2024 IEEE 63rd Conference on Decision and Control (CDC)
December 16-19, 2024. MiCo, Milan, Italy

Learning Stable and Passive Neural Differential Equations

Jing Cheng, Ruigang Wang and Ian R. Manchester

***Abstract*— In this paper, we introduce a novel class of neural differential equation, which are intrinsically Lyapunov stable, exponentially stable or passive. We take a recently proposed Polyak Lojasiewicz network (PLNet) as an Lyapunov function and then parameterize the vector field as the descent directions of the Lyapunov function. The resulting models have a same structure as the general Hamiltonian dynamics, where the Hamiltonian is lower- and upper-bounded by quadratic functions. Moreover, it is also positive definite w.r.t. either a known or learnable equilibrium. We illustrate the effectiveness of the proposed model on a damped double pendulum system.**

I. Introduction

Neural networks have demonstrated great power in machine learning and control, particularly on learning the dynamics and forecasting the behavior of dynamical systems [1], [2]. When approximating the dynamic behaviour, preserving some essential properties especially stability and passivity, is favored by learning and control community. Enforcing stability can benefit the learnt models especially in terms of generalization.

For nonlinear systems, enforcing stability during learning has been studied with Gaussian mixture models and polynomial models in [3], [4], [5] and even in the case of linear systems it is non-trivial [6]. For nonlinear systems, there are various of stability notions with different implications. In the context of learning, a strong stability notion called *contraction* [7] (any pair of trajectories converge to each other) has recently received much attention due to its equilibrium-independent stability nature. For discrete-time setup, [8], [9], [10] have developed contracting, incrementally passive and dissipative neural dynamics. A continuous-time counterpart can be found in [11]. A benefit of [9], [11] is their direct (i.e. unconstrainted & surjective) parameterization of stable models, making training easy. However a limitation is that they enforce contraction with respect to a state-independent quadratic metric, limiting flexibility.

For learning dynamical systems with weak stability properties (e.g., Lyapunov stability w.r.t. a particular equilibrium), it is often desired to apply models that preserve the similar stability property. A key component for stable neural differential equations is the neural Lyapunov function. From [12] and the resolution of the Poincare Conjecture by Perelman [13], all Lyapunov functions have level sets homeomorphic to the unit ball. This suggests searching for candidate Lyapunov functions of the form $V(x) = g(x)^\top g(x)$ where g is a neural network, e.g. multi-layer perception (MLP). But for general MLP, $V(x)$ may not be positive definite or radially unbounded. In [14], a special parameterization for g (i.e., each layer has a larger dimension than the previous layer) is proposed in [14] such that $V(x)$ is positive definite. The drawback is that it may restrict the model flexibility. The work [15] proposed several neural Lyapunov functions built on the input convex neural network (ICNN) [16]. To be specific, given an ICNN $\phi(x)$ where ϕ is a scalar-output network which is convex w.r.t. the input x, the Lyapunov function is constructed as $V(x) = \phi(x)$ or $V(x) = \phi(x) + \epsilon|x|^2$. Although it can guarantee a lower quadratic bound, it is still unknown how to impose an upper quadratic bound. A more recent survey on neural Lyapunov functions can be found in [17].

Given an Lyapunov function $V(x)$, one needs to construct a vector field $f(x)$ such that the Lyapunov inequality $\nabla^\top V(x) f(x) < 0$ holds for nonzero x. An intuitive approach is the gradient flow, i.e., $f(x) = -\alpha(x)\nabla V(x)$ with $\alpha(x) > 0$. However, it is often restrictive as there are infinitely many of descent directions. In [15], a projection method is applied to resolve the case where Lyapunov inequality is infeasible. There are two main drawbacks: 1) the vector field becomes non-smooth due to projection; 2) when projection is activated, the vector field is reduced to a gradient flow of a convex function.

To address the above issues, we take a recently developed bi-Lipschitz network g from [18] to construct a Lyapunov function $H(x) = 0.5|g(x)|^2$. Since g is bi-Lipschitz, then it is also invertible, implying that the level set of H is homeomorphic to a unit ball. Unlike the ICNN-based Lyapunov function, the level sets of H can be non-convex. Thanks to the bi-Lipschitz property of g, both quadratic lower and upper bounds are guaranteed automatically. We then construct the vector field as $f(x) = [J(x) - R(x)]\nabla H(x)$ where J is skew symmetric and R is positive semidefinite. The above f is a smooth field of the descent vectors of H. By parameterizing J and R using neural networks, one can learn flexible stable dynamics. Moreover, this structure can also be realized as the Hamiltonian form that widely exists in physical systems [19], [20].

Contribution. In this paper, we propose a new class of stable neural differential equations, called stable Hamiltonian neural dynamics (SHND).

- It has certified global Lyapunov stability and exponential stability by the parameterized Lyapunov function, with respect to an equilibrium point. The equilibrium point can either be imposed as a prior knowledge or

*This work was supported in part by the Australian Research Council, and the NSW Defence Innovation Network.

The authors are with the Australian Centre for Robotics (ACFR), and the School of Aerospace, Mechanical and Mechatronic Engineering, The University of Sydney, Sydney, NSW 2006, Australia ruigang.wang@sydney.edu.au

learned by the data.
- We further extend it to passive port-Hamiltonian system, which has potential applications in learning passivity-based controllers.
- Empirical results show that it outperforms existing methods in terms of fitting error, simulation error and robustness.

Paper organization. In section II we give a problem definition and review some preliminaries results. We present our main result in section III. Section IV illustrates our approach on a damped double pendulum, followed by conclusions in Section V.

Notation. Given a $x \in \mathbb{R}^n$, we denote a ball around x with radium ϵ by $\mathcal{B}_\epsilon(x) := \{y \in \mathbb{R}^n \mid |y-x| \leq \epsilon\}$ where $|\cdot|$ is the Euclidean norm. We denote $A \succeq B$ if and only if the matrix $(A - B)$ is positive semi-definite. We define the gradient transpose operator as $\nabla_x := (\partial/\partial x)^\top$. The subscript will be omitted when it is clear from the context.

II. Problem Formulation and Preliminaries

A. Problem setup

Given a dataset $\mathcal{D} = \{(x_i, v_i) \mid x_i, v_i \in \mathbb{R}^n\}$, we consider the problem of learning a neural vector field $v = f(x)$ from $\mathcal{D}$ such that the following differential equation

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

is *guaranteed* to be stable in the following senses.

Definition 1. Let $x^\star$ be an equilibrium of (1), i.e., $f(x^\star) = 0$.

1) System (1) is said to be *stable* at $x^\star$ if, for each $\epsilon > 0$, there is $\delta = \delta(\epsilon) > 0$ such that

$$x(0) \in \mathcal{B}_\delta(x^\star) \Rightarrow x(t) \in \mathcal{B}_\epsilon(x^\star), \quad \forall t \geq 0. \tag{2}$$

2) It is said to be *exponentially stable* if there exist $\delta, \lambda, \kappa > 0$ such that

$$x(0) \in \mathcal{B}_\delta(x^\star) \Rightarrow |x(t) - x^\star| \leq \kappa e^{-\lambda t}|x(0) - x^\star| \tag{3}$$

for all $t \geq 0$. It is said to be *globally* exponentially stable if (3) holds for all $\delta > 0$.

We also consider learning $v = f(x, u)$ and $y = h(x)$ from a dataset $\mathcal{D} = \{(x_i, v_i, u_i, y_i) \mid x_i, v_i \in \mathbb{R}^n, u_i, y_i \in \mathbb{R}^m\}$ such that the following controlled system in the input-state-output form

$$\dot{x} = f(x, u), \quad y = h(x) \tag{4}$$

is internally stable and passive.

Definition 2 ([21]). System (4) is said to be *passive* if there exists a continuously differentiable positive semidefinite function $S(x)$ (storage function) such that

$$\frac{\partial S}{\partial x} f(x, u) \leq u^\top y, \quad \forall (x, u) \in \mathbb{R}^n \times \mathbb{R}^m. \tag{5}$$

B. Preliminaries

We review two types of neural networks which will be applied in our stable model construction.

Definition 3. A neural network $g : \mathbb{R}^n \to \mathbb{R}^n$ is said to be *bi-Lipschitz* if there exist $\nu \geq \mu > 0$ such that

$$\mu|x - x'| \leq |g(x) - g(x')| \leq \nu|x - x'|, \quad \forall x, x' \in \mathbb{R}^n. \tag{6}$$

We call such g a (μ, ν)-Lipschitz network. It is worth to point out that g is also invertible and g^{-1} is a bi-Lipschitz network with bound of $(1/\nu, 1/\mu)$. In literature, there are some existing approaches [22], [23], [24], [25] to learn bi-Lipschitz neural networks with certified bounds.

The following definition from [18] was motivated by Polyak-Łojasiewicz condition [26], [27]

Definition 4. A scalar-output neural network $H : \mathbb{R}^n \to \mathbb{R}$ is called a *Polyak-Łojasiewicz network* (PLNet) if there exists a $m > 0$ such that

$$|\nabla_x H|^2 \geq 2m\big(H(x) - H^\star\big), \quad \forall x \in \mathbb{R}^n, \tag{7}$$

where $H^\star := \min_{y \in \mathbb{R}^n} H(y)$.

One main result of [18] is that given any (μ, ν)-Lipschitz network g, we can obtain a PLNet as follows:

$$H(x) = 0.5|g(x)|^2 + c, \quad c \in \mathbb{R}, \tag{8}$$

i.e., (7) holds with $m = \mu^2$. Since our model construction will only use the gradient ∇H, we assume $c = 0$ throughout this paper. Some nice properties of PLNet include:

1) H could be non-convex but always has one minimum $x^\star = g^{-1}(0)$, i.e.,

$$\nabla H(x^\star) = 0, \quad \nabla H(x) \neq 0, \ \forall x \in \mathbb{R}^n \backslash \{x^\star\}. \tag{9}$$

2) H is upper- and lower-bounded by quadratic functions:

$$0.5\mu^2|x - x^\star|^2 \leq H(x) \leq 0.5\nu^2|x - x^\star|^2. \tag{10}$$

3) If g is a smooth mapping, then the level set $\mathbb{L}_\alpha = \{x \in \mathbb{R}^n \mid H(x) \leq \alpha\}$ is homeomorphic to a unit ball in $\mathbb{R}^n$, for any $\alpha > 0$.

These features make the PLNet a promising candidate for learning Lyapunov functions, e.g. using the methods of [28], [17]. In this paper we focus on learning stable dynamics by parameterising flows that are descent directions for such a Lyapunov function, and consider further extensions to passive dynamics.

III. Stable Hamiltonian Neural Dynamics

In this section, we first present a class of stable neural dynamics in the form of Hamiltonian system, and then extend it to passive dynamical models.

A. Stable Hamiltonian models

Our proposed model has the form of general Hamiltonian dynamics as follows:

$$\dot{x} = [J(x) - R(x)]\nabla H(x) \tag{11}$$

where $x(t) \in \mathbb{R}^n$ is the state, $H(x) \in \mathbb{R}$ is the Hamiltonian, and $J(x) = -J^\top(x)$, $R(x) = R^\top(x)$ are the interconnection and damping matrices, respectively, which will be parameterized by neural networks. We now state our main result.

Theorem 1. *Suppose that H is a PLNet* (8) *with g as a (μ, ν)-Lipschitz network. Let $x^\star$ be the global minimum of H. We have the following results:*

a) *If $R(x) \succeq 0$, $\forall x \in \mathbb{R}^n$, then* (11) *is stable at $x^\star$, i.e.,* (2) *holds with $\delta(\epsilon) = \epsilon\mu/\nu$.*

b) *If there exist $\epsilon > 0$ such that $R(x) \succeq \epsilon I$, $\forall x \in \mathbb{R}^n$, then* (11) *is globally exponentially stable at $x^\star$, i.e.,* (3) *holds with $\kappa = \nu/\mu$ and $\lambda = \epsilon$.*

Proof. a) As shown in Section II-B, $H(x)$ is a positive semidefinite function and its derivative satisfies

$$\begin{aligned}\dot{H} &= \nabla^\top H(x)[J(x) - R(x)]\nabla H(x) \\ &= -\nabla^\top H(x) R(x) \nabla H(x) \le 0, \quad \forall x \in \mathbb{R}^n\end{aligned} \tag{12}$$

implying $H(x(t)) \le H(x(0))$. From (10), we have

$$|x(t) - x^\star| \le \frac{\sqrt{2H(x(t))}}{\mu} \le \frac{\sqrt{2H(x(0))}}{\mu} \le \frac{\nu}{\mu}|x(0) - x^\star|,$$

implying $|x(0) - x^\star| \le \mu\epsilon/\nu \Rightarrow |x(t) - x^\star| \le \epsilon$ for all $t \ge 0$.

b) Since $R(x) \succeq \epsilon I$, then (12) implies

$$\dot{H} \le -\epsilon|\nabla H|^2 \le -2\epsilon\mu^2(H(x) - H(x^\star)) = -2\epsilon\mu^2 H(x), \tag{13}$$

Furthermore, we have

$$\begin{aligned}|x(t) - x^\star| &\le \frac{\sqrt{2H(x(t))}}{\nu} \le \frac{\sqrt{2e^{-2\lambda t}H(x(0))}}{\nu} \\ &\le \frac{\nu}{\mu} e^{-\lambda t}|x(0) - x^\star|,\end{aligned} \tag{14}$$

where $\lambda = \epsilon\mu^2$, i.e., (11) is globally exponentially stable with rate $\epsilon\mu^2$ and overshoot ν/μ. □

Remark 1. The hyper-parameters μ, ν and ϵ offer us the flexibility to control model's minimum convergence rate and overshoot.

Remark 2. The Hamiltonian $H(x)$ is a Lyapunov function $V(x)$ of system (11).

If the equilibrium $x^\star$ is unknown, we simply take the PLNet (8) as the learnable Hamiltonian. Otherwise, we can impose the prior knowledge of $x^\star$ into our model by choosing the Hamiltonian $H(x) = 0.5|g(x) - g(x^\star)|^2$, which also is a PLNet as $\tilde{g}(x) := g(x) - g(x^\star)$ is a bi-Lipschitz model with the same bound as g. In this work we will use the bi-Lipschitz network recently proposed in [18]. Specifically, $y = g_\theta(x)$ has the form of

$$\begin{aligned}z_k &= \sigma(W_k z_{k-1} + U_k x + b_k),\ z_0 = 0 \\ y &= \mu x + \sum_{k=1}^{L} Y_k z_k + b_y\end{aligned} \tag{15}$$

where z_k is the kth layer hidden unit, and x, y are the input and output of this network. The learnable parameters of g are $\theta = \{U_k, Y_k, W_k, b_k, b_y\}$. One main result of [18] is a model parameterization $\mathcal{M} : \phi \to \theta$ with $\phi \in \mathbb{R}^N$ as a free variable such that for any $\phi \in \mathbb{R}^N$, the model $g_{\mathcal{M}(\phi)}$ is bi-Lipschitz for some prescribed bound of (μ, ν). Then, the off-shelf optimization algorithm (e.g., stochastic gradient decent) can be directly applied to train bi-Lipschitz models.

We now parameterize the matrix functions $J(x)$ and $R(x)$. Let $S(x)$ and $L(x)$ be two learnable multi-layer perceptions (MLPs), which are mappings from $\mathbb{R}^n$ to $\mathbb{R}^{n\times n}$. We then construct

$$J(x) = S(x) - S^\top(x), \quad R(x) = L^\top(x)L(x) + \epsilon I \tag{16}$$

where $\epsilon \ge 0$ is a user-specified constant.

B. Passive port-Hamiltonian models

We further extend (11) to port-Hamiltonian system of the form

$$\begin{aligned}\dot{x} &= (J(x) - R(x))\nabla H(x) + B(x)u \\ y &= B(x)^\top \nabla H(x)\end{aligned} \tag{17}$$

where $u(t), y(t) \in \mathbb{R}^m$ are the input-output variables, respectively, and $B(x) \in \mathbb{R}^{n\times m}$ is a learnable matrix function. The following theorem shows that (17) is passive with respect to the storage function $H(x)$.

Theorem 2. *Suppose that H is a PLNet and the matrix function R satisfies $R(x) \succeq 0$ for all x. Then,* (17) *is passive.*

Proof. We take $H(x)$ as the storage function and then check the dissipation inequality as follows:

$$\begin{aligned}\dot{H} - u^\top y &= -\nabla^\top H R \nabla H + \nabla^\top H B u - u^\top B^\top \nabla H \\ &= -\nabla^\top H R \nabla H \le 0.\end{aligned} \tag{18}$$

□

Remark 3. An potential application of the above model class is learning stabilizing controllers for unknown but passive systems [29].

IV. Experiments

We illustrate the proposed approach on learning the dynamics of a damped double pendulum system. The pendulum state is $x = [\,\theta_1\ \theta_2\ \dot{\theta}_1\ \dot{\theta}_2\,]^\top$ where θ_1, θ_2 are the pendulum joint angles and $\dot{\theta}_1, \dot{\theta}_2$ are the angular velocities. The dynamics of the damped pendulum is $\dot{x} = f_p(x)$ where the vector filed $f_p : \mathbb{R}^4 \to \mathbb{R}^4$ can be found in [15]. Due to damping on each joint, the above system has a stable equilibrium $x^\star = 0$, i.e., $f_p(0) = 0$. Code is available at `https://github.com/ruigangwang7/StableNODE`.

The main objective is to learn a vector field $f : \mathbb{R}^4 \to \mathbb{R}^4$ based on a set of samples (x_i, v_i) where $v_i = f_p(x_i)$. To be specific, we randomly generate 2000 samples as the train data set $\mathcal{D}_{\text{train}} := \{(x_i, v_i)\}_{1\le i\le 2000}$, where each x_i is uniformly drawn from the region where $|\theta_i| \le \pi$ rad and $|\dot{\theta}_i| \le \pi$ rad/s with $i = 1, 2$. The test data set $\mathcal{D}_{\text{test}}$ consists of 500 samples

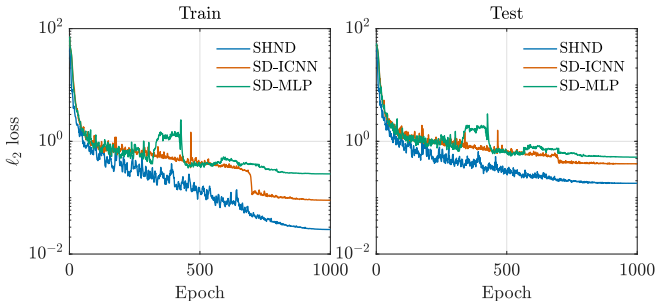

Fig. 1. Training and test loss (19) versus epochs. The proposed SHND model achieves better performance than SD-ICNN and SD-MLP.

with the same distribution. Given a learnable model f, the train/test loss are defined as the mean square ℓ_2 loss:

$$\ell(f) = \frac{1}{n_{\mathcal{D}}} \sum_{i=1}^{n_{\mathcal{D}}} \left| v_i - f(x_i) \right|^2. \tag{19}$$

with $n_{\mathcal{D}}$ as the number of elements in $\mathcal{D}$, where $\mathcal{D}$ is either $\mathcal{D}_{\text{train}}$ or $\mathcal{D}_{\text{test}}$.

Beside loss minimization, we also want to learn a model f such that the associated dynamics is globally exponentially stable at $x^\star$. The main purpose of such requirement is that when simulating the model f, a small fitting loss (19) can lead to a small error between the trajectories generated by $\dot{x} = f_p(x)$ and $\dot{x} = f(x)$.

A. Model architecture and training details

We will compare our model with the Lyapunov projection based stable models from [15], which learn a pair of vector field $\hat{f} : \mathbb{R}^n \to \mathbb{R}^n$ and an Lyapunov function $V : \mathbb{R}^n \to \mathbb{R}$. The dynamical model is defined by

$$\dot{x} = f(x) = \hat{f}(x) - \nabla V(x) \frac{\texttt{ReLU}(\nabla V(x)^T \hat{f}(x) + \alpha V(x))}{|\nabla V(x)|^2} \tag{20}$$

where $\texttt{ReLU}(x) = \max(0, x)$. If the vector filed $\hat{f}$ satisfies the following Lyapunov inequality

$$\nabla V(x)^\top \hat{f}(x) \leq -\alpha V(x), \tag{21}$$

we have $f(x) = \hat{f}(x)$. For the region where the above Lyapunov inequality does not hold, the vector field f then takes the descent direction of V, i.e.,

$$f(x) = -\alpha \frac{V(x)}{|\nabla V(x)|^2} \nabla V(x). \tag{22}$$

For the vector field $\hat{f}$, we choose a 4-100-100-4 multi-layer perception (MLP), i.e., a network with input and output dimension of 4 and two hidden layer with each has 100 neurons. For the Lyapunov function V, we consider two the following two cases extracted from [15]:

- **SD-MLP**: We take a 4-64-64 MLP g and then construct $V(x) = |g(x)|^2$. And the resulting model in (20) is referred as SD-MLP. Note that such choice of Lyapunov function generally does not satisfy (9) and (10). Thus, (20) only ensures that $V(x(t))$ is bounded. The simulation trajectory may contain oscillations.
- **SD-ICNN**: We take a 4-64-64-1 ICNN [16] as the Lyapunov function $V(x)$. Although $V(x)$ is convex, it may have multiple global minimum. By adding a quadratic term $\epsilon|x|^2$ to V, one can ensure that the resulting Lyapunov function has a quadratic lower bound. But it is still not clear how to enforce the quadratic upper bound. Moreover, when the prior knowledge about the equilibrium $x^\star$ is available, it is unclear how to enforce the constraint $V(x^\star) = 0$ to the ICNN. Therefore, the resulting dynamics (20) may converge to an undesired equilibrium.

In summary, since the existing approaches for constructing neural Lyapunov function do not guarantee

$$c_1|x|^2 \leq V(x) \leq c_2|x|^2$$

with $0 < c_1 \leq c_2$, then the dynamics (20) does not provide exponentially guarantees for the equilibrium $x^\star$.

For the proposed SHND model, we use the bi-Lipschitz network (15) from [18] with architecture 4-32-32-4 and Lipschitz bound $(\mu, \nu) = (0.1, 2)$ to construct the Hamiltonian $H(x)$. For the matrix functions $J(x)$ and $R(x)$, we first take a 4-90-90-32 MLP, then partition the output into two 4×4 matrices L and S, and finally compute J, R via (16) with $\epsilon = 0.01$. As shown in Thm 1, the SHND model can provide global exponential stability guarantee for the equilibrium $x^\star$. We will illustrate this point in the next section. For a fair comparison, our SHND model has a similar amount of learnable model parameters (around 15K) as the SD-MLP and SD-ICNN.

We use Adam [30] to train each model with mean square loss for 1000 epochs. The batch size is 200 and the learning rate starts from 0.01 and decays to 0 according to the `cosine` rate scheduler.

B. Results and discussions

We first show the training and test loss over the epochs in Fig. 1. Our proposed SHND model outperforms SD-MLP and SD-ICNN. To test the model stability, we simulate the learned dynamics $\dot{x} = f(x)$ as well as the pendulum dynamics $\dot{x} = f_p(x)$ and then compare the error between those trajectories. To be specific, we take a batch of 100 samples $|\theta_i| \leq \pi/2$ and $\dot{\theta}_i = 0$ as initial conditions, then simulate the dynamics using Runge-Kutta method with step size of 0.01s. We plot the batch averaged and maximum simulation errors in Fig. 2, as well as the trajectories w.r.t. two initial conditions in Fig. 5.

We can observe that our model has much smaller simulation errors compared to SD-ICNN and SD-MLP. In particular, the simulation error of SHND converges to 0 as the simulation time increases, demonstrating that our model can provide stability guarantee for the known equilibrium $x^\star$. For SD-ICNN model, as shown in Fig. 5, with slight changes in the initial condition, it converges to a different equilibrium. The SD-MLP produces oscillating behaviors as its Lyapunov function is not positive definite. Also, its simulation error is much larger than the other two approaches.

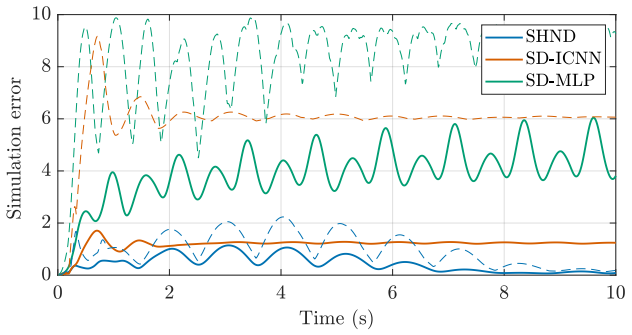

Fig. 2. Batch averaged (solid) and maximum (dashed) simulation error between the trajectories generated by the learned model and the original pendulum dynamics w.r.t. a batch of initial conditions. Our model has much smaller simulation error and converges to 0 as time increases. This is mainly due to the stability guarantee w.r.t. the known equilibrium $x^\star$. The SD-ICCN has a steady error due to the fact that it converges to another equilibrium for some initial conditions. The SD-MLP produces oscillating trajectories.

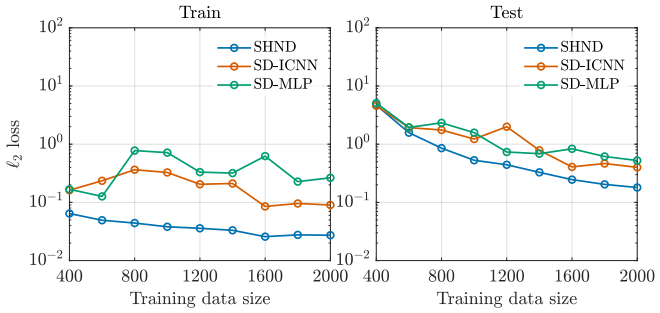

Fig. 3. Training and test loss (19) versus training data size. We use the same test data set with 500 samples. For our model, the performance improves as the size increases while the other two have some variations.

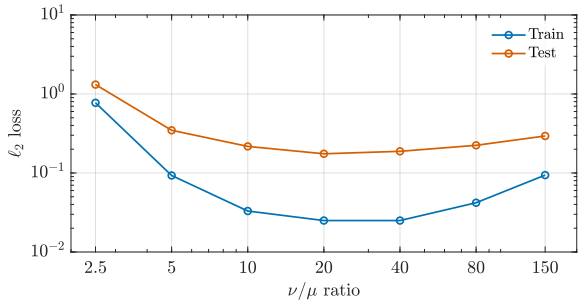

Fig. 4. Effect of the ratio ν/μ on the train/test loss (19). We set $\mu = 0.1$ and trained the SHND models with different ν. The ratio ν/μ can be serve as a regularizer for the proposed SHND model.

We also perform additional model training with respect to different data sizes. From Fig. 3 we can see that our model has a constant performance improvement as the data size increases while the other two approaches have some variation.

Different hyper-parameter ν/μ in our model is also tested. The result in Fig. 4 shows that ν/μ has a regularization effect on our model.

V. Conclusion

In this paper, we have proposed a new class of neural differential equations called stable Hamiltonian neural dynamics (SHND), which has built-in stability with respect to an equilibrium. Empirical results on a vector field fitting problem demonstrate its effectiveness. For future directions, we will consider to learn stable models from time-series data and also explore applications in learning passivity-based controllers.

References

[1] K. Hornik, M. Stinchcombe, and H. White, "Multilayer feedforward networks are universal approximators," *Neural networks*, vol. 2, no. 5, pp. 359–366, 1989.

[2] R. T. Chen, Y. Rubanova, J. Bettencourt, and D. K. Duvenaud, "Neural ordinary differential equations," *Advances in neural information processing systems*, vol. 31, 2018.

[3] M. M. Tobenkin, I. R. Manchester, J. Wang, A. Megretski, and R. Tedrake, "Convex optimization in identification of stable non-linear state space models," in *49th IEEE Conference on Decision and Control (CDC)*, pp. 7232–7237, IEEE, 2010.

[4] S. M. Khansari-Zadeh and A. Billard, "Learning stable nonlinear dynamical systems with gaussian mixture models," *IEEE Transactions on Robotics*, vol. 27, no. 5, pp. 943–957, 2011.

[5] M. M. Tobenkin, I. R. Manchester, and A. Megretski, "Convex parameterizations and fidelity bounds for nonlinear identification and reduced-order modelling," *IEEE Transactions on Automatic Control*, vol. 62, no. 7, pp. 3679–3686, 2017.

[6] J. Umenberger, J. Wågberg, I. R. Manchester, and T. B. Schön, "Maximum likelihood identification of stable linear dynamical systems," *Automatica*, vol. 96, pp. 280–292, 2018.

[7] W. Lohmiller and J.-J. E. Slotine, "On contraction analysis for nonlinear systems," *Automatica*, vol. 34, no. 6, pp. 683–696, 1998.

[8] M. Revay and I. Manchester, "Contracting implicit recurrent neural networks: Stable models with improved trainability," in *Learning for Dynamics and Control*, pp. 393–403, PMLR, 2020.

[9] M. Revay, R. Wang, and I. R. Manchester, "Recurrent equilibrium networks: Flexible dynamic models with guaranteed stability and robustness," *IEEE Transactions on Automatic Control*, 2023.

[10] F. Fan, B. Yi, D. Rye, G. Shi, and I. R. Manchester, "Learning stable koopman embeddings for identification and control," *arXiv preprint arXiv:2401.08153*, 2024.

[11] D. Martinelli, C. L. Galimberti, I. R. Manchester, L. Furieri, and G. Ferrari-Trecate, "Unconstrained parametrization of dissipative and contracting neural ordinary differential equations," in *2023 62nd IEEE Conference on Decision and Control (CDC)*, pp. 3043–3048, IEEE, 2023.

[12] F. Wilson Jr, "The structure of the level surfaces of a lyapunov function," 1967.

[13] M. T. Anderson, "Geometrization of 3-manifolds via the ricci flow," *Notices AMS*, vol. 51, pp. 184–193, 2004.

[14] S. M. Richards, F. Berkenkamp, and A. Krause, "The lyapunov neural network: Adaptive stability certification for safe learning of dynamical systems," in *Conference on Robot Learning*, pp. 466–476, PMLR, 2018.

[15] J. Z. Kolter and G. Manek, "Learning stable deep dynamics models," *Advances in neural information processing systems*, vol. 32, 2019.

[16] B. Amos, L. Xu, and J. Z. Kolter, "Input convex neural networks," in *International Conference on Machine Learning (ICML)*, pp. 146–155, PMLR, 2017.

[17] C. Dawson, S. Gao, and C. Fan, "Safe control with learned certificates: A survey of neural lyapunov, barrier, and contraction methods for robotics and control," *IEEE Transactions on Robotics*, 2023.

[18] R. Wang, K. Dvijotham, and I. R. Manchester, "Monotone, bi-Lipschitz, and Polyak-Lojasiewicz networks," *arXiv preprint arXiv:2402.01344*, 2024.

[19] A. Van Der Schaft, D. Jeltsema, *et al.*, "Port-hamiltonian systems theory: An introductory overview," *Foundations and Trends® in Systems and Control*, vol. 1, no. 2-3, pp. 173–378, 2014.

[20] V. Duindam, A. Macchelli, S. Stramigioli, and H. Bruyninckx, *Modeling and control of complex physical systems: the port-Hamiltonian approach*. Springer Science & Business Media, 2009.

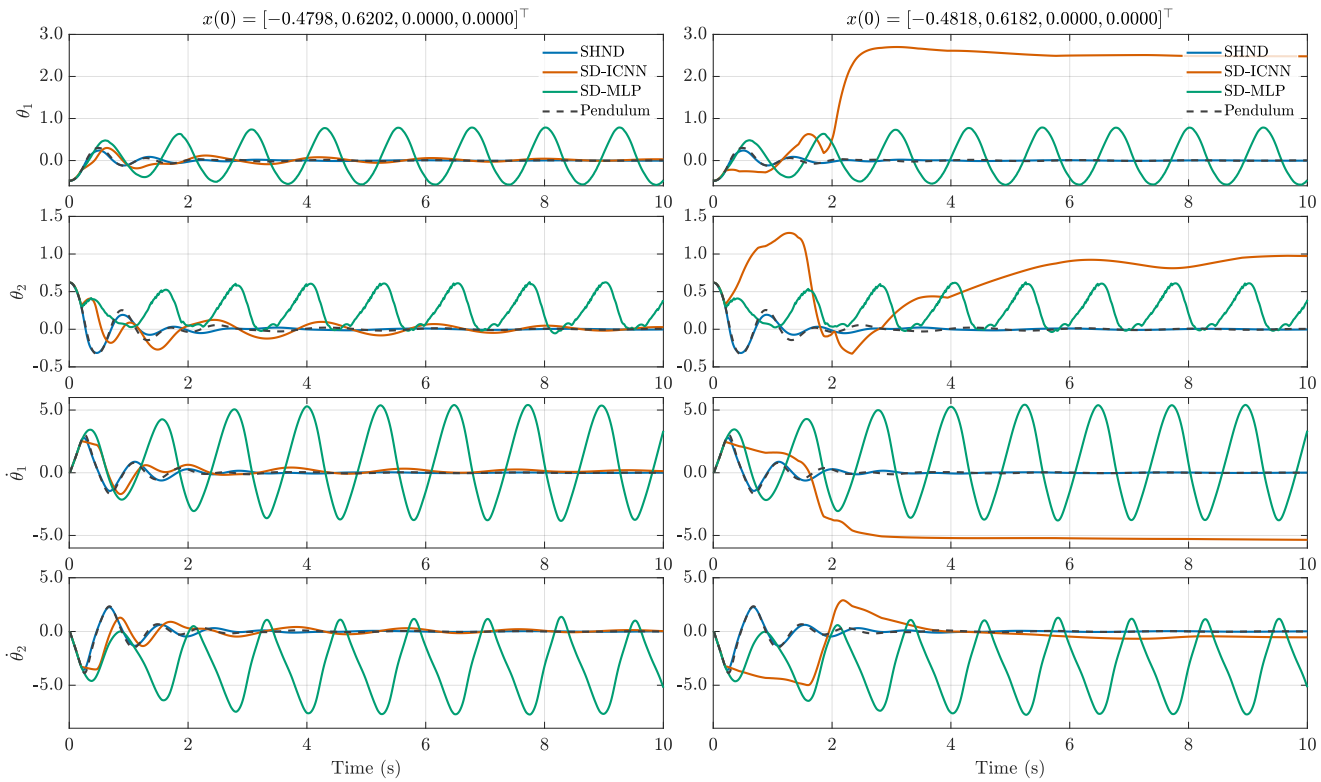

Fig. 5. Simulation trajectories of different models with two initial states. The SD-MLP model generates oscialtting behaviors. Both SD-ICNN and the proposed SHND converge. But the SD-ICNN converges to a non-zero equilibrium point despite a small change in the initial condition.

[21] H. K. Khalil, *Nonlinear systems*. Prentice Hall, New York, NY, 2002.

[22] R. T. Chen, J. Behrmann, D. K. Duvenaud, and J.-H. Jacobsen, "Residual flows for invertible generative modeling," *Advances in Neural Information Processing Systems*, vol. 32, 2019.

[23] J. Behrmann, W. Grathwohl, R. T. Chen, D. Duvenaud, and J.-H. Jacobsen, "Invertible residual networks," in *International conference on machine learning (ICML)*, pp. 573–582, PMLR, 2019.

[24] C. Lu, J. Chen, C. Li, Q. Wang, and J. Zhu, "Implicit normalizing flows," in *International Conference on Learning Representations (ICLR)*, 2021.

[25] B. Ahn, C. Kim, Y. Hong, and H. J. Kim, "Invertible monotone operators for normalizing flows," *Advances in Neural Information Processing Systems*, vol. 35, pp. 16836–16848, 2022.

[26] B. Polyak, "Gradient methods for minimizing functionals (in russian)," *USSR Computational Mathematics and Mathematical Physics*, vol. 3, no. 4, pp. 643–653, 1963.

[27] S. Lojasiewicz, "A topological property of real analytic subsets," *Coll. du CNRS, Les équations aux dérivées partielles*, vol. 117, no. 87-89, p. 2, 1963.

[28] N. Boffi, S. Tu, N. Matni, J.-J. Slotine, and V. Sindhwani, "Learning stability certificates from data," in *Proceedings of the 2020 Conference on Robot Learning* (J. Kober, F. Ramos, and C. Tomlin, eds.), vol. 155 of *Proceedings of Machine Learning Research*, pp. 1341–1350, PMLR, 16–18 Nov 2021.

[29] C. Secchi, S. Stramigioli, and C. Fantuzzi, *Control of interactive robotic interfaces: A port-Hamiltonian approach*, vol. 29. Springer Science & Business Media, 2007.

[30] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *International Conference on Learning Representations (ICLR)*, 2015.