Learning Robust State Observers using Neural ODEs

Keyan Miao *   KEYAN.MIAO@ENG.OX.AC.UK
Konstantinos Gatsis   KONSTANTINOS.GATSIS@ENG.OX.AC.UK
Department of Engineering Science, University of Oxford

Editors: N. Matni, M. Morari, G. J. Pappas

Abstract

Relying on recent research results on Neural ODEs, this paper presents a methodology for the design of state observers for nonlinear systems based on Neural ODEs, learning Luenberger-like observers and their nonlinear extension (Kazantzis-Kravaris-Luenberger (KKL) observers) for systems with partially-known nonlinear dynamics and fully unknown nonlinear dynamics, respectively. In particular, for tuneable KKL observers, the relationship between the design of the observer and its trade-off between convergence speed and robustness is analysed and used as a basis for improving the robustness of the learning-based observer in training. We illustrate the advantages of this approach in numerical simulations.

1. Introduction

The emergence of deep artificial neural network architectures (DNNs) as powerful function approximations in machine learning and computer vision tasks has generated an interest in using DNNs, and data-driven approaches more generally, in complex problems that involve systems with dynamics, such as for the purpose of control (Lillicrap et al., 2015). Finding suitable methodologies for incorporating and training DNNs for systems with dynamics is an important challenge. In this paper, we explore the use of the recent tool of Neural Ordinary Differential Equations (NODEs) for the design of NNs for dynamical systems. Specifically, we consider the task of designing state estimators/observers for systems with nonlinear dynamics.

Observers are used to estimate the unmeasured state of a dynamical system based on its output, and they are a key component of closed loop control systems. For linear systems, a general approach is based on the Luenberger observer design originally presented by Luenberger (1964). In contrast, there are not many general approaches to observer design for nonlinear systems, a review of which is given in Bernard (2019). High-gain observers (HGO) (Bornard and Hammouri, 1991) and Extended Kalman Filter (EKF) (Reif and Unbehauen, 1999) are most commonly used. However, HGO show poor transient performance and a high sensitivity to noise, while EKF only guarantee local convergence. In addition, Luenberger method has also been extended to nonliner systems, leading to the so-called Kazantzis-Kravaris-Luenberger (KKL) observer (Kazantzis and Kravaris, 1998). The idea of this approach is first immersing the nonlinear system into a latent linear system of higher dimension with an output injection and then mapping this linear system to a state estimate. The existence and injectivity of this mapping is guaranteed by mild observability conditions, which makes this design relatively general. Exponential convergence of KKL observer can be guaranteed by appropriately designing the observer dynamics to have a contraction property and is tuneable

---

under additional observability conditions (Andrieu, 2014). However, the main challenge lies in computing the latent linear system and the mapping between it and the nonlinear system.

To overcome this challenge, an initial attempt at neural network-based KKL observer design is reported by Ramos et al. (2020), and follow-up research were proposed within the last year (Peralez and Nadri (2021), Buisson-Fenet et al. (2022a), Niazi et al. (2022)). In general this approach trains a neural network to approximate the mapping and its left inverse, using supervised learning methods by fixing the linear dynamics of the KKL observer or using auto-encoders.

In this paper, to design nonlinear systems observers, we propose the tool of Neural ODEs. This tool was popularized by Chen et al. (2018), which used it to abstract the hidden layers of DNNs as dynamics of a continuous-time system described by an ODE. This approach sparked significant interest in the machine learning community recently (e.g. Rubanova et al. (2019), Massaroli et al. (2020), Zakwan et al. (2022), Rodriguez et al. (2022), Djeumou et al. (2022b), Kidger (2022)). Similar ideas have been discussed by Ee (2017) and Li et al. (2022). But beyond static machine learning problems, it is natural to use Neural ODEs to solve problems in dynamical systems since dynamical systems are typically represented by ODEs, and initial studies using Neural ODEs for system identification were recently proposed (Djeumou et al. (2022a), Buisson-Fenet et al. (2022b)). The contributions of our paper are: a) we propose a complete approach to the design of state observers for nonlinear systems using Neural ODEs, b) we show the relationship between the parameters of tuneable KKL observer and its rate of convergence and robustness to model uncertainty and measurement noise, and c) we incorporate this relationship in the training process as a way to design robust observers and show numerically the advantages of this approach compared to the literature. With respect to the literature on learning nonlinear observers, our approach based on Neural ODEs allows us to tune both the latent observer dynamics and its mapping to enhance robustness. With respect to the Neural ODEs literature mentioned above, our paper expands the use of this tool from static machine learning problems to problems when DNNs interact in a closed loop with physical system dynamics. After providing preliminaries on state observers for nonlinear systems and Neural ODEs in Section 2 and 3, we state the problems addressed in this paper and the methodology in Section 4. Numerical results are presented in Section 5, showing the benefits of our approach and the limitations particularly in terms of generalization, and Section 6 concludes with conclusions and future perspectives.

Notation: Throughout this paper, for a vector or a matrix, $|\cdot|$ denotes (induced) norm 2; $\Re$ denotes the real part.

2. Problem formulation for Nonlinear State Observer

Consider an autonomous nonlinear system:

$$\dot{x}(t) = f(x(t)), \ y(t) = h(x(t)) \tag{1}$$

with state $x \in \mathbb{R}^{n_x}$, output $y \in \mathbb{R}^{n_y}$, and maps $f : \mathbb{R}^{n_x} \to \mathbb{R}^{n_x}$ and $h : \mathbb{R}^{n_x} \to \mathbb{R}^{n_y}$ sufficiently smooth. Our aim is to design a state observer that estimates the value of the state x from the output y, and makes the estimated value $\hat{x}$ converge to the real state value x. The observer of this system takes the general form:

$$\dot{z}(t) = \mathcal{F}(z(t), y(t)), \ \hat{x}(t) = \mathcal{G}(z(t)) \tag{2}$$

with state $z \in \mathbb{R}^{n_z}$ (the internal latent state of the observer), output $y \in \mathbb{R}^{n_y}$, and maps $\mathcal{F} : \mathbb{R}^{n_z} \times \mathbb{R}^{n_y} \to \mathbb{R}^{n_z}$ and $\mathcal{G} : \mathbb{R}^{n_z} \to \mathbb{R}^{n_x}$ which need to be designed. Unlike linear system where Luenberger observers

([Luenberger](), [1964]()) are wildly recognized, there are not many general approaches to observer design for nonlinear systems. In a seminal paper, [Kazantzis and Kravaris]() ([1998]()) have proposed to extend the Luenberger observer for linear systems to the nonlinear case – called the KKL observer. [Andrieu and Praly]() ([2006]()) proved the existence of such an observer under the following conditions.

Assumption 2.1 *There exists a compact set $\mathcal{X}$ such that for any solutions x to ([1]()) of interest, $x(t) \in \mathcal{X}$ for all $t \geq 0$.*

Assumption 2.2 *There exists an open bounded set $\mathcal{O}$ containing $\mathcal{X}$ such that ([1]()) is backward $\mathcal{O}$-distinguishable on $\mathcal{X}$, namely for any trajectories x_a and x_b of ([1]()) such that $(x_a(0), x_b(0)) \in \mathcal{X} \times \mathcal{X}$ and $x_a(0) \neq x_b(0)$, there exists $t \leq 0$ such that $h(x_a(t)) \neq h(x_b(t))$ and $(x_a(\tau), x_b(\tau)) \in \mathcal{O} \times \mathcal{O}$ for all $\tau \in [t, 0]$. In other words, their respective outputs become different in backward finite time before leaving $\mathcal{O}$.*

Theorem 1 ([Andrieu and Praly]() ([2006]())) *Suppose Assumptions [2.1]() and [2.2]() hold. Define $d_z = d_y(d_x + 1)$, then there exists $l > 0$ and a set $\mathcal{S}$ of zero measure in $\mathbb{C}^{d_z}$ such that for any matrix $D \in \mathbb{R}^{d_z \times d_z}$ with eigenvalues $(\lambda_1, ..., \lambda_{d_z})$ in $\mathbb{C}^{d_z} \setminus \mathcal{S}$ with $\mathfrak{R}\lambda_i < -l$, and any $F \in \mathbb{R}^{d_z \times d_x}$ such that (D, F) is controllable, there exists an injective mapping $\mathcal{T}: \mathbb{R}^{d_x} \to \mathbb{R}^{d_z}$ and a pseudo-inverse $\mathcal{T}^*: \mathbb{R}^{d_z} \to \mathbb{R}^{d_x}$ such that the trajectories of ([1]()) remaining in $\mathcal{X}$ and any trajectory of*

$$\dot{z}(t) = Dz(t) + Fy(t) \tag{3}$$

satisfy

$$|z(t) - \mathcal{T}(x(t))| \leq M|z(0) - \mathcal{T}(x(0))|e^{\lambda_{\max}t}, \ \lambda_{\max} = \max\{\mathfrak{R}\lambda_1, ..., \mathfrak{R}\lambda_{d_z}\} \tag{4}$$

for some $M > 0$, and $\mathcal{T}$ is the solution of the partial differential equation

$$\frac{\partial \mathcal{T}}{\partial x}(x)f(x) = D\mathcal{T}(x) + Fh(x) \tag{5}$$

and

$$\lim_{t \to +\infty} |x(t) - \mathcal{T}^*(z(t))| = 0 \tag{6}$$

The linear dynamics ([3]()) and the mapping $\hat{x} = \mathcal{T}^*(z)$ define the KKL observer, which is a special case of [2](). Although the above theorem guarantees the observer exists in a wide class of systems, it is still very difficult to find such $\mathcal{T}^*$. Additionally, there are many tuning choices that affect the performance of the observer, for example the matrix D affects convergence and sensitivity to measurement noise ([Buisson-Fenet et al.](), [2022a]()), although the relationship between performance and variables was not made explicit.

3. Preliminaries on Neural ODEs

It has been shown that DNNs can learn nonlinear mappings and can generalize to unseen data under certain conditions for a diverse range of problems. At one time, neural networks faced bottlenecks due to the limitations of the depth problem. The impetus for this breakthrough was the Residual Network (ResNet) proposed by [He et al.]() ([2016]()) which added skip connections to the network, resulting in a remarkable improvement in the performance of the network. Its universal approximation power has recently been explained from a non-linear control perspective ([Tabuada and Gharesifard](), [2020]()).

Meanwhile, the idea of treating the hidden layers of neural networks as states of a dynamical system became popular when ResNet was proposed. Chen et al. (2018) proposed an ODE specified by the neural network to parameterize the continuous dynamics:

Definition 2 (Neural ODEs) *With $h_x : \mathbb{R}^{n_x} \to \mathbb{R}^{n_z}$, $h_y : \mathbb{R}^{n_z} \to \mathbb{R}^{n_y}$ representing the input network and output network respectively, a Neural ODE is a system of the form*

$$\begin{cases} \dot{z}(t) = f(t, z(t), \theta) \\ z(t_0) = h_x(x) & t \in \mathcal{S} \\ \hat{y}(t) = h_y(z(t)) \end{cases} \tag{7}$$

where $\mathcal{S} := [t_0, t_f]$ $(t_0, t_f \in \mathbb{R}^+)$ is the depth domain and f is a neural network called ODENet which is chosen as a part of the machine learning model with parameter θ.

The solution of this ODE at some time t_f from an initial value $z(t_0)$ is $z(t_f)$, obtained with a differential equation solver by means of a specific solution scheme according to desired accuracy. The number of times the ODE solver evaluates the function in one forward pass can be interpreted as the number of hidden layers of the neural network, i.e., the depth. Now ResNet can be seen a special case of Neural ODEs using Euler discretization.

For common machine learning scenarios, such as image classification, an input data point x is mapped by Neural ODEs to $\hat{y}(t_f)$, e.g., a label. In other words, the inference of Neural ODEs is carried out by solving the initial value problem (IVP):

$$\hat{y}(t_f) = h_y \left(h_x(x) + \int_{t_0}^{t_f} f(t, z(t), \theta) \, dt \right) \tag{8}$$

The training of the Neural ODEs with parameters θ proposed in Chen et al. (2018) considers only a loss function that depends on the terminal state $\hat{y}(t_f)$, which is also the common scenario for supervised learning problems where the terminal state is compared to a true label. In this case, the training can be cast into a *Mayer* optimal control problem. However, for the purpose of our paper, it is valuable to introduce

$$\ell := \Phi(x(t_f)) + \int_{t_0}^{t_f} L(z(t), \theta, t) \, dt \tag{9}$$

since in the framework of Neural ODEs, the latent states evolve through layers, and then generate outputs. With such a loss function, the training can be cast into an optimization problem of the form

$$\begin{aligned} \min_{\theta \in U} \quad & \ell \\ s.t. \quad & \dot{z}(t) = f(t, z(t), \theta), t \in \mathcal{S} \\ & z(t_0) = h_x(x), \hat{y}(t) = h_y(z(t)) \end{aligned} \tag{10}$$

which is a *Bolza* optimal control problem (10) (Pontryagin, 1987), and the problem can be solved recursively by gradient descent (GD).

4. Learning Nonlinear State Observers using Neural ODEs

We now return to our main problem of learning state observers of the form (2) for systems described in (1). We propose the general framework of solving the state observation problem of nonlinear

systems based on Neural ODEs:

$$\dot{z}(t) = \mathcal{F}(z(t), y(t), \theta), \; \hat{x}(t) = \mathcal{G}(z(t), \eta)$$
$$\text{s.t.} \quad \dot{x}(t) = f(x(t)), \; y(t) = h(x(t)) \tag{11}$$

where $\hat{x}$ is the estimated state. The ODE that describes the dynamics of latent state $z(t)$ is represented by a neural network $\mathcal{F}$ whose parameters are θ, taking $z(t)$ and $y(t)$ as input. Transformation $\mathcal{G}$ parameterized by η can be seen as the output network stated in Neural ODEs structure (7). Furthermore, we propose the following loss function to be minimized for (11)

$$\ell := \int_{t_0}^{t_f} L dt = \int_{t_0}^{t_f} |x(t) - \hat{x}(t)|^2 dt \tag{12}$$

which is a *Lagrange* optimal control problem. Gradients of ℓ with respect to parameter θ can be computed via automatic differentiation or adjoint sensitivity analysis as follows which is with only $O(N_f)$ memory cost during training (Zhuang et al., 2020) where N_f denotes the number of layers of $\mathcal{F}$. $\mathcal{F}$ can incorporate a priori knowledge, such as physical information about the system, and the theory of nonlinear systems observer design, as described below.

Proposition 3 *Consider the problem* (11)-(12), *the gradient of loss ℓ with respect to parameter θ is*

$$\nabla_\theta \ell = \mu(t_0) \tag{13}$$

where $z(t)$, $p(t)$ and $\mu(t)$ satisfy the boundary value problem:

$$\dot{z}(t) = \mathcal{F}, \; z(t_0) = z_0$$
$$\dot{p}(t) = -p(t)\frac{\partial \mathcal{F}}{\partial z} - \frac{\partial L}{\partial z}, \; p(t_f) = \mathbb{0}_{n_z} \tag{14}$$
$$\dot{\mu}(t) = -p(t)\frac{\partial \mathcal{F}}{\partial \theta} - \frac{\partial L}{\partial \theta}, \; \mu(t_f) = \mathbb{0}_{n_\theta}$$

Detailed derivations are provided in Miao and Gatsis (2022). With the gradients with respect to the ODE parameters θ computed, the training process based on GD then can be carried out. The training data is generated by solving the dynamics with a numerical ODE solver over a finite time interval from many initial conditions, chosen randomly from a Gaussian distribution, and during the generation, output $y(t)$ is stored for the training as the input of ODENet. In our numerical experiments, it is possible to sample the data and train the model simultaneously for convenience, in which case the states of Neural ODEs can be extended as a stack of z with x, and the dynamics of x comes from the system rather than the network. This means that *our training does not require knowledge of the nonlinear system dynamics*, however it requires measurements of the true system states and outputs.
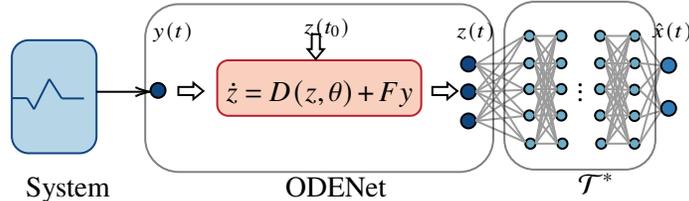

Figure 1: Schematics of the learned KKL Observer with Neural ODEs

4.1. Luenberger-like Observer for Systems with Partially-known Dynamics based on Neural ODEs

For systems whose linear output function C and the linear part A of dynamics are known, the Luenberger-like observer can be designed as (15) show. We incorporate the prior knowledge of system into Neural ODEs framework and take advantage of the strong ability of neural networks to represent nonlinear functions, here we use a three-layer network $\hat{g}$ parameterized by θ to learn the unknown nonlinear part (Abdollahi et al., 2006). G is the observer gain selected such that the matrix $A - GC$ is a Hurwitz matrix. The framework of Neural ODEs observer for this problem is proposed as:

$$\dot{\hat{x}}(t) = A\hat{x}(t) + \hat{g}(\hat{x}(t),\theta) + G(y(t) - \hat{y}(t)), \ \hat{y}(t) = C\hat{x}(t)$$
$$\text{s.t.} \quad \dot{x}(t) = Ax(t) + g(x(t)), \quad y(t) = Cx(t) \tag{15}$$

Based on Neural ODEs stated in (7), in this framework, the input network h_x and output network h_y can both be interpreted as identity transformations.

4.2. KKL observer for Systems with Unknown Dynamics based on Neural ODEs

When both the nonlinear system dynamics and the output measurement mappings are completely unknown given by (1), we propose to use the Neural ODEs framework to learn KKL observers as defined in Section 2, learning the matrix D and the mapping $\mathcal{T}^*$ jointly. To ensure that the matrix D is Hurwitz, it is designed as a diagonal matrix and θ represents the values on the diagonal, i.e., the eigenvalues of D, restricted to be negative real numbers. F is designed as $F = \mathbf{1}_{d_z \times d_y}$ to guarantee the controllability of (D, F). Let $\hat{\mathcal{T}}^*$ represent the neural network parameterized by η that approximates $\mathcal{T}^*$. The problem is stated as (16) where $z(t)$ is the latent state whose dynamics is described by ODENet.

$$\dot{z}(t) = D(z(t),\theta) + Fy(t) = Diag\{\theta_1,...,\theta_{d_z}\}z(t) + Fy(t)$$
$$\hat{x}(t) = \hat{\mathcal{T}}^*(z(t),\eta) \tag{16}$$
$$\text{s.t.} \quad \dot{x}(t) = f(x(t)), \ y(t) = h(x(t))$$

The schematics of the Neural ODEs observer incorporating KKL is illustrated in Figure 1. Based on Neural ODEs stated in (7), in this framework, the input network h_x can be interpreted as an identity transformation and the output network is $\hat{\mathcal{T}}^*$. We propose the loss function defined as (12) which penalizes the error between estimated state and true state value. Finally, our approach can be extended to non-autonomous systems by applying a stationary transformation (Bernard and Andrieu, 2019) that does not require a specific, well-chosen excitation that Ramos et al. (2020) proposed.

4.2.1. Robustness of KKL observer

The aim of this subsection is to characterize robustness, i.e., what happens if we set up the design of the state observer as in section 2, but then when deploying the state observer onto the system, the latter is subject to model uncertainties and measurement moise that will impact the state observer. Consider the case where noise in the system dynamics and measurements is present, we model a nonlinear system

$$\dot{x}(t) = f(x) + w(t), y(t) = h(x(t)) + v(t) \tag{17}$$

where $w \in \mathbb{R}^{n_x}$ and $v \in \mathbb{R}^{n_y}$. They represent model uncertainties and measurement noise respectively which are unknown but bounded signals such that $\sup_{t \in \mathcal{S}}|w(t)| \leq \bar{w}$, $\sup_{t \in \mathcal{S}}|Fv(t)| \leq \bar{v}$. Here, we focus on how the choice of D will affect the robustness of the observer by scaling its eigenvalues.

Proposition 4 *Suppose Assumptions 2.1 and 2.2 hold and assume infinitesimal distinguishability property holds (Andrieu, 2014). Consider the mapping $\mathcal{T}$ obtained by (5) where matrix $D = Diag\{\lambda_1, ..., \lambda_{d_z}\}$. Assume there exist positive real numbers $L_\mathcal{T} > 0$ and $L_{\mathcal{T}_x} > 0$ such that for all $x \in \mathcal{X}$*

$$|x_1 - x_2| \le L_\mathcal{T}|\mathcal{T}(x_1) - \mathcal{T}(x_2)|, \qquad and \quad |\frac{\partial \mathcal{T}}{\partial x}(x)| \le L_{\mathcal{T}_x}. \tag{18}$$

For any Hurwitz matrix $D_k = kD$, which means that all the eigenvalues are multiplied by a positive real number $k \ge 1$, let the corresponding mappings be $\mathcal{T}_k$ and $\mathcal{T}_k^*$ obtained by solving $\frac{\partial \mathcal{T}_k}{\partial x}(x) f(x) = D_k \mathcal{T}_k(x) + Fh(x)$. Then the trajectory $\{x(t), \tilde{x}(t)\}$, where x represents the state of system (17) and $\tilde{x} = \mathcal{T}_k^*(z)$ where the dynamics of z is $\dot{z}(t) = D_k z(t) + Fy(t)$, satisfies

$$|x(t) - \tilde{x}(t)| \le k^{n_z}\sqrt{n}_x L_\mathcal{T} \exp(k\lambda_{max}t) |z(t_0) - \mathcal{T}_k(x(t_0))| + \frac{k^{n_z}\sqrt{n}_x L_\mathcal{T}}{|k\lambda_{max}|}\left(\frac{L_{\mathcal{T}_x}}{k}\bar{w} + \bar{v}\right) \tag{19}$$

This proposition demonstrates the trade-off between convergence speed and robustness to model uncertainties and measurement noise: when increasing the coefficient k, i.e., enlarging the eigenvalues, the observer's convergence speed increases, but its robustness to model uncertainties and measurement noise decreases. A detailed proof is provided in Miao and Gatsis (2022).

Remark 5 *Learning-based KKL Observer Robustness to Approximation Error: The mapping that is learned by our Neural ODE approach, denoted here as $\hat{\mathcal{T}}^*(z)$, is only an approximation of $\mathcal{T}^*$. Therefore, the performance of the observer is further affected by the approximation error. Given a Lipschitz continuous activation function such as ReLU, it can be derived that the error between estimated state $\hat{x} = \hat{\mathcal{T}}^*(z)$ and x is bounded. A detailed proof is provided in Miao and Gatsis (2022). Moreover, the upper bound can be reduced by improving the design and learning techniques of the neural network and by increasing the size of training dataset.*

4.2.2. Training method to improve robustness

According to Proposition 4, the choice of matrix D, or more precisely its eigenvalues, will have a significant impact on the speed of convergence and robustness of the observer, and if no adjustments are made to the training, the learned observer will be concerned only with the speed of convergence. Here, we propose two methods to improve its performance on noise rejection.

- Adding noise to the data during training

- Using a regularization technique for the parameters (eigenvalues) of D through a penalty added to loss function

$$\tilde{\ell} := \ell + \gamma\ell_{reg} = \ell + \gamma \int_{t_0}^{t_f} |\theta(t)|^2 dt \tag{20}$$

Regularization techniques are often used in machine learning to solve overfitting problems, and poor robustness often implies overfitting. However, it should be clarified that the use of regularization in general machine learning problems is based on the premise that the overfitting problem originates from an overly complex network model, whereas here we use it based on the aforementioned analysis of robustness, i.e., that fast poles can lead to poor robustness of the observer. The addition of noise to the training process is considered to play an equivalent role to regularization (Bishop, 1995).

Remark 6 (Comparison with other approaches) *First, in the approach proposed by Ramos et al. (2020), the matrix D needs to be pre-specified to generate the dataset $\{x(t_i), z(t_i)\}$. Hence, the opportunity to fully tune the observer is missed in prior work, and similarly in Niazi et al. (2022). In contrast, our Neural ODEs approach makes it easy to design D and we show numerically the advantages of this approach in the following section. Buisson-Fenet et al. (2022a) proposed a method to improve the choice of D inspired by $\mathcal{H}_\infty$ control design, but in practice their method has shortcomings. It is an a posteriori method that requires the mappings corresponding with different D in order to select a better observer whereas their auto-encoder method requires full knowledge of the nonlinear system dynamics, and it does not directly improve robustness. In contrast, our Neural ODE approach is flexible enough to overcome these shortcomings.*

5. Numerical Results

We illustrate and evaluate our approach through numerical simulations.

Example 1 Consider an autonomous system

$$\begin{cases} \dot{x}_1(t) = x_2(t) + \sin x_1(t) \\ \dot{x}_2(t) = -x_1(t) + \cos x_2(t) \end{cases} \qquad y(t) = x_1(t) \tag{21}$$

where output function is known. Applying the methodology presented in Section 4, the training data (initial state) is generated from a uniform random distribution $\mathcal{X} = [-5, 5] \times [-5, 5]$, and the dynamics are solved over the time interval $[0, t_f]$ with $t_f = 50s$. The resulting observer is shown in Figure 2 for an arbitrary given initial condition. It can be found that the observer exhibits good performance and the error stabilizes after about $5s$.
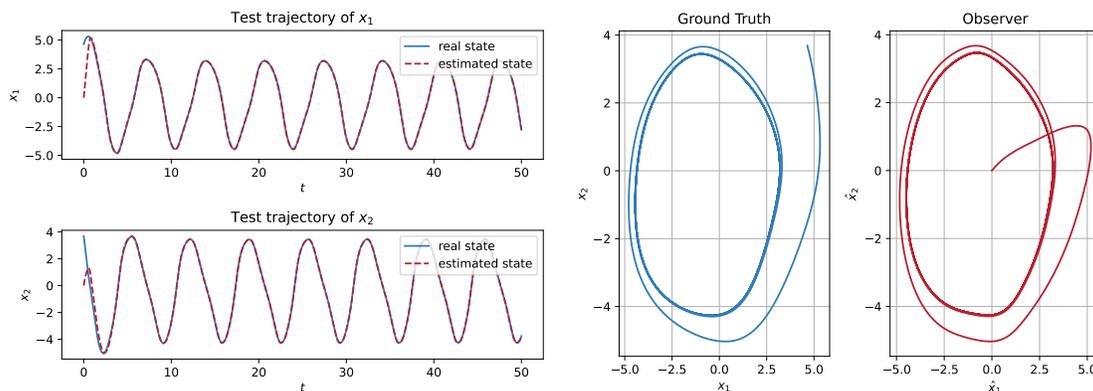

Figure 2: NODEs-based Luenberger-like observer result for a given initial condition

Example 2: Van der Pol Consider the autonomous Van der Pol oscillator

$$\begin{cases} \dot{x}_1(t) = x_2(t) \\ \dot{x}_2(t) = \left(1 - x_1^2(t)\right) x_2(t) - x_1(t) \end{cases} \qquad y(t) = x_1(t) \tag{22}$$

which admits a unique limit cycle. To demonstrate how the choice of D will affect the convergence speed and robustness of the KKL observer, we conducted the following experiments respectively: using the method in Ramos et al. (2020), specifying the eigenvalues of the diagonal matrix D as

Table 1: RMSE of learned observers for Van der Pol oscillator tested in different scenarios

Observer Test Scenario	Learned with fixed D (Ramos et al. (2020))			Learned via Neural ODEs (our method)	
	{-5, -6, -7}	{-0.1, -6, -7}	{-0.1, -0.2, -0.3}	Learned with Gaussian noise $\mathcal{N}(0, 0.5)$	Learned with Regularization
No Noise	**0.0548**	0.1786	0.2080	0.0603	0.0712
Gaussian Noise $\mathcal{N}(0, 0.5)$	0.1160	0.1903	0.2273	**0.0667**	0.0863
Uniform Noise $\mathcal{U}(-3, 3)$	0.3205	0.2586	0.2560	**0.1111**	0.1462
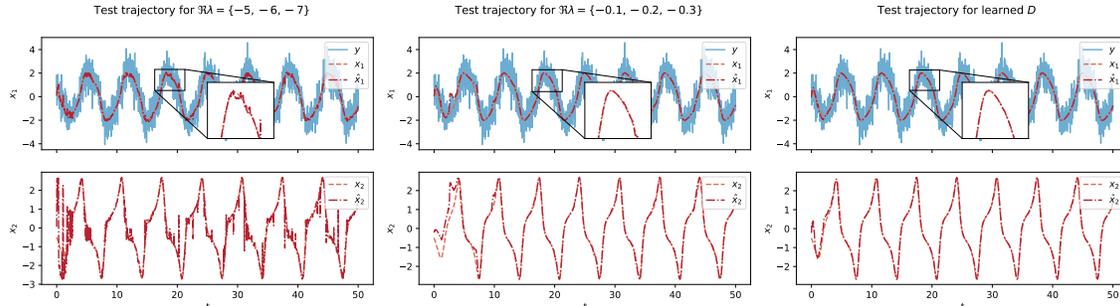
Figure 3: Test trajectories of Van der Pol for $x_0 = (-0.5, 0.5)$ with Gaussian noise $\mathcal{N}(0, 0.5)$

$\{-5, -6, -7\}$, $\{-0.1, -6, -7\}$ and $\{-0.1, -0.2, -0.3\}$ as baseline; using the aforementioned NODEs-based scheme, learning the KKL observer with noise added to the training data and using regularization technique respectively. The training data (initial state) is generated from a uniform random distribution $\mathcal{X} = [-1, 1] \times [-1, 1]$, and the dynamics are solved over the time interval $[0, t_f]$ with $t_f = 50s$. The learned observers based on our approach and baseline are then tested on arbitrary initial conditions and the root mean square error (RMSE) of different learned observers and their performance are illustrated in Table 1 and Figure 3. The third observer shown in Figure 3 is learned with training data added by Gaussian noise. It can be found that when D is specified with small absolute eigenvalues, the observer suffers little from noise but takes about 10 seconds to converge, whereas when a matrix D with larger absolute eigenvalues is chosen, it converges quickly but is severely affected by noise. In contrast, the observer based on Neural ODEs showed good performance in terms of both convergence speed and robustness. This demonstrates the advantages of our approach over previous approaches.

Example 3: Reverse Duffing Oscillator   Consider the reverse Duffing oscillator

$$\begin{cases} \dot{x}_1(t) = x_2^3(t) \\ \dot{x}_2(t) = -x_1(t) \end{cases} \quad y(t) = x_1(t) \tag{23}$$

which admits bounded trajectories where $x_1^2 + x_2^4$ is constant. The size of the training state space subset is critical, and Figure 4 shows RMSE of results for observers trained under different subsets tested over the same space for a trajectory starting from point (x_1, x_2).

The observer on the left side was trained on the gaussian distributed initial conditions $\mathcal{X}$ where $\mathcal{X}_1 = [-1, 1] \times [-1, 1]$. Due to the boundedness property of reverse duffing oscillator, the observer did not see the data in the outer range during training, so the estimation error is large, while the right side observer was trained on $\mathcal{X}_2 = [-3, 3] \times [-3, 3]$ and has learned the data in this range during the training process and has a significantly smaller error. Figure 5 illustrated the estimated state trajectories under the initial condition $x(t_0) = (2, 0)$ which is outside the training domain of the first observer while inside that of the second observer. Unsurprisingly, the results indicate the extremely limited generalization capability which is consistent with the results in Ramos et al. (2020). The PDE constraint (5) can also be added to the loss function to constrain the learning of T as Niazi et al. (2022) inspired to improve the generalization, but since the effect is not quite significant, the results are not shown here.
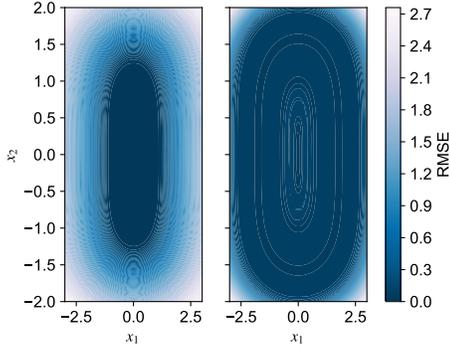
9

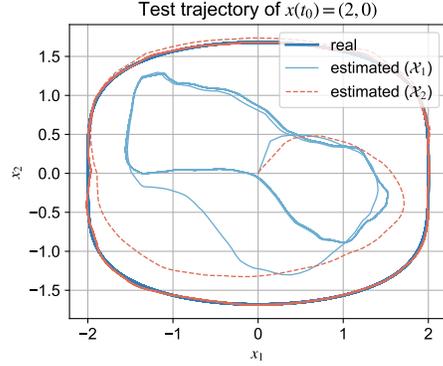Figure 4: RMSE mapping with different training domain

Figure 5: Test trajectory with different training domain

Remark 7 *As introduced in Bernard and Andrieu (2019), a stationary transformation $\mathcal{T}$ and $\mathcal{T}^*$ for an autonomous system can be used for time-varying systems*

$$\dot{x} = f(x) + g(x)u \qquad (24)$$

where the latent state z satisfies

$$\dot{z} = Dz + Fy + \varphi(z)u, \quad \varphi(z) = \frac{\partial \mathcal{T}}{\partial x}(\mathcal{T}^*(z))g(\mathcal{T}^*(z)) \qquad (25)$$

In this case, mapping $\mathcal{T}$ is also needed, hence the loss function for training autonomous system observer is modified as $\ell = \int_{t_0}^{t_f} |z(t) - \mathcal{T}(x(t), \eta_1)|^2 +$

Figure 6: Test trajectory of oscillator with excitation

$|x(t) - \mathcal{T}^*(\mathcal{T}(x(t), \eta_1), \eta_2)|^2 dt$. Finally, the observer learned by our approach can be easily implemented to a non-autonomous system with excitation. When the system (23) has an excitation as $\dot{x}_2(t) = -x_1(t) + u(t)$, $u(t) = \cos(12*t)$, the previously learned observer is shown in Figure 6 which still exhibits satisfactory performance.
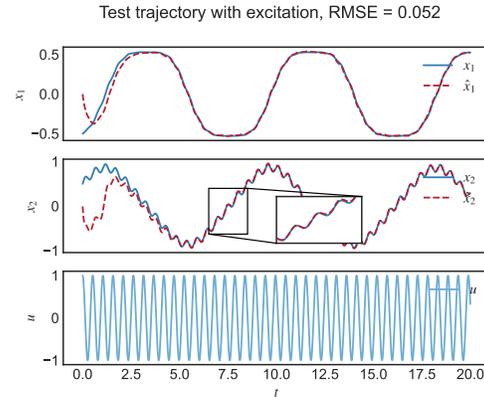
6. Conclusions

In this paper, we propose a Neural ODEs-based framework to design nonlinear state observers and cast the design problem into an optimal control problem. In particular, for KKL observer, we discuss the relationship between the observer design and its trade-off between convergence speed and robustness, and then use Neural ODEs to learn more robust observers than those previously addressed. Besides, our framework can then be extended to non-autonomous systems by applying stationary transformation. We have evaluated our approach through numerical simulations and the results show that our method outperforms the existing baseline.

There are also a number of outstanding problems to be addressed in the future. One of the most notable problem is that current learning-based KKL observers all suffer from poor generalization ability, and theoretical guarantees of a certain level of generalization remains an open problem. The application of observer methods to output prediction and system identification problems is also a direction of interest.

References

F. Abdollahi, H.A. Talebi, and R.V. Patel. A stable neural network-based observer with application to flexible-joint manipulators. *IEEE Transactions on Neural Networks*, 17(1):118–129, 2006.

Vincent Andrieu. Convergence speed of nonlinear luenberger observers. *SIAM Journal on Control and Optimization*, 52(5):2831–2856, 2014.

Vincent Andrieu and Laurent Praly. On the existence of a kazantzis–kravaris/luenberger observer. *SIAM Journal on Control and Optimization*, 45(2):432–456, 2006.

Pauline Bernard. Observer design for nonlinear systems. In *Lecture Notes in Control and Information Sciences*. Springer International Publishing, 2019.

Pauline Bernard and Vincent Andrieu. Luenberger observers for nonautonomous nonlinear systems. *IEEE Transactions on Automatic Control*, 64(1):270–281, 2019.

Chris M. Bishop. Training with noise is equivalent to tikhonov regularization. *Neural Computation*, 7(1):108–116, 1995.

G. Bornard and H. Hammouri. A high gain observer for a class of uniformly observable systems. In *Proceedings of the 30th IEEE Conference on Decision and Control*, volume 2, 1991.

Mona Buisson-Fenet, Lukas Bahr, Valery Morgenthaler, and Florent Di Meglio. Towards gain tuning for numerical kkl observers. *arXiv preprint arXiv:12204.00318*, 2022a.

Mona Buisson-Fenet, Valery Morgenthaler, Sebastian Trimpe, and Florent Di Meglio. Recognition models to learn dynamics from partial observations with neural odes. *arXiv preprint arXiv:2205.12550*, 2022b.

Ricky TQ Chen, Yulia Rubanova, Jesse Bettencourt, and David K Duvenaud. Neural ordinary differential equations. *Advances in neural information processing systems*, 31, 2018.

Franck Djeumou, Cyrus Neary, Eric Goubault, Sylvie Putot, and Ufuk Topcu. Neural networks with physics-informed architectures and constraints for dynamical systems modeling. In *Proceedings of The 4th Annual Learning for Dynamics and Control Conference*, volume 168 of *Proceedings of Machine Learning Research*, pages 263–277. PMLR, 23–24 Jun 2022a.

Franck Djeumou, Cyrus Neary, Éric Goubault, Sylvie Putot, and Ufuk Topcu. Taylor-lagrange neural ordinary differential equations: Toward fast training and evaluation of neural odes. In *International Joint Conference on Artificial Intelligence*, 2022b.

Weinan Ee. A proposal on machine learning via dynamical systems. *Communications in Mathematics and Statistics*, 5:1–11, 02 2017.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

Nikolaos Kazantzis and Costas Kravaris. Nonlinear observer design using lyapunov's auxiliary theorem. *Systems & Control Letters*, 34(5):241–247, 1998.

Patrick Kidger. On neural differential equations. *arXiv preprint arXiv:2202.02435*, 2022.

Qianxiao Li, Ting Lin, and Zuowei Shen. Deep learning via dynamical systems: An approximation perspective. *Journal of the European Mathematical Society*, 2022.

Timothy P Lillicrap, Jonathan J Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. Continuous control with deep reinforcement learning. *arXiv preprint arXiv:1509.02971*, 2015.

David G. Luenberger. Observing the state of a linear system. *IEEE Transactions on Military Electronics*, 8(2):74–80, 1964.

Stefano Massaroli, Michael Poli, Jinkyoo Park, Atsushi Yamashita, and Hajime Asama. Dissecting neural odes. *Advances in Neural Information Processing Systems*, 33:3952–3963, 2020.

Keyan Miao and Konstantinos Gatsis. Learning robust state observers using neural odes (longer version). *arXiv preprint arXiv:2212.00866*, 2022.

Muhammad Umar B Niazi, John Cao, Xudong Sun, Amritam Das, and Karl Henrik Johansson. Learning-based design of luenberger observers for autonomous nonlinear systems. *arXiv preprint arXiv:2210.01476*, 2022.

Johan Peralez and Madiha Nadri. Deep learning-based luenberger observer design for discrete-time nonlinear systems. In *2021 60th IEEE Conference on Decision and Control (CDC)*, pages 4370–4375. IEEE, 2021.

LS Pontryagin. *Mathematical theory of optimal processes*. CRC press, 1987.

Louise da C Ramos, Florent Di Meglio, Valery Morgenthaler, Luís F Figueira da Silva, and Pauline Bernard. Numerical design of luenberger observers for nonlinear systems. In *2020 59th IEEE Conference on Decision and Control (CDC)*, pages 5435–5442. IEEE, 2020.

Konrad Reif and Rolf Unbehauen. The extended kalman filter as an exponential observer for nonlinear systems. *IEEE Transactions on Signal processing*, 47(8):2324–2328, 1999.

Ivan Dario Jimenez Rodriguez, Aaron Ames, and Yisong Yue. Lyanet: A lyapunov framework for training neural odes. In *International Conference on Machine Learning*, pages 18687–18703. PMLR, 2022.

Yulia Rubanova, Ricky TQ Chen, and David K Duvenaud. Latent ordinary differential equations for irregularly-sampled time series. *Advances in neural information processing systems*, 32, 2019.

Paulo Tabuada and Bahman Gharesifard. Universal approximation power of deep residual neural networks via nonlinear control theory. *arXiv preprint arXiv:2007.06007*, 2020.

Muhammad Zakwan, Liang Xu, and Giancarlo Ferrari-Trecate. Robust classification using contractive hamiltonian neural odes. *IEEE Control Systems Letters*, 7:145–150, 2022.

Juntang Zhuang, Nicha Dvornek, Xiaoxiao Li, Sekhar Tatikonda, Xenophon Papademetris, and James Duncan. Adaptive checkpoint adjoint method for gradient estimation in neural ode. In *International Conference on Machine Learning*, pages 11639–11649. PMLR, 2020.