Learning Dynamics Models with Stable Invariant Sets

Naoya Takeishi [1,2]**, Yoshinobu Kawahara** [3,1]

[1] RIKEN Center for Advanced Intelligence Project
[2] University of Applied Sciences and Arts Western Switzerland
[3] Kyushu University
naoya.takeishi@riken.jp,   kawahara@imi.kyushu-u.ac.jp

Abstract

Invariance and stability are essential notions in dynamical systems study, and thus it is of great interest to learn a dynamics model with a stable invariant set. However, existing methods can only handle the stability of an equilibrium. In this paper, we propose a method to ensure that a dynamics model has a stable invariant set of general classes such as limit cycles and line attractors. We start with the approach by Manek and Kolter (2019), where they use a learnable Lyapunov function to make a model stable with regard to an equilibrium. We generalize it for general sets by introducing projection onto them. To resolve the difficulty of specifying a to-be stable invariant set analytically, we propose defining such a set as a primitive shape (e.g., sphere) in a latent space and learning the transformation between the original and latent spaces. It enables us to compute the projection easily, and at the same time, we can maintain the model's flexibility using various invertible neural networks for the transformation. We present experimental results that show the validity of the proposed method and the usefulness for long-term prediction.

1 Introduction

Machine learning of dynamical systems appears in diverse disciplines, such as physics (Raissi, Perdikaris, and Karniadakis 2019), biology (Costello and Martin 2018), chemistry (Li, Kermode, and De Vita 2015), and engineering (Morton et al. 2018). Recent progress in dynamics models includes the Gaussian process dynamics models (Wang, Fleet, and Hertzmann 2006) and models based on deep neural networks (e.g., Takeishi, Kawahara, and Yairi 2017; Lusch, Kutz, and Brunton 2018; Chen et al. 2018; Manek and Kolter 2019; Greydanus, Dzamba, and Yosinski 2019). Whatever models are employed, we often would like to know and control the nature of a learned dynamics model, for example, to reflect prior knowledge of the dynamics and to ensure specific behavior of the learned dynamics model.

Invariance and *stability* of some subsets of a state space play a key role in dynamical systems study as they concisely describe the asymptotic behavior (i.e., in $t \to \infty$) of a system. For example, dynamical systems with stable invariant sets are used to explain neurological functions such as working memory and eye control (Eliasmith 2005). Moreover, various self-sustained oscillations in physical, chemical, or biological phenomena are modeled as systems with stable closed orbits (Strogatz 2015). In the dynamical systems study, *analyzing* system's stability has been a classical yet challenging problem. In contrast, *synthesizing* (i.e., achieving) stability of some dynamics models is a problem that has been addressed mainly in automatic control and machine learning. Whereas control theory has been trying to achieve stable systems by designing control inputs, another possible strategy is to learn a dynamics model from data with a constraint that the model attains some desired stability property.

In this work, we tackle the problem of learning dynamics models with guaranteed invariance and stability. This problem is important in many practices of machine learning. For example, we often have prior knowledge that a target phenomenon shows self-sustained oscillations (Strogatz 2015). Such prior knowledge is a powerful inductive bias in learning and can be incorporated into learning by forcing a model to have a stable limit cycle. Likewise, we often want to assure the invariance and stability (i.e., time-asymptotic behavior) of a learned forecasting model for meaningful prediction or safety issues. To these ends, we need a method to guarantee invariance and stability of a dynamics model.

Learning dynamics models with provable stability is not a new task. Learning linear dynamical systems with stability (e.g., Lacy and Bernstein 2003; Siddiqi, Boots, and Gordon 2008; Huang et al. 2016) is a long-standing problem, and learning stable nonlinear dynamics has also been addressed by many researchers (e.g., Khansari-Zadeh and Billard 2011; Neumann, Lemme, and Steil 2013; Umlauft and Hirche 2017; Duncker et al. 2019; Chang, Roohi, and Gao 2019; Manek and Kolter 2019; Massaroli et al. 2020). However, these methods can only handle the stability of a finite number of equilibria (i.e., points where a state remains if no external perturbation applies) and are not suitable for guaranteeing general stable invariant sets (e.g., limit cycles, limit tori, and continuous sets of infinitely many equilibria). This limitation has been hindering useful applications of machine learning on dynamical systems, for example in physics and biology.

We develop a dynamics model with provable stability of general invariant sets. The starting point of our model is the approach by Manek and Kolter (2019), where they use a learnable Lyapunov function to modify a base dynamics model to ensure the stability of an equilibrium. We generalize

it for handling general subsets of state space (e.g., closed orbits, surfaces, and volumes) as stable invariant sets, by introducing projection onto such sets in the definition of the learnable Lyapunov function. A practical difficulty arising here is that in general, we cannot specify the geometry of a to-be stable invariant set analytically. To resolve this difficulty, we propose defining such a set as a primitive shape (e.g., a sphere) in a latent space and learning the transformation between the original state space and the latent space (see Figure 1). We can configure such a primitive shape so that the projection is easily computed. At the same time, we can maintain the flexibility of the model using the rich machinery of invertible neural networks that have been actively studied recently (see, e.g., Papamakarios et al. 2019).

In the remainder, we review the technical background in Section 2. We first give a general definition of the proposed dynamics model in Section 3 and then show its concrete constructions in Section 4. We introduce some related studies in Section 5. We present experimental results in Section 6, with which we can confirm the validity of the proposed method and its usefulness for the application of long-term prediction. The paper is concluded in Section 7.

2 Background

2.1 Invariant Sets of Dynamical Systems

We primarily consider a continuous-time dynamical system described by an ordinary differential equation

$$\dot{\boldsymbol{x}} = \boldsymbol{f}(\boldsymbol{x}), \tag{1}$$

where $\boldsymbol{x} \in \mathcal{X} \subset \mathbb{R}^d$ is a state vector in a state space $\mathcal{X}$. $\dot{\boldsymbol{x}}$ denotes the time derivative, $\mathrm{d}\boldsymbol{x}/\mathrm{d}t$. We assume that $\boldsymbol{f} : \mathcal{X} \to \mathcal{X}$ is a locally Lipschitz function. We denote the solution of (1) with initial condition $\boldsymbol{x}(0) = \boldsymbol{x}_0$ as $\boldsymbol{x}(t)$. An *invariant set* of a dynamical system is defined as follows:

Definition 1 (Invariant set). An invariant set S of dynamical system (1) is a subset of $\mathcal{X}$ such that a trajectory $\boldsymbol{x}(t)$ starting from $\boldsymbol{x}_0 \in S$ remains in S, i.e., $\boldsymbol{x}(t) \in S$ for all $t \geq 0$.

2.2 Stability of Equilibrium

A state $\boldsymbol{x}_e$ s.t. $\boldsymbol{f}(\boldsymbol{x}_e) = 0$ is called an *equilibrium* of (1) and constitutes a particular class of invariant sets. One of the common interests in analyzing dynamical systems is the *Lyapunov stability* of equilibria (e.g., Hirsch, Smale, and Devaney 2003; Giesl and Hafstein 2015). Informally, an equilibrium $\boldsymbol{x}_e$ is stable if the trajectories starting near $\boldsymbol{x}_e$ remain around it all the time. More formally;

Definition 2 (Stability of equilibrium). An equilibrium $\boldsymbol{x}_e$ is said to be Lyapunov stable if for every $\epsilon > 0$, there exists $\delta > 0$ such that, if $\|\boldsymbol{x}(0) - \boldsymbol{x}_e\| < \delta$, then $\|\boldsymbol{x}(t) - \boldsymbol{x}_e\| < \epsilon$ for all $t \geq 0$. Moreover, if $\boldsymbol{x}_e$ is stable, and $\boldsymbol{x}(t) \to \boldsymbol{x}_e$ as $t \to \infty$, $\boldsymbol{x}_e$ is said to be *asymptotically stable*.

The (asymptotic) stability of equilibria plays a crucial role in analyzing dynamical systems as well as in applications. For example, in computational neuroscience, dynamical systems with stable equilibria are used to explain phenomena such as associative memory and pattern completion. In physics, coupled phase oscillators whose equilibria are stable are used

to model synchronization phenomena. In engineering, equilibrium stability is used for roughly assessing the safety of controlled agents and the plausibility of forecasting.

Lyapunov's direct method is a well-known way to assess the stability of equilibria (see, e.g., Hirsch, Smale, and Devaney 2003), which can be summarized as follows:

Theorem 1 (Lyapunov's direct method). *Let $\boldsymbol{x}_e$ be an equilibrium of dynamical system* (1). *Let $V : \mathcal{U} \to \mathbb{R}$ be a function on a neighborhood $\mathcal{U}$ of $\boldsymbol{x}_e$, and further suppose:*

(A) V has a minimum at $\boldsymbol{x}_e$; e.g., a sufficient condition is:
$$(\forall \boldsymbol{x} \in \mathcal{U} \ V(\boldsymbol{x}) \geq 0) \wedge (V(\boldsymbol{x}) = 0 \Leftrightarrow \boldsymbol{x} = \boldsymbol{x}_e).$$
(B) V is strictly decreasing along trajectories of (1)*; e.g., when V is differentiable, a sufficient condition is:*
$$(\forall \boldsymbol{x} \in \mathcal{U} \backslash \{\boldsymbol{x}_e\} \ \dot{V} = \mathrm{d}V/\mathrm{d}t = \langle \nabla V(\boldsymbol{x}), \boldsymbol{f}(\boldsymbol{x}) \rangle < 0.$$
If such a function V exists, then it is called a Lyapunov function, and $\boldsymbol{x}_e$ is asymptotically stable.

2.3 Dynamics Models with Stable Equilibrium

Manek and Kolter (2019) proposed a concise method to ensure the stability of an equilibrium of a dynamics model by construction. They suggested learning a function V that satisfies condition (A) in Theorem 1 with neural networks and projecting outputs of a base dynamics model onto a space where condition (B) also holds. Consequently, the modified model's equilibrium becomes asymptotically stable.

Their dynamics model, $\dot{\boldsymbol{x}} = \boldsymbol{f}(\boldsymbol{x})$, is built as

$$\boldsymbol{f}(\boldsymbol{x}) = \begin{cases} \hat{\boldsymbol{f}}(\boldsymbol{x}) - \frac{\beta(\boldsymbol{x})}{\|\nabla V(\boldsymbol{x})\|_2^2} \nabla V(\boldsymbol{x}), & \text{if } \beta(\boldsymbol{x}) \geq 0, \\ \hat{\boldsymbol{f}}(\boldsymbol{x}), & \text{otherwise,} \end{cases} \tag{2}$$
$$\text{where} \quad \beta(\boldsymbol{x}) = \nabla V(\boldsymbol{x})^\mathsf{T} \hat{\boldsymbol{f}}(\boldsymbol{x}) + \alpha V(\boldsymbol{x}).$$

Here, $\hat{\boldsymbol{f}} : \mathbb{R}^d \to \mathbb{R}^d$ is a base dynamics model, and $\alpha \geq 0$ is a nonnegative constant. Function $V : \mathbb{R}^d \to \mathbb{R}$ works as a Lyapunov (candidate) function. V is designed so that it has a global minimum at $\boldsymbol{x} = \boldsymbol{0}$ and no local minima:

$$V(\boldsymbol{x}) = \sigma \left(q(\boldsymbol{x}) - q(\boldsymbol{x}_e) \right) + \varepsilon \|\boldsymbol{x} - \boldsymbol{x}_e\|_2^2, \tag{3}$$

where $\varepsilon > 0$ is a positive constant to ensure the positivity of V, and $\sigma : \mathbb{R} \to \mathbb{R}_{\geq 0} = [0, \infty)$ is a convex nondecreasing function with $\sigma(0) = 0$. Function $q : \mathbb{R}^d \to \mathbb{R}$ also needs to be convex, and they use the input convex neural networks (Amos, Xu, and Kolter 2017) for q.

3 Proposed Method

3.1 Stability of General Invariant Set

We begin with reviewing the theory around stability of general invariant sets, which comprises the theoretical backbone of the proposed method. First, stability of a general invariant set is formally defined as follows:

Definition 3 (Stability of invariant set). Let $S \subset \mathcal{X}$ be a positively invariant set of dynamical system (1), and let $\mathrm{dist}(\boldsymbol{x}, S) = \inf_{\boldsymbol{s} \in S} \|\boldsymbol{x} - \boldsymbol{s}\|$ denote the distance between $\boldsymbol{x}$ and S. S is said to be stable if for every $\epsilon > 0$, there exists $\delta > 0$ such that, if $\mathrm{dist}(\boldsymbol{x}(0), S) < \delta$, then $\mathrm{dist}(\boldsymbol{x}(t), S) < \epsilon$ for all $t \geq 0$. Moreover, if S is stable, and $\mathrm{dist}(\boldsymbol{x}, S) \to 0$ as $t \to \infty$, S is said to be asymptotically stable.

Stable invariant sets appear in various forms in a variety of dynamics. For example, a closed invariant orbit is called a limit cycle, and if it is asymptotically stable, nearby trajectories approach to it as $t \to \infty$. Such stable limit cycles play a key role in understanding behavior of various physical and biological phenomena with oscillation (Strogatz 2015). Moreover, invariant sets comprising infinitely many equilibria are often considered in analyzing higher-order coupled oscillators in physics (Tanaka and Aoyagi 2011) and continuous attractor networks in neuroscience (Eliasmith 2005).

The LaSalle's theorem characterizes the asymptotic stability of a general invariant set (see, e.g., Khalil 2002):

Theorem 2 (LaSalle's theorem). *Let $\Omega \subset \mathcal{D} \subset \mathbb{R}^d$ be a compact set that is positively invariant for the dynamical system* (1). *Let $V : \mathcal{D} \to \mathbb{R}$ be a differentiable function such that $\dot{V}(\boldsymbol{x}) \leq 0$ in Ω. Let $\mathcal{E} \subset \Omega$ be the set of all points in Ω such that $\dot{V}(\boldsymbol{x}) = 0$. Let $S \subset \mathcal{E}$ be the largest invariant set in $\mathcal{E}$. Then, every solution of* (1) *starting from a point in Ω approaches to S as $t \to \infty$.*

3.2 Dynamics Models with Stable Invariant Set

We give a general framework to construct a dynamics model with a general stable invariant set. This framework can be implemented with any parametric function approximators, such as neural networks, as its components. We provide concrete examples of implementation later in Section 4.

The proposed dynamics model, $\dot{\boldsymbol{x}} = \boldsymbol{f}(\boldsymbol{x})$, is depicted in Figure 1. It comprises five steps. Given a state vector $\boldsymbol{x}$ as an input, it first computes a transformed latent state $\boldsymbol{z} = \phi(\boldsymbol{x})$ via a learnable bijective function ϕ (**Step 1**). Latent state $\boldsymbol{z}$ is fed into a base dynamics model $\boldsymbol{h}$ (**Step 2**). Then, $\boldsymbol{h}(\boldsymbol{z})$ may be modified to be $\boldsymbol{g}(\boldsymbol{z})$ to ensure stability of some set $\tilde{S}$ (**Step 3**). $\boldsymbol{g}(\boldsymbol{z})$ may further be modified to be $\tilde{\boldsymbol{f}}(\boldsymbol{z})$ to ensure invariance of $\tilde{S}$ (**Step 4**). Finally, it computes $\boldsymbol{f}(\boldsymbol{x}) = \phi^{-1}(\tilde{\boldsymbol{f}}(\boldsymbol{x}))$ via the inverse of ϕ (**Step 5**). In the following, we explain the details of the five steps.

Step 1: Learnable Invertible Feature Transform Given a state vector $\boldsymbol{x} \in \mathcal{X} \subset \mathbb{R}^d$ as an input, we transform it into a latent state $\boldsymbol{z} \in \mathcal{Z} \subset \mathbb{R}^d$ using a learnable *bijective* function $\phi : \mathcal{X} \to \mathcal{Z}$, that is,

$$\boldsymbol{z} = \phi(\boldsymbol{x}). \tag{4}$$

We restrict ϕ to be bijective for *provable* existence of a stable invariant set. A bijective function ϕ can be modeled in a number of ways given the development of invertible neural networks in the area of normalizing flows (see, e.g., Papamakarios et al. 2019, and references therein). Hence, we believe that restricting ϕ to be bijective does not severely limit the flexibility of the proposed dynamics model.

Such a feature transform, together with its inverse in **Step 5**, is indispensable when we cannot exactly parametrize the geometry of a *to-be* stable invariant set (in $\mathcal{X}$) that a learned dynamics model should have. This is often the case in practice; for example, we may only know the presence of a limit cycle but cannot describe its shape analytically in advance. Our proposal lies in avoiding such a difficulty by defining a set that has a primitive shape (e.g., a unit sphere)
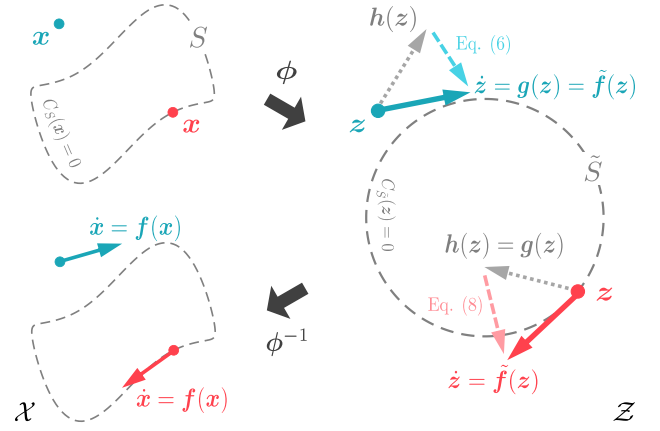

Figure 1: Proposed dynamics model, $\dot{\boldsymbol{x}} = \boldsymbol{f}(\boldsymbol{x})$, for the case where latent stable invariant set $\tilde{S}$ is defined as $\tilde{S}^{\text{surf}}$ in (7b). Two states, $\boldsymbol{x} \notin S$ (blue) and $\boldsymbol{x} \in S$ (red), are shown. The dotted orbits are the original and latent stable invariant sets, $S \subset \mathcal{X}$ and $\tilde{S} \subset \mathcal{Z}$. Input $\boldsymbol{x} \in \mathcal{X}$ is first transformed into a latent state $\boldsymbol{z} \in \mathcal{Z}$ by a learnable bijective function ϕ and then fed into a base dynamics model $\boldsymbol{h}$. $\boldsymbol{h}(\boldsymbol{z})$ is modified to ensure the stability and/or invariance of $\tilde{S}$ by (6) and/or (8), respectively. Finally, things are projected back to $\mathcal{X}$ by ϕ^{-1}.

in $\mathcal{Z}$, expecting (inverse of) ϕ should learn an appropriate transformation of the primitive shape in $\mathcal{Z}$ into a to-be stable invariant set in the original space, $\mathcal{X}$. Hereafter, a to-be stable invariant set in $\mathcal{X}$ and the corresponding primitive set in $\mathcal{Z}$ are denoted by $S \subset \mathcal{X}$ and $\tilde{S} \subset \mathcal{Z}$, respectively.

Step 2: Base Dynamics Model The second component is a base dynamics model $\boldsymbol{h} : \mathcal{Z} \to \mathcal{Z}$ that acts on the latent state $\boldsymbol{z}$. We can use any parametric models as $\boldsymbol{h}$. Note that we do not have control on the invariance and stability properties of base dynamics model $\dot{\boldsymbol{z}} = \boldsymbol{h}(\boldsymbol{z})$. Hence, we need to modify the output of $\boldsymbol{h}$ to make $\tilde{S}$ a stable invariant set.

Step 3: Ensuring Stability In this step, we modify the output of the base dynamic model, $\boldsymbol{h}$, so that $\boldsymbol{z}$'s trajectories converge to some limit set $\tilde{S} \subset \mathcal{Z}$ in $t \to \infty$. According to Theorem 2, for trajectories to converge to $\tilde{S}$, it is sufficient that there exists a function $V : \mathcal{Z} \to \mathbb{R}$ whose value decreases along trajectories everywhere outside $\tilde{S}$. To this end, we construct a candidate function for V by generalizing the method of Manek and Kolter (2019).

Suppose $\tilde{S} \subset \mathcal{Z}$ is convex. This assumption does not make us lose much generality because even if $\tilde{S}$ is convex, corresponding $S \subset \mathcal{X}$ is not necessarily convex thanks to the feature transform ϕ in **Step 1** and **Step 5**. Let $\mathbb{P}_{\tilde{S}}\boldsymbol{z}$ denote the orthogonal projection of $\boldsymbol{z}$ onto $\tilde{S}$, that is, $\mathbb{P}_{\tilde{S}}\boldsymbol{z} = \arg\min_{\boldsymbol{s} \in \tilde{S}} \|\boldsymbol{z} - \boldsymbol{s}\|_2^2$. Let $q : \mathcal{Z} \to \mathbb{R}$ be a convex function, and $\sigma : \mathbb{R} \to \mathbb{R}_{\geq 0}$ be a convex nonnegative nondecreasing function with $\sigma(0) = 0$. We define a function V as

$$V(\boldsymbol{z}) = \sigma\left(q(\boldsymbol{z}) - q(\mathbb{P}_{\tilde{S}}\boldsymbol{z})\right) + \varepsilon\|\boldsymbol{z} - \mathbb{P}_{\tilde{S}}\boldsymbol{z}\|_2^2, \tag{5}$$

with $\varepsilon > 0$. It reaches the minimum $V(\boldsymbol{z}) = 0$ at $\boldsymbol{z} \in \tilde{S}$ and

does not have any local minima at $z \notin \tilde{S}$ from construction. Given such a function (5), we modify the outputs of the base dynamics model, $h(z)$, into $g(z)$ as follows:

$$g(z) = \begin{cases} h(z), & z \in \tilde{S}, \\ h(z) - u(\beta(z)) \frac{\beta(z) + \eta(z)}{\|\nabla V(z)\|_2^2} \nabla V(z), & z \notin \tilde{S}, \end{cases} \quad (6)$$
$$\text{where} \quad \beta(z) = \nabla V(z)^\mathsf{T} h(z) + \alpha V(z).$$

Here, u is the unit step function (i.e., $u(a) = 1$ if $a \geq 0$ and $u(a) = 0$ otherwise), and $\alpha \geq 0$ is a nonnegative constant. $\eta : \mathbb{R}^d \to \mathbb{R}_{\geq 0}$ is a nonnegative function that works like a slack variable. Merely setting $\eta(z) = 0$ also ensures the stability of $\tilde{S}$, but it may be useful to make η a learnable component if we want more flexibility. Note that (5) and (6) do not ensure anything about the positive invariance property of $\tilde{S}$; it is deferred in **Step 4**.

Comparing (5) and (6) to (3) and (2), we can see that **Step 3** here is indeed a generalized version of the method of Manek and Kolter (2019). We note that such a generalization is meaningful only with other components of the proposed method; we need the learnable feature transform of **Step 1** and **Step 5** to avoid difficulty of parametrizing a stable invariant set analytically, and the procedure of **Step 4** is indispensable to ensure that trajectories do not escape from a limit set. **Step 3** does not work without these remedies.

We may compute $\mathbb{P}_{\tilde{S}}$ in a closed form when $\tilde{S}$ is a simple-shaped set like a sphere or a 2-torus. Such a simple $\tilde{S}$ does not severely drop the flexibility of a dynamics model if we set ϕ to be flexible enough. Meanwhile, we can also adopt $\tilde{S}$ with nontrivial $\mathbb{P}_{\tilde{S}}$ if needed, by employing the technique of the convex optimization layer (Agrawal et al. 2019).

Step 4: Ensuring Invariance Recall that the previous step only ensures $\tilde{S}$ is a limit set. Even if trajectories converge to $\tilde{S}$ in $t \to \infty$, they may escape unless it is also invariant. To make $\tilde{S}$ invariant, we further modify the output of g.

Without loss of generality, we consider the following two types of the definition of $\tilde{S}$:

$$\tilde{S}^{\mathrm{vol}} = \{z \mid C_{\tilde{S}}(z) \geq 0\}, \quad (7a)$$
$$\tilde{S}^{\mathrm{surf}} = \{z \mid C_{\tilde{S}}(z) = 0\}, \quad (7b)$$

where $C_{\tilde{S}} : \mathcal{Z} \to \mathbb{R}$ is a continuously differentiable function. The invariance of such sets can be characterized as follows:

Proposition 1. *For a dynamical system $\dot{z} = F(z)$ with some $F : \mathcal{Z} \to \mathcal{Z}$,*
  (a) *If $C_{\tilde{S}}(z) = 0 \Rightarrow \nabla C_{\tilde{S}}(z)^\mathsf{T} F(z) > 0$, then $\tilde{S}^{vol}$ in (7a) is a positively invariant set.*
  (b) *If $C_{\tilde{S}}(z) = 0 \Rightarrow \nabla C_{\tilde{S}}(z)^\mathsf{T} F(z) = 0$, then $\tilde{S}^{surf}$ in (7b) is a positively invariant set.*

Proof. Let $z(t)$ be a trajectory of the dynamical system, $\dot{z} = F(z)$. Let $c : \mathbb{R} \to \mathbb{R}$ be a function such that $c(\tau) = C_{\tilde{S}}(z(\tau))$. First, let us consider case (a). For a proof by contradiction, assume the negation of (a), that is, $C_{\tilde{S}}(z) = 0 \Rightarrow \nabla C_{\tilde{S}}(z)^\mathsf{T} F(z) > 0$ *and* $\tilde{S} = \tilde{S}^{\mathrm{vol}}$ is *not* a positively invariant set (i.e., $z(t) \in \tilde{S}$ and $z(s) \notin \tilde{S}$ for some

$t \leq s$). Then, from the definition of $\tilde{S}^{\mathrm{vol}}$ in (7a), we have $c(t) \geq 0$ and $c(s) < 0$. By the continuity of c, there is at least one point $r \in [t, s]$ where $c(r) = 0$ and $\dot{c}(r) \leq 0$. At this point, $C_{\tilde{S}}(z(r)) = 0$ and $\nabla C_{\tilde{S}}(z(r))^\mathsf{T} F(z(r)) = \dot{c}(r) \leq 0$, which is a contradiction to what we assumed. Therefore, (a) holds.

Second, let us see (b). Analogously to the above case, assume $C_{\tilde{S}}(z) = 0 \Rightarrow \nabla C_{\tilde{S}}(z)^\mathsf{T} F(z) = 0$ *and* $\tilde{S} = \tilde{S}^{\mathrm{surf}}$ is *not* a positively invariant set. Then, from the definition of $\tilde{S}^{\mathrm{surf}}$ in (7b), we have $c(t) = 0$ and $c(s) \neq 0$. Hence, we have $c(r) = 0$ and $\dot{c}(r) \neq 0$ at some point $r \in [t, s]$, which is a contradiction. Therefore, (b) holds. $\square$

Given this fact, we modify the outputs of the previous step, $g(z)$, into $\tilde{f}(z)$ as follows:

$$\tilde{f}(z) = \begin{cases} g(z) - \frac{\gamma(z) - \xi(z)}{\|\nabla C_{\tilde{S}}(z)\|_2^2} \nabla C_{\tilde{S}}(z), & C_{\tilde{S}}(z) = 0, \\ g(z), & C_{\tilde{S}}(z) \neq 0, \end{cases} \quad (8)$$
$$\text{where} \quad \gamma(z) = \nabla C_{\tilde{S}}(z)^\mathsf{T} g(z).$$

The definition of ξ depends on that of $\tilde{S}$; if $\tilde{S}$ is defined as $\tilde{S}^{\mathrm{vol}}$ in (7a), $\xi : \mathcal{Z} \to \mathbb{R}_{>0}$ is a positive-valued function; if $\tilde{S}$ is as $\tilde{S}^{\mathrm{surf}}$ in (7b), it is simply $\xi(z) = 0$. Note that in actual computation, condition $C_{\tilde{S}}(z) = 0$ in (8) should be replaced by $|C_{\tilde{S}}(z)| \leq \epsilon$ with a tiny ϵ.

Step 5: Projecting Back Things have been described in terms of the latent state $z \in \mathcal{Z}$ after **Step 1**. However, what we want is a dynamics model on $x \in \mathcal{X}$, namely $\dot{x} = f(x)$. As the final part of the proposed method, we project things back to $\mathcal{X}$ via the inverse of ϕ, that is,

$$f(x) = \phi^{-1}\big(\tilde{f}(z)\big) = \phi^{-1}\big(\tilde{f}(\phi(x))\big). \quad (9)$$

Recall that we assumed ϕ is invertible in **Step 1**.

3.3 Analysis

The dynamics model in (9) has a stable invariant set that can be learned from data. We summarize such a property as follows, where we describe the cases of $\cdot^{\mathrm{vol}}$ and $\cdot^{\mathrm{surf}}$ in parallel.

Proposition 2. *Let $\tilde{S}^{vol}$ (or $\tilde{S}^{surf}$) be a subset of $\mathcal{Z} \subset \mathbb{R}^d$ defined in (7a) (or (7b)). Let $\tilde{f} : \mathcal{Z} \to \mathcal{Z}$ be the function in (8). Then, for a dynamical system $\dot{z} = \tilde{f}(x)$, $\tilde{S}^{vol}$ (or $\tilde{S}^{surf}$) is a positively invariant set and is asymptotically stable.*

Proof. Let us consider the case of $\tilde{S}^{\mathrm{vol}}$ (the discussion holds analogously for $\tilde{S}^{\mathrm{surf}}$). Recall that from the definition, $C_{\tilde{S}}(z) = 0$ implies $z \in \tilde{S}$. Hence, from (8), if $C_{\tilde{S}}(z) = 0$, then $\nabla C_{\tilde{S}}(z)^\mathsf{T} \tilde{f}(z) = \xi(z) > 0$, which proves the invariance of $\tilde{S}$ (Proposition 1). As for stability, we should show

$$\begin{aligned} \dot{V}(z) &= 0, \quad \text{if and only if} \quad z \in \tilde{S}, \\ \dot{V}(z) &< 0, \quad \text{otherwise.} \end{aligned} \quad (10)$$

Suppose $z \in \tilde{S}$. We have $V(z) = 0$ for every $z \in \tilde{S}$ from the construction, and the orbits of $\tilde{f}$ stay in $\tilde{S}$ because $\tilde{S}$ is a

positively invariant set. Hence, $\dot{V}(\boldsymbol{z}) = 0$ for $\boldsymbol{z} \in \tilde{S}$. On the other hand, suppose $\boldsymbol{z} \notin \tilde{S}$. Then, we have

$$
\begin{aligned}
&\nabla V(\boldsymbol{z})^{\mathsf{T}} \tilde{\boldsymbol{f}}(\boldsymbol{z}) + \alpha V(\boldsymbol{z}) \\
&\quad = \nabla V(\boldsymbol{z})^{\mathsf{T}} \boldsymbol{g}(\boldsymbol{z}) + \alpha V(\boldsymbol{z}) \\
&\quad = \nabla V(\boldsymbol{z})^{\mathsf{T}} \boldsymbol{h}(\boldsymbol{z}) + \alpha V(\boldsymbol{z}) - u\big(\nabla V(\boldsymbol{z})^{\mathsf{T}} \boldsymbol{h}(\boldsymbol{z}) + \alpha V(\boldsymbol{z})\big) \\
&\qquad\qquad \cdot \big(\nabla V(\boldsymbol{z})^{\mathsf{T}} \boldsymbol{h}(\boldsymbol{z}) + \alpha V(\boldsymbol{z}) + \eta(\boldsymbol{z})\big) \\
&\quad = \beta(\boldsymbol{z}) - u\big(\beta(\boldsymbol{z})\big)\big(\beta(\boldsymbol{z}) + \eta(\boldsymbol{z})\big).
\end{aligned}
$$

First, suppose $\beta(\boldsymbol{z}) \geq 0$. Then $u(\beta(\boldsymbol{z})) = 1$, and thus $\nabla V(\boldsymbol{z})^{\mathsf{T}} \tilde{\boldsymbol{f}}(\boldsymbol{z}) + \alpha V(\boldsymbol{z}) = -\eta(\boldsymbol{z}) \leq 0$. Second, suppose $\beta(\boldsymbol{z}) < 0$. Then $u(\beta(\boldsymbol{z})) = 0$, and thus $\nabla V(\boldsymbol{z})^{\mathsf{T}} \tilde{\boldsymbol{f}}(\boldsymbol{z}) + \alpha V(\boldsymbol{z}) = \beta(\boldsymbol{z}) < 0$. As $V(\boldsymbol{z}) > 0$ at $\boldsymbol{z} \notin \tilde{S}$ from the construction of V, in either of the cases above, we have $\nabla V(\boldsymbol{z})^{\mathsf{T}} \tilde{\boldsymbol{f}}(\boldsymbol{z}) \leq -\alpha V(\boldsymbol{z}) < 0$. Hence, $\dot{V}(\boldsymbol{z}) < 0$ for all $\boldsymbol{z} \notin \tilde{S}$. This proves (10), from which we can say that $\tilde{S}$ is the largest subset of the state space such that $\dot{V}(\boldsymbol{z}) = 0$. Therefore, from Theorem 2, $\tilde{S}$ is asymptotically stable. $\square$

Corollary 1. *Suppose a subset of $\mathcal{X}$, namely $S^{vol} = \{\boldsymbol{x} \mid C_{\tilde{S}}(\phi(\boldsymbol{x})) \geq 0\}$ (or $S^{surf} = \{\boldsymbol{x} \mid C_{\tilde{S}}(\phi(\boldsymbol{x})) = 0\}$), where $C_{\tilde{S}}$ is the function that appeared in (7). Let $\boldsymbol{f} = \phi^{-1} \circ \tilde{\boldsymbol{f}} \circ \phi$ as in (9). Then, S^{vol} (or S^{surf}) is an asymptotically stable invariant set of a dynamical system defined as $\dot{\boldsymbol{x}} = \boldsymbol{f}(\boldsymbol{x})$.*

3.4 Extension

The proposed dynamics model in Section 3.2 works not only as a standalone machine learning model, but also as a module embedded in a larger machine learning method. For example, suppose we have high-dimensional observations $\boldsymbol{y} \in \mathcal{Y}$ (e.g., observation of fluid flow). In such a case, we often try to transform $\boldsymbol{y}$ into lower-dimensional vectors, namely $\boldsymbol{x}$, using methods like principal component analysis and autoencoders. We can then consider a dynamics model (with a stable invariant set) on $\boldsymbol{x}$ as in Section 3.2, rather than on $\boldsymbol{y}$ directly. Temporal forecasting is performed in the space of $\boldsymbol{x}$ and then returned to the space of $\boldsymbol{y}$.

Such an extension is straightforward yet useful, but a drawback is that if the dimensionality reduction is lossy, which is often the case, we can no longer guarantee a stable invariant set in $\mathcal{Y}$. Nonetheless, such an approximative model may still be useful. We exemplify such a case in Section 6.4, where we reduce the dimensionality of fluid flow observations by principal component analysis and learn a dynamics model on the low-dimensional space.

4 Implementation Examples

4.1 Learnable Components

The components of the proposed method, namely ϕ, $\boldsymbol{h}$, q, η, and ξ, can be any parametric models such as neural networks. Let us introduce examples for each component.

ϕ Choice of ϕ's model depends on the availability of prior knowledge of the dynamics to be learned (see also Table 1). For example, if we know the topology of S (e.g., it is a closed orbit), we can model ϕ as a diffeomorphic function such as the neural ODE (NODE) (Chen et al. 2018) and

coupling flows (see, e.g., Teshima et al. 2020). In fact, such prior knowledge is often available from our scientific understanding of physical, chemical, and biological phenomena (see, e.g., Strogatz 2015; Tanaka and Aoyagi 2011; Eliasmith 2005). We may use other types of invertible models if less prior knowledge is available; for example, a neural ODE with auxiliary variables (namely, ANODE) (Dupont, Doucet, and Teh 2019) can represent non-homeomorphic functions. If perfect prior knowledge is available (i.e., we can specify the geometry of $S \subset \mathcal{X}$ analytically), simply set $\boldsymbol{z} = \phi(\boldsymbol{x}) = \boldsymbol{x}$.

$\boldsymbol{h}$ We can substitute arbitrary models to the base dynamics, $\boldsymbol{h}$, in accordance with the nature of data.

q The convex function, q in (5), can be modeled using the input-convex neural networks (Amos, Xu, and Kolter 2017) as in the previous work (Manek and Kolter 2019).

η **and** ξ The slack-like functions, η and ξ in (6) and (8), respectively, can be modeled as neural networks with output-value clipped to be nonnegative or positive.

4.2 Stable Invariant Set

Besides the learnable components, we should prepare a to-be stable invariant set. If we do not know the analytic form of $S \subset \mathcal{X}$ (i.e., C_S), which is usually the case, we are to define $\tilde{S} \subset \mathcal{Z}$ (i.e., $C_{\tilde{S}}$) instead. A general guideline we suggest is to set $\tilde{S}$ as a simple primitive shape, such as spheres and tori. As stated earlier, setting $\tilde{S}$ to be a primitive shape does not severely restrict the flexibility of the model, thanks to the learnable feature transform, ϕ, which "deforms" a simple $\tilde{S}$ to various S. We can regard unknown coefficients in $C_{\tilde{S}}$ (e.g., radius of sphere) as learnable parameters, too.

We can also consider the case of low-dimensional S by setting $\tilde{S}$ also low-dimensional. Here, axes of $\mathcal{Z}$ ignored by a low-dimensional $\tilde{S}$ can be arbitrary because the feature transform ϕ modeled by a neural network is usually flexible enough to learn a rotation between $\mathcal{Z}$ and $\mathcal{X}$.

Care may have to be taken in the computation of $\mathbb{P}_{\tilde{S}}$. If we do not know a closed form of $\mathbb{P}_{\tilde{S}} \boldsymbol{z}$, as long as $\tilde{S}$ is convex as we assumed, we can use the differentiable convex optimization layer (Agrawal et al. 2019) to allow gradient-based optimization. For example, suppose we have $\tilde{S}$ of the type of (7a). Then, $\mathbb{P}_{\tilde{S}}$ is an optimization problem:

$$
\mathbb{P}_{\tilde{S}} \boldsymbol{z} = \arg\min_{\boldsymbol{s}} \|\boldsymbol{z} - \boldsymbol{s}\|_2^2 \quad \text{s.t.} \quad C_{\tilde{S}}(\boldsymbol{s}) \geq 0. \quad (11)
$$

The derivative of its output (i.e., $\partial \mathbb{P}_{\tilde{S}} \boldsymbol{z} / \partial \boldsymbol{z}$) can be computed by the techniques of Agrawal et al. (2019) via the implicit function theorem on the optimality condition of (11).

Examples Let us provide concrete examples of the configuration of ϕ and $C_{\tilde{S}}$ (also summarized in Table 1).

Example 1. If we exactly know that S is a sphere around the origin, we can set ϕ to be the identity function, i.e., $\boldsymbol{z} = \phi(\boldsymbol{x}) = \boldsymbol{x}$, and set $C_{\tilde{S}}(\boldsymbol{z}) = \|\boldsymbol{z}\|^2 - r^2$. In this case, $\mathbb{P}_{\tilde{S}} \boldsymbol{z} = \mathbb{P}_S \boldsymbol{x} = r\boldsymbol{x}/\|\boldsymbol{x}\|$ for $\boldsymbol{x} \neq \boldsymbol{0}$ (and arbitrary for $\boldsymbol{x} = \boldsymbol{0}$). Radius r may or may not be a learnable parameter.

Example 2. If we know the dynamics should have a stable limit cycle, we can set S to be a circle along a pair of axes of $\mathcal{Z}$, expecting ϕ learns an appropriate coordinate transform

	$\phi(\boldsymbol{x})$	$C_{\tilde{S}}(\boldsymbol{z})$
Perfect knowledge available (i.e., exact $S \subset \mathcal{X}$ is known)	identity	$\tilde{S} = S$ (cf. Example 1)
Partial knowledge available (i.e., rough behavior of phenomenon is known)		
e.g., self-sustained oscillations	NODE	Example 2
e.g., quasiperiodic patterns	NODE	Example 3
e.g., neural integrators	(A)NODE	Example 4

Table 1: Implementation examples in accordance with availability of prior knowledge. NODE (Chen et al. 2018) and ANODE (Dupont, Doucet, and Teh 2019) are mentioned here, but other invertible neural nets are applicable, too. For concrete examples of each case of "partial knowledge," e.g., Strogatz (2015) and Eliasmith (2005) are informative.

to adjust the axes to those in the original space. For example, $C_{\tilde{S}}(\boldsymbol{z}) = z_1^2 + z_2^2 - r^2$ (and ignore z_i's for $i > 2$).

Example 3. We may set $\tilde{S}$ to be a 2-torus, $C_{\tilde{S}}(\boldsymbol{z}) = (\sqrt{z_1^2 + z_2^2} - R)^2 + z_3^2 - r^2$, onto which the orthogonal projection $\mathbb{P}_{\tilde{S}}\boldsymbol{z}$ can be computed analytically.

Example 4. Another common option is a hyperplane $C_{\tilde{S}}(\boldsymbol{z}) = \boldsymbol{c}^\mathsf{T}\boldsymbol{z} - b$. This is useful in modeling, for example, sets of infinitely many equilibria, which often appear in computational neuroscience (Eliasmith 2005).

Example 5. More generally, we may set $\tilde{S}$ to be a quadric, $C_{\tilde{S}}(\boldsymbol{z}) = \boldsymbol{z}^\mathsf{T}\boldsymbol{Q}\boldsymbol{z} + \boldsymbol{p}^\mathsf{T}\boldsymbol{z} + r$. In this case, we need the differentiable optimization layer (Agrawal et al. 2019).

4.3 Learning Procedures

Given a dataset and a dynamics model $\dot{\boldsymbol{x}} = \boldsymbol{f}(\boldsymbol{x})$ constructed as above, we are to learn the parameters of the unknown functions, ϕ, $\boldsymbol{h}$, and q, and possibly $\boldsymbol{\eta}$, $\boldsymbol{\xi}$, and C. The learning scheme can be designed in either or both of the following two ways. First, if we have paired observations of $\boldsymbol{x}$ and $\dot{\boldsymbol{x}}$, we simply minimize some loss (e.g., square loss) between $\dot{\boldsymbol{x}}$ and $\boldsymbol{f}(\boldsymbol{x})$. This is also applicable when we can estimate $\dot{\boldsymbol{x}}$ from $\boldsymbol{x}$'s (e.g., Chartrand 2011). Second, if we have unevenly-sampled sequences $(\boldsymbol{x}_{t_1}, \ldots, \boldsymbol{x}_{t_n})$, we utilize the adjoint state method or backpropagation in forward ODE solvers for optimization (see, e.g., Chen et al. 2018).

5 Related Work

Learning Stable Dynamics   Learning stable *linear* dynamical systems, e.g., $\boldsymbol{x}_{t+1} = \boldsymbol{A}\boldsymbol{x}_t$ s.t. $\rho(\boldsymbol{A}) < 1$, is indeed a nontrivial problem and has been addressed for decades (e.g., Lacy and Bernstein 2003; Siddiqi, Boots, and Gordon 2008; Huang et al. 2016; Mamakoukas, Xherija, and Murphey 2020). The problem of learning stable nonlinear systems has also been studied for various models, such as Gaussian mixtures (Khansari-Zadeh and Billard 2011; Blocher, Saveriano, and Lee 2017; Umlauft and Hirche 2017), kernel methods (Khosravi and Smith 2021), Gaussian processes (Duncker et al. 2019), and neural networks (Neumann, Lemme, and

Steil 2013; Manek and Kolter 2019; Tuor, Drgona, and Vrabie 2020; Massaroli et al. 2020). However, they only handle the stability of finite number of equilibria.

We note that, in parallel to the current work, Urain et al. (2020) developed a method for learning a dynamics model as an invertible transform from a primitive model for which asymptotic behavior is specified. Their method requires to specify a particular primitive model with desired stability property. In contrast, our method is more general while it might need more meticulous attention in model configuration.

Learning Stabilizing Controllers   Another related direction is to learn a controller that stabilizes a given dynamical system. For example, Chang, Roohi, and Gao (2019) proposed a method to learn neural controllers by constructing a neural Lyapunov function simultaneously. They adopt a self-supervised learning scheme where a neural controller and a neural Lyapunov function are trained so that the violation of the stability condition (in Theorem 1) is minimized. Such an approach is also applicable to dynamics learning, but existing methods only focus on discrete equilibria, too.

Learning Physically Meaningful Systems   Another related thread of studies is to learn physical systems, such as Lagrangian (Lutter, Ritter, and Peters 2019; Cranmer et al. 2020) and Hamiltonian (Greydanus, Dzamba, and Yosinski 2019; Matsubara, Ishikawa, and Yaguchi 2020) mechanics using neural networks. Extension to port-Hamiltonian systems (Zhong, Dey, and Chakraborty 2020) is also considered.

6 Experiment

6.1 Configuration

Implementation   We implemented the learnable components (i.e., ϕ, $\boldsymbol{h}$, q, η, and ξ) with neural networks. We used ANODE (Dupont, Doucet, and Teh 2019) for ϕ in Sections 6.3 and 6.4 to allow much flexibility, while non-augmented NODE (Chen et al. 2018) was also sufficient. For the other components, we used networks with fully-connected hidden layers. We used the exponential linear unit as the activation function. Other details are found in the appendix.

Baselines   Besides the proposed model in (9), we tried either or both of the following models as baselines:

1) Base dynamics model *without stability nor invariance*, i.e., $\dot{\boldsymbol{x}} = \phi^{-1}(\boldsymbol{h}(\phi(\boldsymbol{x})))$; we may refer to this baseline as a **vanilla model**.

2) Stable dynamics model like ours, but *the stable invariant set is fixed to be an equilibrium* at $\boldsymbol{x} = 0$ (i.e., almost the same with Manek and Kolter (2019)); we may refer to this baseline as a **stable equilibrium model**.

6.2 Simple Examples

As a proof of concept, we examined the performance of the proposed method on simple dynamical systems whose stable invariant set is known analytically. Hence, we do not need ϕ (i.e., set $\phi(\boldsymbol{x}) = \boldsymbol{x}$) in the two experiments in this section.

Limit Cycle   We examined the system with a limit cycle:

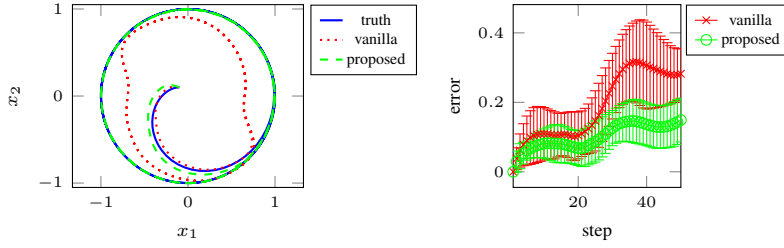$$\dot{x}_1 = x_1 - x_2 - x_1(x_1^2 + x_2^2), \quad \dot{x}_2 = x_1 + x_2 - x_2(x_1^2 + x_2^2),$$

Figure 2: Test results on the system with limit cycle in Section 6.2. (*left*) Examples of long-term prediction from $x_1, x_2 = -.1, .1$ for 200 steps. (*right*) Average (and stdev) long-term prediction errors against prediction steps.
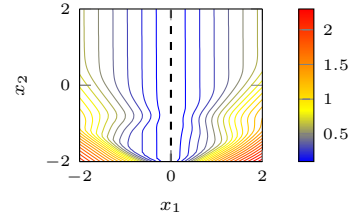

Figure 3: Contour plot of $V(\boldsymbol{x})$ learned on the data generated from the system with line attractor in Section 6.2. The dotted line is the true line attractor, $x_1 = 0$.
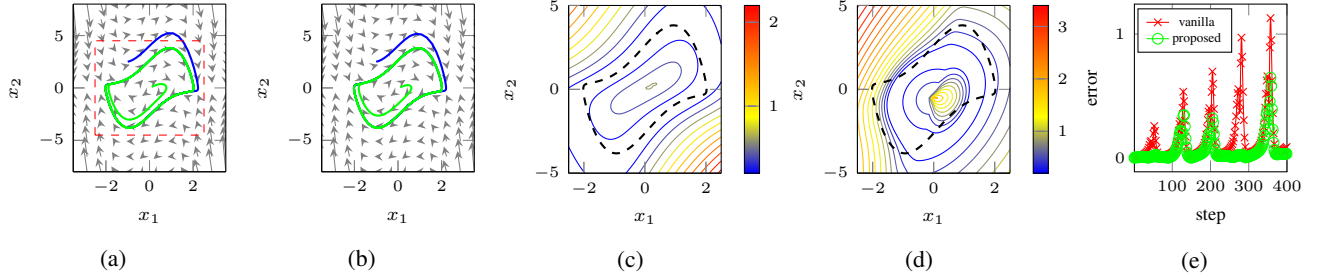

(a)　(b)　(c)　(d)　(e)

Figure 4: Results of Section 6.3. (a) True vector field of (12) and two trajectories. Red dashed-line rectangle is the training data region. (b) Learned vector field and trajectories from it. (c) Learned $V(\boldsymbol{x})$. (d) Learned $V(\boldsymbol{x})$ without $\boldsymbol{\phi}$. (e) Prediction errors.

whose orbits approach to a unit circle as $t \to \infty$. We generated four sequences of length 20 with $\Delta t = .075$ and used the pairs of $\boldsymbol{x}$ and $\dot{\boldsymbol{x}}$ as training data. For the proposed model, we set $C_S(\boldsymbol{x}) = x_1^2 + x_2^2 - 1$ (i.e., the truth) with S defined as in (7b).

In Figure 2, we show the results of long-term prediction given only $\boldsymbol{x}_{t=0}$ that was *not in the training data*. The left panel depicts trajectories of length 200 predicted by the true system, the vanilla model, and the proposed model. The proposed model's trajectory successfully reaches the plausible limit cycle, while it is not surprising as a natural consequence of the model's construction. The right panel shows the average long-term prediction errors against prediction steps (a single step corresponds to the Δt). The average was taken with regard to 20 test sequences with different $\boldsymbol{x}_{t=0}$. We can observe that the proposed stable model achieves consistently lower prediction errors.

Line Attractor　We examined another simple system:

$$\dot{x}_1 = x_1(1 - x_2), \quad \dot{x}_2 = x_1^2.$$

Line $x_1 = 0$ constitutes a line attractor of this system as a set of infinitely many stable equilibria; every orbit starting at $x_1 \neq 0$ approaches to some point on this line as $t \to \infty$. We generated eight sequences of length 80 with $\Delta t = .05$ as training data. We learned the proposed model with $\boldsymbol{\phi}(\boldsymbol{x}) = \boldsymbol{x}$ and $C_S(\boldsymbol{x}) = c_1 x_1 + c_2 x_2$, where c_1 and c_2 were learnable parameters, and S was defined as in (7b).

In Figure 3, we show the values of learned $V(\boldsymbol{x})$. We can say it is successfully learned because $V(\boldsymbol{x})$ monotonically decreases toward the line $x_1 = 0$. Moreover, it reflects the fact that a state of this system moves faster when $|x_1| \gg 0$ and $x_2 \ll 0$ (i.e., in the lower part of the $\boldsymbol{x}$-plane).

6.3 Learning Vector Field of Nonlinear Oscillator

The Van der Pol oscillator:

$$\dot{x}_1 = x_2, \quad \dot{x}_2 = \mu(1 - x_1^2)x_2 - x_1 \tag{12}$$

is well known as a basis for modeling many physical and biological phenomena. It has a stable limit cycle, whose exact shape cannot be described analytically. As training data (Figure 4a), we used the values of $\boldsymbol{x}$ and $\dot{\boldsymbol{x}}$ sampled from an even grid on area $[-2.5, 2.5] \times [-4.5, 4.5]$ with $\mu = 2$. For the proposed model, we set $\tilde{S}$ to be a circle defined as in (7b), expecting $\boldsymbol{\phi}$ would be learned so that it transforms a circle into the limit cycle of the system.

In Figure 4b, we show the learned vector field and two trajectories generated from it. They successfully resemble the truth (in Figure 4a), even outside the area of the training data. In Figure 4c, we depict the values of learned $V(\boldsymbol{x})$, wherein we can observe $V(\boldsymbol{x})$ decreases toward the limit cycle of the system (the dashed orbit). In Figure 4d, for comparison, we show learned $V(\boldsymbol{x})$ *without* a learnable feature transform $\boldsymbol{\phi}$; not surprisingly, it fails to capture the shape of the limit cycle. In Figure 4e, we show an example of long-term prediction errors against prediction steps (a single step corresponds to the Δt). The model with the proposed stability guarantee achieves significantly lower long-term prediction errors.

6.4 Application: Fluid Flow Prediction

We apply the proposed stable dynamics model to an application of fluid flow prediction. The target flow is so-called cylinder wake (see Figure 5a); a cylinder-like object is located in a 2D field, fluids come from one side uniformly, and there occurs a series of vortices past the object in certain conditions. This is a limit cycle known as the Kármán's vortex
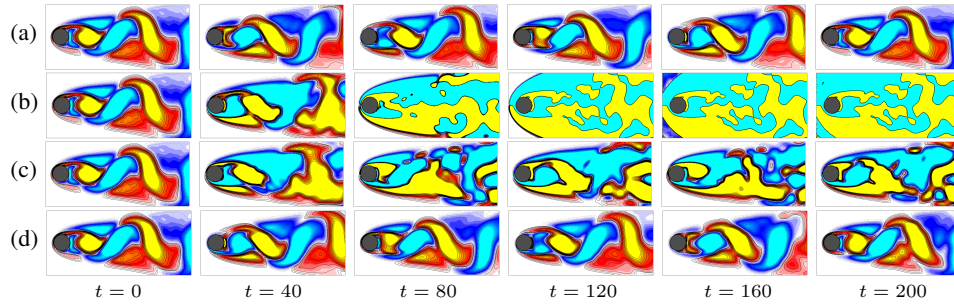
Figure 5: Long-term predictions of fluid flow. Red&yellow and blue&cyan denote positive and negative values of vorticity, respectively. (a) Ground truth. (b) The vanilla model. (c) The stable equilibrium model. (d) The proposed model.

street. Before the flow reaches the limit cycle, it typically starts from an unstable equilibrium, and then the vortices begin to grow gradually. This stage is called off-attractor. Cylinder wake has been studied as one of standard problems of fluid dynamics and also appears as a testbed of forecasting methods even recently (see, e.g., Kutz et al. 2016).

As training data, we generated such flow using the immersed boundary projection method (Taira and Colonius 2007; Colonius and Taira 2008) and used the part from near the equilibrium to a time point *before* the limit cycle is completely observed; hence the training data were off-attractor. The data comprised the observations of the vorticity in the field of size 199×449. As preprocessing, we reduced the dimensionality of data from 89351 to 26 by PCA, which lost only 0.1% of the energy. We contaminated the data with Gaussian noise. We estimated $\dot{x}$ by $(x_{t+\Delta t} - x_t)/\Delta t$ and learned the proposed dynamics model with a cycle along the first two axes of $\mathcal{Z}$ as $\tilde{S}$ (i.e., $C_{\tilde{S}}(z) = z_1^2 + z_2^2 - r^2$).

In Figure 5, we show the results of long-term prediction starting at a time point where the flow is almost on the limit cycle (i.e., on-attractor). The two baselines (in Figures 5b and 5c) fail to replicate the true limit cycle (in Figure 5a). In contrast, the long-term prediction by the proposed method (in Figure 5d) shows a plausible oscillating pattern, though the oscillation phase is slightly different from the truth. It is worth noting that with the proposed stable dynamics model, we were able to predict the on-attractor oscillating patterns only from off-attractor training data.

7   Conclusion

We proposed a dynamics model with the provable existence of a stable invariant set. It can handle the stability of general types of invariant sets, for example, limit cycles and line attractors. Future directions of research include the treatment of random dynamical systems as the current method is limited to deterministic dynamics. Consideration of the input-to-state stability of controlled systems is also an interesting problem.

Acknowledgements

References

Agrawal, A.; Amos, B.; Barratt, S.; Boyd, S.; Diamond, S.; and Kolter, J. Z. 2019. differentiable convex optimization layers. In *Advances in Neural Information Processing Systems 32*, 9562–9574.

Amos, B.; Xu, L.; and Kolter, J. Z. 2017. Input convex neural networks. In *Proceedings of the 34th International Conference on Machine Learning*, 146–155.

Blocher, C.; Saveriano, M.; and Lee, D. 2017. Learning stable dynamical systems using contraction theory. In *Proceedings of the 14th International Conference on Ubiquitous Robots and Ambient Intelligence*, 124–129.

Chang, Y.-C.; Roohi, N.; and Gao, S. 2019. Neural Lyapunov control. In *Advances in Neural Information Processing Systems 32*, 3245–3254.

Chartrand, R. 2011. Numerical differentiation of noisy, nonsmooth data. *ISRN Applied Mathematics* 2011: 164564.

Chen, T. Q.; Rubanova, Y.; Bettencourt, J.; and Duvenaud, D. K. 2018. Neural ordinary differential equations. In *Advances in Neural Information Processing Systems 31*, 6572–6583.

Colonius, T.; and Taira, K. 2008. A fast immersed boundary method using a nullspace approach and multi-domain farfield boundary conditions. *Computer Methods in Applied Mechanics and Engineering* 197(25): 2131–2146.

Costello, Z.; and Martin, H. G. 2018. A machine learning approach to predict metabolic pathway dynamics from timeseries multiomics data. *npj Systems Biology and Applications* 4(1): 19.

Cranmer, M.; Greydanus, S.; Hoyer, S.; Battaglia, P.; Spergel, D.; and Ho, S. 2020. Lagrangian neural networks. arXiv:2003.04630.

Duncker, L.; Bohner, G.; Boussard, J.; and Sahani, M. 2019. Learning interpretable continuous-time models of latent stochastic dynamical systems. In *Proceedings of the 36th International Conference on Machine Learning*, 1726–1734.

Dupont, E.; Doucet, A.; and Teh, Y. W. 2019. Augmented neural ODEs. In *Advances in Neural Information Processing Systems 32*, 3140–3150.

Eliasmith, C. 2005. A unified approach to building and controlling spiking attractor networks. *Neural Computation* 17(6): 1276–1314.

Giesl, P.; and Hafstein, S. 2015. Review on computational methods for Lyapunov functions. *Discrete and Continuous Dynamical Systems Series B* 20(8): 2291–2331.

Greydanus, S.; Dzamba, M.; and Yosinski, J. 2019. Hamiltonian neural networks. In *Advances in Neural Information Processing Systems 32*, 15379–15389.

Hirsch, M. W.; Smale, S.; and Devaney, R. L. 2003. *Differential equations, dynamical systems, and an introduction to chaos*. Academic Press, 2nd edition.

Huang, W.; Cao, L.; Sun, F.; Zhao, D.; Liu, H.; and Yu, S. 2016. Learning stable linear dynamical systems with the weighted least square method. In *Proceedings of the 25th International Joint Conference on Artificial Intelligence*, 1599–1605.

Khalil, H. K. 2002. *Nonlinear systems*. Prentice Hall, 3rd edition.

Khansari-Zadeh, S. M.; and Billard, A. 2011. Learning stable nonlinear dynamical systems with Gaussian mixture models. *IEEE Transactions on Robotics* 27(5): 943–957.

Khosravi, M.; and Smith, R. S. 2021. Nonlinear system identification with prior knowledge of the region of attraction. *IEEE Control Systems Letters* 5(3): 1091–1096.

Kutz, J. N.; Brunton, S. L.; Brunton, B. W.; and Proctor, J. L. 2016. *Dynamic mode decomposition: Data-driven modeling of complex systems*. SIAM.

Lacy, S. L.; and Bernstein, D. S. 2003. Subspace identification with guaranteed stability using constrained optimization. *IEEE Transactions on Automatic Control* 48(7): 1259–1263.

Li, Z.; Kermode, J. R.; and De Vita, A. 2015. Molecular dynamics with on-the-fly machine learning of quantum-mechanical forces. *Physical Review Letters* 114(9): 096405.

Lusch, B.; Kutz, J. N.; and Brunton, S. L. 2018. Deep learning for universal linear embeddings of nonlinear dynamics. *Nature Communications* 9(1): 4950.

Lutter, M.; Ritter, C.; and Peters, J. 2019. Deep Lagrangian networks: Using physics as model prior for deep learning. In *Proceedings of the 7th International Conference on Learning Representations*.

Mamakoukas, G.; Xherija, O.; and Murphey, T. D. 2020. Memory-efficient learning of stable linear dynamical systems for prediction and control. In *Advances in Neural Information Processing Systems 33*.

Manek, G.; and Kolter, J. Z. 2019. Learning stable deep dynamics models. In *Advances in Neural Information Processing Systems 32*, 11128–11136.

Massaroli, S.; Poli, M.; Bin, M.; Park, J.; Yamashita, A.; and Asama, H. 2020. Stable neural flows. arXiv:2003.08063.

Matsubara, T.; Ishikawa, A.; and Yaguchi, T. 2020. Deep energy-based modeling of discrete-time physics. In *Advances in Neural Information Processing Systems 33*.

Morton, J.; Jameson, A.; Kochenderfer, M. J.; and Witherden, F. 2018. Deep dynamical modeling and control of unsteady fluid flows. In *Advances in Neural Information Processing Systems 31*, 9258–9268.

Neumann, K.; Lemme, A.; and Steil, J. J. 2013. Neural learning of stable dynamical systems based on data-driven Lyapunov candidates. In *Proceedings of the 2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 1216–1222.

Papamakarios, G.; Nalisnick, E.; Rezende, D. J.; Mohamed, S.; and Lakshminarayanan, B. 2019. Normalizing flows for probabilistic modeling and inference. arXiv:1912.02762.

Raissi, M.; Perdikaris, P.; and Karniadakis, G. E. 2019. Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. *Journal of Computational Physics* 378: 686–707.

Siddiqi, S. M.; Boots, B.; and Gordon, G. J. 2008. A constraint generation approach to learning stable linear dynamical systems. In *Advances in Neural Information Processing Systems 20*, 1329–1336.

Strogatz, S. H. 2015. *Nonlinear dynamics and chaos: With applications to physics, biology, chemistry, and engineering*. CRC Press, 2nd edition.

Taira, K.; and Colonius, T. 2007. The immersed boundary method: A projection approach. *Journal of Computational Physics* 225(2): 2118–2137.

Takeishi, N.; Kawahara, Y.; and Yairi, T. 2017. Learning Koopman invariant subspaces for dynamic mode decomposition. In *Advances in Neural Information Processing Systems 30*, 1130–1140.

Tanaka, T.; and Aoyagi, T. 2011. Multistable attractors in a network of phase oscillators with three-body interactions. *Physical Review Letters* 106(22): 224101.

Teshima, T.; Ishikawa, I.; Tojo, K.; Oono, K.; Ikeda, M.; and Sugiyama, M. 2020. Coupling-based invertible neural networks are universal diffeomorphism approximators. In *Advances in Neural Information Processing Systems 33*.

Tuor, A.; Drgona, J.; and Vrabie, D. 2020. Constrained neural ordinary differential equations with stability guarantees. arXiv:2004.10883.

Umlauft, J.; and Hirche, S. 2017. Learning stable stochastic nonlinear dynamical systems. In *Proceedings of the 34th International Conference on Machine Learning*, 3502–3510.

Urain, J.; Ginesi, M.; Tateo, D.; and Peters, J. 2020. ImitationFlow: Learning deep stable stochastic dynamic systems by normalizing flows. In *Proceedings of the 2020 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 5231–5237.

Wang, J. M.; Fleet, D. J.; and Hertzmann, A. 2006. Gaussian process dynamical models. In *Advances in Neural Information Processing Systems 18*, 1441–1448.

Zhong, Y. D.; Dey, B.; and Chakraborty, A. 2020. Dissipative SymODEN: Encoding Hamiltonian dynamics with dissipation and control into deep learning. arXiv:2002.08860.