The weights are initialized according to Xavier initialization. For Lyapunov regularization, we use $\sigma = \sinh$.

We use Adam to solve the optimization problem. In its current manifestation, the code supports mini-batching, although caution must be exercised when selecting the batch size in order not to deteriorate the accuracy of (14). (The results presented in §IV do not use mini-batching.) We have found no need for specifying learning rate schedules in the optimizer. Different stopping criteria may be used, depending on how the vector field and linearized operator are computed. If $F(x_n)$ and $L(x_n)$ are evaluated from the governing equations, and the hypothesis class is reasonably large, then optimization may be terminated when the loss function (11) is smaller than a user-specified tolerance. (As a result, the overall error is dominated by the approximation error.) If $d(x)$ is small, but $d(x)$ is large. Should that occur, a quick fix is to increase the dimension of the reduced subspace $\Phi$.

To evaluate the accuracy of the learning algorithm, we define the distance

$$d_i(x) = ||u_i^{\text{deep}}(x), u_i^{\text{num}}(x)||,$$

where $u_i^{\text{deep}}$ denotes the $i$th dOTD mode, and $u_i^{\text{num}}$ denotes the $i$th OTD mode computed by direct numerical integration of (5). The distance $d_i$ takes values between 0 and 1, with the former indicating that $u_i^{\text{deep}}$ and $u_i^{\text{num}}$ are orthogonal, and the latter indicating that $u_i^{\text{deep}}$ and $u_i^{\text{num}}$ coincide. We could also use the distance between the subspaces $\{u_i^{\text{deep}}\}_{i=1}^r$ and $\{u_i^{\text{num}}\}_{i=1}^r$, but that measure, unlike (34), assigns the same score to all the SLBs, regardless of stability (when $r = d$).

If the OTD modes are learned in a reduced subspace, then it is useful to compute

$$d_i(x) = ||u_i^{\text{deep}}(x), \Theta \Theta u_i^{\text{num}}(x)||,$$

where $\Theta$ is a placeholder for $\Psi(x)$ or $\Pi$, depending on whether the “local” or “bundle” approach is used. The above is equivalent to computing the distance function in the reduced subspace in which the modes are learned. This quantity is important, because dimensionality reduction of the tangent space (or tangent bundle) introduces an additional error over which the neural network has no control. Thus, situations may arise in which $d_i(x)$ is small, but $d_i(x)$ is large. Should that occur, a quick fix is to increase the dimension of the reduced subspace.

### A. Low-dimensional nonlinear system

We begin with a three-dimensional nonlinear system that was proposed by Noack et al. as a testbed for investigating Hopf bifurcations in laminar bluff-body flows. We choose this system because it provides a good illustration for many of the comments made in §II and §III. The governing equations are given by

$$\begin{align*}
\dot{x} &= \mu x - y - xz, \\
\dot{y} &= \mu y + x - yz, \\
\dot{z} &= -z + x^2 + y^2,
\end{align*}$$

with $\mu$ a positive constant. The system admits a linearly unstable fixed point ($x = y = z = 0$), and a stable periodic solution,

$$x = \sqrt{\mu} \cos t, \quad y = \sqrt{\mu} \sin t, \quad z = \mu,$$

which defines a limit cycle of radius $\sqrt{\mu}$ in the $z = \mu$ plane. As noted by Noack et al., the limit cycle is asymptotically and globally stable.

For $\mu \leq 1/8$, it is possible to derive analytical expressions for the OTD modes on the limit cycle:

$$u_1 = \begin{bmatrix} -\sin t \\ \cos t \\ 0 \end{bmatrix}, \quad u_2 = \begin{bmatrix} -a \cos t \\ a \sin t \\ -b \end{bmatrix}, \quad u_3 = \begin{bmatrix} -b \cos t \\ -a \sin t \\ a \end{bmatrix},$$

where

$$a = \frac{1 + \sqrt{1 - 8\mu}}{4\sqrt{\mu}}, \quad b = \sqrt{\frac{1 + 4\mu - \sqrt{1 - 8\mu}}{2(1 + \mu)}}.$$
\[ x = [x \ y \ z]^{T} \] as follows:

\[
\begin{align*}
\mathbf{u}_{1}(x) &= \frac{\mathbf{F}(x)}{||\mathbf{F}(x)||}, \quad \mathbf{u}_{2}(x) = -\frac{a}{\sqrt{\beta}} x - \begin{bmatrix} 0 \\ 0 \\ b + a/\sqrt{\beta} \end{bmatrix}, \\
\mathbf{u}_{3}(x) &= -\frac{b}{\sqrt{\beta}} x + \begin{bmatrix} 0 \\ 0 \\ a - b/\sqrt{\beta} \end{bmatrix}.
\end{align*}
\]

(39)

The ordered set \( \{\mathbf{u}_{1}(x), \mathbf{u}_{2}(x), \mathbf{u}_{3}(x)\} \) is the unique stable SLB at point \( x \) on the limit cycle, although other unstable SLBs exist. If we define

\[
a^\pm = \frac{1 \pm \sqrt{1 - 8\mu}}{4\sqrt{\beta}} b^\pm, \quad b^\pm = \sqrt{\frac{1 + 4\mu \pm \sqrt{1 - 8\mu}}{2(1 + \mu)}},
\]

(40)

then any of the ordered triples \( \{\mathbf{u}_{1}, \mathbf{u}_{3}^+, \mathbf{u}_{3}^+\} \), \( \{\mathbf{u}_{2}^+, \mathbf{u}_{1}, \mathbf{u}_{1}^+\} \), and \( \{\mathbf{u}_{3}^-, \mathbf{u}_{3}^-, \mathbf{u}_{1}\} \) are solutions to the OTD equations, but only \( \{\mathbf{u}_{1}, \mathbf{u}_{3}^+, \mathbf{u}_{3}^+\} \) is stable.

We consider the case \( \mu = 0.1 \), and use the neural-network approach to learn the graphs \( \{x \rightarrow \mathbf{u}(x)\}_{i=1}^{3} \) from data. We begin by generating a long trajectory initialized on the limit cycle with \( x = \sqrt{\beta}, y = 0 \) and \( z = \mu \). This trajectory is computed by a third-order Adams–Bashforth method with time-step size \( \Delta t = 0.01 \) for a total duration of \( T = 600 \) time units. From this long trajectory, we uniformly sample \( N = 10 \) points over one period of the limit cycle; these are our training points. The vector field and linearized operator are computed analytically, and passed to the neural network as dummy inputs. The neural network is composed of one hidden layer with 20 neurons, and the learning rate for the Adam optimizer is set to 0.04. Lyapunov regularization is active for the first 1000 iterations, and turned off for the remainder of the optimization.

The dOTD modes are learned sequentially, with the maximum number of iterations specified as 2000. The testing data consists of 1256 points sampled uniformly (with time-step size \( \Delta t \)) over two periods of the limit cycle.

Figures 2a-c show time series for the dOTD modes \( \mathbf{u}^{\text{deep}} \) and the numerically integrated OTD modes \( \mathbf{u}^{\text{num}} \) at the training points (black stars) and testing points (colored lines). The neural network is able to learn the graphs \( \{x \rightarrow \mathbf{u}(x)\}_{i=1}^{3} \) from the limited number of training points supplied to it. The agreement at the testing points is excellent. We also note that Lyapunov regularization is instrumental in the optimizer converging to the stable SLB (39). The learned Lyapunov exponents (averaged over 10 learning experiments) are found to be \( \lambda_{1} = 1.6 \times 10^{-8}, \lambda_{2} = -0.276 \) and \( \lambda_{3} = -0.724 \), virtually identical to their counterparts computed by numerical integration \( \lambda_{1} = 0, \lambda_{2} = -0.276 \) and \( \lambda_{3} = -0.723 \).

### B. Charney–DeVore system

Next, we consider a modified version of the classical Charney–DeVore model, which describes atmospheric circulations at midlatitudes. We consider a six-dimensional truncation of the original system, which models barotropic flow in a plane channel with topography. The governing equations are given by

\[
\begin{align*}
\dot{z}_{1} &= \gamma_{1} z_{3} - C(z_{1} - z_{1}^{*}), \\
\dot{z}_{2} &= -(\alpha_{1} z_{1} - \beta_{1} z_{2} - C z_{2} - \delta_{1} z_{4} z_{6}), \\
\dot{z}_{3} &= (\alpha_{1} z_{1} - \beta_{1} z_{2} - \gamma_{1} z_{1} - C z_{3} + \delta_{1} z_{4} z_{5}), \\
\dot{z}_{4} &= \gamma_{2} z_{6} - C(z_{4} - z_{4}^{*}) + \mu(z_{2} z_{6} - z_{3} z_{5}), \\
\dot{z}_{5} &= -(\alpha_{2} z_{2} - \beta_{2} z_{5} - C z_{5} - \delta_{2} z_{4} z_{3}), \\
\dot{z}_{6} &= (\alpha_{2} z_{1} - \beta_{2} z_{6} - \gamma_{2} z_{4} - C z_{6} + \delta_{2} z_{4} z_{2}^{*}),
\end{align*}
\]

(41)

with parameters

\[
\begin{align*}
\alpha_{m} &= 8 \sqrt{2} m^{2} (b^{2} + m^{2} - 1)/[\pi(4m^{2} - 1)(b^{2} + m^{2})], \\
\delta_{m} &= 64 \sqrt{2}(b^{2} + m^{2} - 1)/[15 \pi(b^{2} + m^{2})], \\
\mu &= 16 \sqrt{2}/(5 \pi), \\
\beta_{m} &= \beta b^{2}/(b^{2} + m^{2}), \\
\gamma_{m} &= 4 \sqrt{2} m^{3} \gamma b/[\pi(4m^{2} - 1)(b^{2} + m^{2})], \\
\gamma_{m}^{*} &= 4 \sqrt{2} m^{3} \gamma b/[\pi(4m^{2} - 1)],
\end{align*}
\]

where \( m = 1 \) or 2. The parameters \( \sigma_{m} \) and \( \delta_{m} \) account for zonal advection in the \( z_{1} \) and \( z_{4} \) directions, respectively; \( \beta_{m} \) for the so-called \( \beta \) effects; \( \gamma_{m} \) and \( \gamma_{m}^{*} \) for topographic interactions; \( C \) for Ekman damping; and \( z_{1}^{*} \) and \( z_{4}^{*} \) for zonal forcing in the \( z_{1} \) and \( z_{4} \) directions, respectively. We set \( z_{1}^{*} = 0.95, z_{4}^{*} = -0.76095, C = 0.1, \beta = 1.25, \gamma = 0.2 \) and \( b = 0.5 \). These values of the parameters give rise to significant transitions between regimes of “zonal” and “blocked” flow, resulting from nonlinear interaction between barotropic and topographic instabilities. The extreme episodes of blocked flow are of main interest to us, because they are the manifestation of transient instabilities. Thus, it is in those intervals where we attempt to learn the OTD modes from data.

We use a third-order Adams–Bashforth scheme with time-step size \( \Delta t = 0.05 \) to generate a long trajectory spanning 4000 time units. We use zero initial conditions, except for \( z_{1}(0) = 1.14 \) and \( z_{4}(0) = -0.91 \). We consider the interval \( 1075 \leq t \leq 1165 \) during which the trajectory passes through a regime of blocked flow (figures 3a–f), and select 50 uniformly-spaced training points in that interval. For each training point, the velocity field \( \mathbf{F}(x_{0}) \) and the Jacobian matrix \( \mathbf{L}(x_{0}) \) are reconstructed using 60 nearest neighbors, and then passed to the neural network as dummy inputs. The neural network has two hidden layers, each with 128 neurons. The learning rate for the Adam algorithm is 0.001. Lyapunov regularization is used for the first 2000 iterations, and switched off thereafter. Optimization is terminated at 5000 iterations. We attempt to learn only the first OTD mode \( \mathbf{u}_{1} \). For the testing data, we use all the data points from the long trajectory in the interval \( 500 \leq t \leq 4000 \). This interval contains multiple episodes of blocked flow (figure 4a).

Figure 4b shows time series for the distance \( d_{1}(x(t)) \), which measures agreement between \( \mathbf{u}^{\text{deep}} \) and \( \mathbf{u}^{\text{num}} \). Not surprisingly, the neural network is able to learn the mapping from phase space to OTD space in the interval \( 1075 \leq t \leq 1165 \) in which the training points where supplied. Much more remarkable is the outstanding agreement in other intervals of
FIG. 2. For system (36a–c) with $\mu = 0.1$, time series for the dOTD modes evaluated on testing data (solid red), along with time series for the numerically integrated OTD modes (dashed blue), and markers identifying the 10 training points for which $x$ was provided to the learning algorithm (black stars); (a) first mode, (b) second mode, and (c) third mode.

FIG. 3. For the Charney–DeVore system, details of a trajectory passing through a regime of blocked flow.

blocked flow (e.g., $1340 \leq t \leq 1410$ and $2900 \leq t \leq 3100$)—that is, the region in phase space synonymous with transient instabilities—showing that the neural network only needs to know what a single interval of blocked flow looks like to be able to predict all other such intervals, past or future. This level of prediction capability is unprecedented in the context
of extreme events in dynamical systems. Figure 4b also shows that agreement is generally poor outside intervals of blocked flow. This should come as no surprise, because neural networks are known to perform poorly when the testing data looks nothing like the training data. (The fundamental assumption for generalizability is that training and testing data are drawn independently from the same probability distribution.)

Our attempts to train the neural network in an interval of zonal flow, or in an interval containing both blocked and zonal flow regimes, were unsuccessful. We suspect that this is because the intervals of zonal flow are more “chaotic” (with more time scales involved) that intervals of blocked flow. Our numerical experiments suggest that improving expressivity of the neural network (by increasing the number of hidden layers and neurons) does not solve the problem. We note that a similar observation was made Raissi, who attempted to machine-learn the Kuramoto–Sivashinsky equation with a neural network. Raissi noted that for this system, intervals of laminar flow posed no difficulty to the neural network, while chaotic intervals were “stubbornly” challenging, with the optimization algorithm not converging to the “right values” of the network parameters. This description aligns with what we have seen in the present investigation of the Charney–DeVore system. We leave investigation of this issue for future work.

C. Flow past a cylinder

We conclude this section with an application of the learning algorithm to a high-dimensional dynamical system. This is to provide an illustration of the Galerkin approach proposed in §III C. Specifically, we consider the flow of a two-dimensional fluid of density \( \rho \) and kinematic viscosity \( \nu \) past a rigid circular cylinder of diameter \( D \) with uniform free-stream velocity \( U \). The Navier–Stokes equations can be written in dimensionless form as

\[
\begin{align*}
\partial_t x + x \cdot \nabla x &= -\nabla p + \frac{1}{Re} \nabla^2 x, \\
\nabla \cdot x &= 0,
\end{align*}
\]

with no-slip boundary condition \((x = 0)\) on the cylinder surface, and uniform flow \((x = e_r)\) in the far field. Velocity, time and length have been scaled with cylinder diameter \( D \) and free-stream velocity \( U \), and the Reynolds number is \( Re = UD/v \). We consider the case \( Re = 50 \), for which there exists a limit-cycle attractor which is believed to be globally and asymptotically stable. Our computational approach (mesh topology, spatial discretization, and time-stepping scheme) is identical to that used by Blanchard et al.\(^{36,37}\)

This flow lends itself to dimensionality reduction, because the limit-cycle attractor, while being part of an infinite-dimensional phase space, is low-dimensional, with a handful of POD modes faithfully capturing nearly all of the energy. (In fact, the system discussed in §IV A was originally introduced as a simplified model for this flow.) Low-dimensionality of the attractor is important for leveraging the full power of the reduced-order learning algorithm proposed in §III C. We also note that learning the dOTD modes on the limit cycle does not have much merit from the standpoint of predicting instabilities, but it does from the standpoint of flow control. As discussed in §V, having access to the OTD modes at any point along the periodic orbit is the stepping stone for application of the OTD control strategy proposed by Blanchard et al.\(^{37}\).

We begin with the generation of a long trajectory on the limit cycle by integrating the Navier–Stokes equations for 400 time units (corresponding to about 52 periods). The POD modes \( \Phi \) are computed using 192 snapshots equally spaced over one period. We retain the first \( n_s = 8 \) POD modes, accounting for more than 99.9% of the cumulative energy. Time series for the POD coefficients \( \xi \in \mathbb{R}^8 \) are generated by projecting the 400-time-unit trajectory on the POD modes. The training set in \( \xi \)-space is formed by uniformly sampling \( N = 20 \) points over one period of the limit cycle. For each training point \( \xi_n \), we reconstruct the vector field \( G(\xi_n) \) using an Euler-forward finite-difference approximation in \( \xi \)-space. (Alternatively, one can project (16) on the POD modes.)

To compute the reduced basis in which the OTD modes will be learned, we consider the local approach proposed in §III C. We compute the tPOD modes \( \{\Psi(x_n)\}_{n=1}^N \) using the 50 nearest neighbors of each \( x_n \), and then use (29) and (30) to reconstruct the reduced linearized operator and the last term on the right-hand side of (27), respectively. We consider local tangent bases with dimension ranging from \( n_t = 2 \) to 6. The neural network has two hidden layers, each comprising 32 hidden units. The Adam optimization algorithm uses a learning rate of 0.01, and is terminated after 2000 iterations. Lyapunov regularization is turned off after 100 iterations. The testing data consists of 110 points sampled uniformly over two periods of the limit cycle.

For \( n_t = 4 \), the distances \((d_i(x))^2_{i=1}^2\) and \((d_i^n(x))^2_{i=1}^2\) are virtually equal to one for all the testing points. Specifically, the standard deviation of \( 1 - d_i \) is found to be \( 3.8 \times 10^{-3} \) and \( 1.1 \times 10^{-2} \) for \( i = 1 \) and 2, respectively; and the standard deviation of \( 1 - d_i^\Psi \) is found to be \( 1.2 \times 10^{-4} \) and \( 1.9 \times 10^{-4} \) for \( i = 1 \) and 2, respectively. (These numbers were obtained by averaging over ten learning experiments.) This shows that a) the error introduced by the low-dimensional reconstruction of the tangent space is essentially zero, and b) the neural network finds the best representation of the OTD modes in that reduced tangent space. Figures 5a–f provide visual confirmation that the dOTD modes learned in the reconstructed tangent space are indistinguishable from their numerically integrated counterparts. These results illustrate the benefits of learning the OTD modes in a reduced subspace. Equally good agreement was obtained for \( n_t = 2 \) and 6 with the same network parameters.

V. DISCUSSION

We now discuss possible improvements and modifications to the OTD learning algorithm introduced in §III, as well as implications for data-driven control of instabilities in dynamical systems.

In the examples discussed in §IV, the nonlinearity appear-
FIG. 4. For the Charney–DeVore system, time series of (a) the first state coordinate $z_1$, (b) the distance function with $F(x_n)$ and $L(x_n)$ reconstructed from data, and (c) the distance function with $F(x_n)$ and $L(x_n)$ computed analytically. The 50 training points are equally spaced in the interval $1075 \leq t \leq 1165$.

The governing equations was no stronger than quadratic with respect to the state variables, and the neural network was able to learn the OTD graph using the state $x$ as its only “active” input. In cases in which the nonlinearity is known to be, or suspected to be, stronger than quadratic (e.g., with terms involving higher-order polynomials, trigonometric functions, or nonlinear differential operators), it is likely that supplying $x$ as the only input will call for wider, deeper networks than used in §IV. To keep the number of network parameters reasonably small and facilitate convergence of the optimization algorithm, one possibility is to use as additional inputs a library of nonlinear functions of the state; for example, ${x_n, F(x_n), \sin(x_n), \exp(-x_n^2), x_n \cdot \nabla x_n}_{n=1}^N$. This approach is in the same spirit as the SINDy algorithm, in which sparse regression is applied to a library of nonlinear functions of $x$ in order to discover governing equations from state measurements.

We also note that in any laboratory experiment, sensing capabilities are limited by the apparatus, leading to errors in state estimation and reconstruction. Uncertainty in state measurements may be accounted for by trading the neural-network approach for one based on Gaussian processes (GPs), because GPs have the advantage of providing error estimates at each testing points. GPs have been found capable of handling sizable noise levels in a number of problems similar to the present, including deep learning of partial differential equations and discovery of governing equations from noisy measurements.

The method proposed in §III is fully data-driven, in the sense that no input other than state snapshots is required to learn the dOTD modes. (The vector field and linearized operator are reconstructed using nothing but state snapshots.) If governing equations are available (either derived from first principles or reconstructed from data), then there is another possibility to learn the graphs $x \rightarrow u_i(x)$; that is, generate a large number of ${x_n, u_i(x_n)}$ pairs by solving the state and OTD equations numerically, and model the input–output relationship by a neural network, whose parameters are found by minimizing the empirical risk

$$
\ell_{emp}^{\text{emp}}(\theta_i) = \frac{1}{N} \sum_{n=1}^N \ell(x_n, u_i(x_n; \theta_i)),
$$

where $\ell$ is an appropriate loss function (e.g., quadratic loss or log-cosh loss). To $\ell_{emp}^{\text{emp}}$ may be appended the loss function (11), which then acts as a physics-informed (equation-based) regularization term. The downsides of this approach are that a) access to governing equations is mandatory, and b) generating the input–output pairs requires solving $r+1$ d-dimensional initial-boundary-value problems, which may be computationally expensive.

From the standpoint of predicting instabilities, the benefit of learning the dOTD modes from data is that it gives access
FIG. 5. For flow past a cylinder at \(Re = 50\), vorticity distributions of (a) \(u^\text{deep}_1\), (b) \(u^\text{deep}_2\), (c) \(u^\text{num}_1\), (d) \(u^\text{num}_2\), (e) \(u^\text{num}_1\), and (f) \(u^\text{num}_2\), computed with \(n_t = 4\) for a particular state on the limit cycle attractor.

to directions of instabilities at any point in phase space, and to the leading Lyapunov exponents, regardless of their sign. But the dOTD learning algorithm also has significant implications from the standpoint of controlling instabilities. As discussed in §II A, we have recently shown that the OTD modes can be incorporated into reduced-order control algorithms for stabilization of unsteady high-dimensional flows. The OTD control strategy proposed in Ref. 37 requires solving the OTD equations concurrently with the state equations, because the control force belongs to the OTD subspace. With the dOTD learning approach, this requirement disappears, because the neural-network approach can be used to build a library of dOTD modes for various regions of the phase space. (Construction of the OTD library may be done offline.) Then, as the controlled trajectory wanders about in phase space, the controller can look up in the OTD library the dOTD modes associated with the current state. (If the trajectory visits a state that is not present in the library, then one can interpolate between nearby states for which dOTD modes are available.) Library look-up can be done in real time because the computational complexity of the look-up algorithm scales with the dimension of the OTD subspace, which makes the approach very attractive from the standpoint of controlling high-dimensional systems in real time. We note that similar ideas were employed in the context of nonlinear model order reduction by Amsallem et al.\textsuperscript{(20,55)}.

Finally, there is the issue of how the predictive capabilities of the neural network are affected by changes in system parameters. We first note that even a small change in system parameters has the potential of considerably altering the topology of the phase space, and in particular, the number, nature, and properties of the attractors. (This is apparent when a bifurcation occurs.) We also note that one of the prerequisites for the neural network to perform well on unseen data is that the training and testing data be drawn from the same underlying probability distribution. Large variations in system parameters are likely to violate this assumption, seriously compromising generalizability of the neural network. Small variations in system parameters may lead to good generalizability provided that the changes in phase-space topology and underlying probability distribution are also small.

VI. CONCLUSION

For a large class of dynamical systems, the optimally time-dependent (OTD) modes, a set of deformable orthonormal tangent vectors that track directions of instabilities along any trajectory, are known to depend pointwise on the state of the system on the attractor, and not on the history of the trajectory. We have developed a learning algorithm based on neural networks to learn this pointwise mapping from phase space to
ODT space using data collected along one or more trajectories of the system. The proposed method is fully data-driven as it requires no other input than snapshots of the state, and is applicable regardless of the dimensionality of the system. The learning process provides a cartography of directions associated with strongest instabilities in phase space, as well as accurate estimates for the leading Lyapunov exponents of the attractor. This has significant implications for data-driven prediction of dynamical instabilities, with the learning algorithm exhibiting predictive capabilities of extreme events to a degree that is unprecedented, but also for design and implementation of reduced-order controllers capable of operating in real time.

ACKNOWLEDGMENTS

The authors acknowledge discussions with Dr. Hassan Arbabi. This work was supported by the Army Research Office (Grant No. W911NF-17-1-0306).

REFERENCES
