Deep Learning-based Luenberger observer design for discrete-time nonlinear systems

Johan Peralez, Madiha Nadri

Deep Learning-based Luenberger observers design for discrete-time nonlinear systems

Johan Peralez[1], Madiha Nadri[1]

***Abstract*— In this paper we address the problem of observer design for nonlinear discrete-time systems. Combining the theory of so-called Kazantzis–Kravaris-Luenberger (KKL) observers and Deep Learning, we aim to identify the mapping which transforms a nonlinear dynamics to a stable linear system modulo an output injection and design an asymptotic discrete-time observer. The proposed approach leverages the power of Machine Learning to provide an algorithm based on an unsupervised learning of the mapping, which allows to properly explore the state space.**

The approach is illustrated on two examples of the autonomous case and two of the non-autonomous one. These examples have been taken from the literature and judiciously chosen to compare the proposed approach with existing results.

I. INTRODUCTION

Online estimation of the state of a dynamical system is a crucial problem with numerous practical applications, especially in monitoring or control. Despite the interest given to the problem of observer design for nonlinear systems, only a few results can be found in the literature for discrete-time systems. In a deterministic context, the Extended Kalman Filter (EKF) is one of the most widely explored methods in academic and industrial literature, as it attempts to handle process nonlinearities and addresses a large class of systems ([1], [2], [3]). The extended Luenberger observer has also been used. However, it is known that these approaches rely on linearization methods, and thus provide local convergence only.

An alternative strategy dealing with strong non-linearities in observer design is to identify a mapping, which makes nonlinear dynamics approximately linear or in canonical forms [4], [5]. These transformations have the potential to allow observer design for nonlinear systems using standard linear theory. However, this still is a challenge in the observer design field and particularly for discrete-time. Note that most of the approaches concerning the class of continuous-time Lipschitz systems cannot be transposed directly to the case of discrete-time systems.

Luenberger's initial methodology has also been applied for nonlinear time-continuous systems leading to the so-called Kazantzis-Kravaris-Luenberger (KKL) observers. This method is based on an immersion of nonlinear systems into linear systems modulo an output injection [6]. This line of work has been transposed to the discrete-time nonlinear case by N. Kazantzis and C. Kravaris in [7] and more recently extended in [8] by considering weaker assumptions and designing a global observer. While promising, the difficulty here is the need to synthesize the coordinate transformation, which makes industrial applications challenging [9].

In this work, we will address this problem in a data-driven way and benefit from the power of Deep Learning to express this change through deep neural networks trained from large amounts of data. Deep neural network architectures have shown to be able to learn non-linear mappings and to generalize to unseen data under certain conditions for a diverse range of problems. Different variants of the universal approximation theorem guarantee that a multi layer perceptron can express any arbitrary function, including our desired mappings, under mild conditions either for infinitely wide [10] or infinitely deep (i.e layered) [11] model architectures.

A first attempt for the design of KKL observers for nonlinear systems based on neural networks has been reported in [12]. Here, the mapping is identified by a supervised learning method which does not guarantee a proper exploration of the state space during training, especially for systems which quickly converge to a limit cycle.

To address these shortcomings, and in contrast to [12], we propose a formulation based on unsupervised learning of the mapping, which allows to properly explore the state space. Experimental results confirm the benefits in term of performance. Moreover, for non-autonomous systems, our approach apply to stationary transformation first introduced in [6] that do not require a specific, well-chosen excitation.

The paper is organized as follows. The problem statement and some technical details concerning the KKL observer framework are given in Section II. In Section III, the structure of the proposed neural model and the associated training algorithm are developed for autonomous and non autonomous cases. We evaluate the method on numerical simulations of four examples providing results and comparisons to competing approaches of the literature.

II. PROBLEM STATEMENT AND PRELIMINARY RESULTS

A. Problem statement

In this paper, we consider nonlinear discrete-time systems of general form

$$\begin{cases} x_{k+1} = F(x_k, u_k) \\ y_k = h(x_k) \end{cases} \tag{1}$$

where the state $x \in \mathbb{R}^{d_x}$, $u \in \mathbb{R}^{d_u}$, the control input, $y \in \mathbb{R}^{d_y}$ the output, F and h are suitable functions, (0,0) is an equilibrium pair, i.e. $F(0, 0) = 0$, and $h(0) = 0$. We

[1] Univ Lyon, Université Claude Bernard Lyon 1, CNRS, LAGEPP UMR 5007, 43 boulevard du 11 novembre 1918, F-69100, VilleurbaNeural Networke, France.

denote $X_k\left(x_0, u_{[0,k]}\right)$ the value at time k of the unique solution of system (1) initialized at $x_0 \in \mathbb{R}^{d_x}$, with input $u_{[0,k]}$. Let a subset $\mathcal{X}_0 \subset \mathcal{X} \subset \mathbb{R}^n$ such that for all initial conditions $x_0 \in \mathcal{X}_0$ and all $k \in \mathbb{N}$, $X_k\left(x_0, u_{[0,k]}\right) \in \mathcal{X}$.

For the rest of the paper, we consider the case in which the vector fields F and h satisfy the following assumptions.

Assumption 1: F is invertible and F^{-1} and h are of class C^1 and globally Lipschitz.

For all non-negative integers i we denote $\circ$ the composition operator and

$$F^i = \underbrace{F \circ F \circ \cdots \circ F}_{i \text{ times}}, \quad F^{-i} = \left(F^{-1}\right)^i.$$

Assumption 2: For all $(x_1, x_2) \in \mathcal{X}^2$ of (1) with input u, if $x_1 \neq x_2$, then there exists a positive integer i such that $h(F^{-i}(x_1)) \neq h(F^{-i}(x_2))$. This means that for any input u of interest, there exists an open bounded set $\mathcal{O}$ containing $\mathcal{X}$ such that (1) is backward $\mathcal{O}$ -distinguishable on $\mathcal{X}$.

We address the following problem: for any input $u \in \mathcal{U}$ a set of interest, design a KKL observer to online estimate x_k from the knowledge of a sequence of the past and current values of the output y_k and input u_k.

Our *Ansatz* consists in finding an injective and continuous map T which transforms the system (1) into a Hurwitz form. Then, thanks to the contraction properties, the implementation of this new dynamic with any initial condition then provides an asymptotically convergent estimate of $T(X(x_0, u))$. Since T is injective, an estimate of the solution $X(x_0, u)$ can be obtained. However, even if sufficient conditions for the existence of this map have been given for a general form of nonlinear continuous systems, and consequently of the observer, a computable solution is unfortunately often difficult or impossible to obtain.

In this work, we rely on theoretical results on the existence of this transformation ([8] [6]), to address the problem of computing the map T and its pseudo inverse T^{-1} in the autonomous and non autonomous cases.

Before introducing our methodology let's recall the theoretical results needed for the observer design.

B. Preliminary results

Consider a particular class of system (1), given by the following autonomous system

$$\begin{cases} x_{k+1} = F(x_k, 0) \\ y_k = h(x_k) \end{cases} \tag{2}$$

Following the Luenberger-like methodology, the authors in [8] have shown that, if Assumptions 1 and 2 hold, for almost any controllable pair (A, B) of dimension $d_z := d_y\left(d_x + 1\right)$ with A Hurwitz there exists a map $T : \mathcal{X} \to \mathbb{R}^{d_z}$ that satisfies

$$T(F(x)) = AT(x) + Bh(x) \quad \forall x \in \mathcal{X}, \tag{3}$$

and a pseudo-inverse T^{-1} such that the following system is an observer for (2)

$$\begin{aligned} z_{k+1} &= Az_k + By_k & (4)\\ \hat{x}_k &= T^{-1}(z_k). & (5) \end{aligned}$$

The unique solution of (3) is given by

$$T(x) = \sum_{i=0}^{+\infty} A^i Bh\left(F^{-(i+1)}(x)\right). \tag{6}$$

Based on results obtained in [8], we derive the following statement.

Theorem 1: Assume that assumptions 1-2 hold for system (2) and let $d_z = d_y\left(d_x + 1\right)$. Then, there exists a set $\mathcal{S}$ of zero measure in $\mathbb{C}^{d_z}$ such that for any matrix $A = Diag\left(\lambda_1, \ldots, \lambda_{d_z}\right) \in \mathbb{R}^{d_z \times d_z}$ with eigenvalues $\left(\lambda_1, \ldots, \lambda_{d_z}\right)$ in $\mathbb{C}^{d_z} \backslash S$ and $\rho(A) < 1$ and B the vector $(1, \ldots, 1)^\top$ in $\mathbb{R}^{d_x}$ there exists an injective mapping $T : \mathcal{X} \to \mathbb{R}^{d_z}$ and a left inverse T^{-1} such that the trajectories of (2) remaining in $\mathcal{X}$ and the trajectories of (4) satisfy

$$\lim_{k \to +\infty} \left|T\left(X_k\left(x_0\right)\right) - Z_k\left(x_0, z_0\right)\right| = 0,$$

$$\lim_{k \to +\infty} \left|X_k\left(x_0\right) - T^{-1}\left(Z_k\left(x_0, z_0\right)\right)\right| = 0,$$

where $|\cdot|$ refers to the usual Euclidean norm and $Z_k\left(x_0, z_0\right)$ denote the value at time k of the unique solution of system (4) with initial condition $z_0 \in \mathbb{R}^{d_z}$.

Unfortunately, in most situations the function T given by (6) is difficult to compute and an explicit expression for its pseudo inverse is rarely available. Therefore, our objective in the next section is to rely on the results of Theorem 1 to develop an algorithm to numerically identify these mappings.

III. Deep Learning and observer design

Since the existence of the change of coordinate for the KKL observer is guaranteed for the autonomous system under the previous assumptions, we first propose a constructive method to learn the mappings T and T^{-1} from data. The central question of machine learning is generalization, which studies the errors made by models when providing predictions on data unseen during training. When learning the mapping T, it is thus of uttermost importance, that a sufficiently representative portion of the state space be seen during training. This is one of the main advantages of our unsupervised training algorithm, which allows control over the exploration of the state space. We show that, under additional assumptions, the learned mappings can be used for designing an observer for the system subject to excitation.

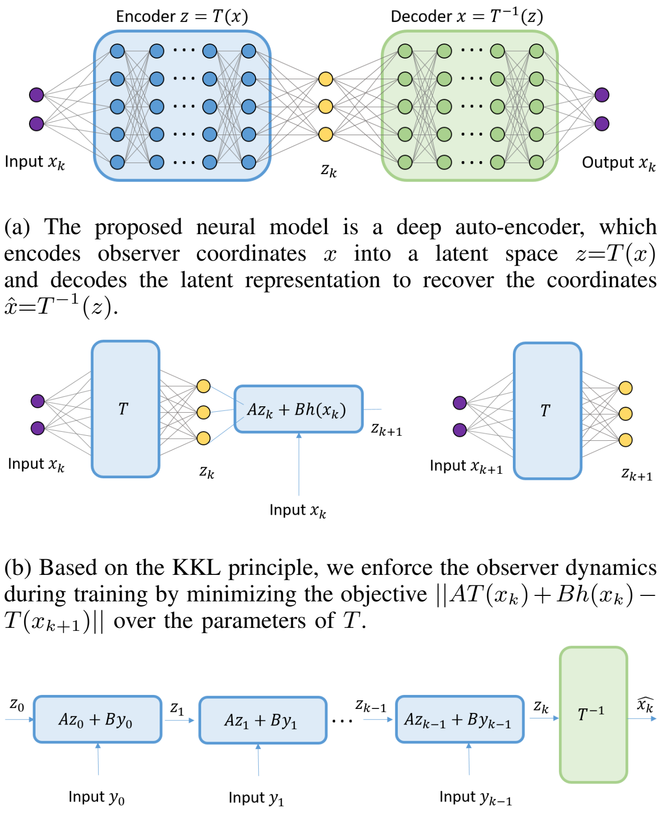
A. Autonomous systems

The network architecture of our proposed model is shown in Fig. 1. The objective of this network is to identify the mapping T that satisfies (3) along with its pseudo-inverse T^{-1}. We enforce two high-level requirements corresponding to two different loss functions during training:

- We want to identify a latent space $z = T(x)$, which respects the observer dynamics given by equation (3). We enforce this dynamics using the following loss:

$$\mathcal{L}_{dyn} = ||T(x_{k+1}) - (AT(x_k) + Bh(x_k))||. \tag{7}$$

- We seek to learn the pseudo-inverse T^{-1} so that an estimation $\hat{x}$ of the state is be recovered, which is achieved



(a) The proposed neural model is a deep auto-encoder, which encodes observer coordinates x into a latent space $z{=}T(x)$ and decodes the latent representation to recover the coordinates $\hat{x}{=}T^{-1}(z)$.

(b) Based on the KKL principle, we enforce the observer dynamics during training by minimizing the objective $||AT(x_k)+Bh(x_k) - T(x_{k+1})||$ over the parameters of T.

(c) The resulting observer structure.

Fig. 1: Diagram of our learning based method to identify the change of coordinates $z_k = T(x_k)$ and its pseudo-inverse T^{-1}.

using an reconstruction loss of the auto-encoder (see Fig. 1a), where T is the encoder and T^{-1} is the decoder:

$$\mathcal{L}_{recon} = ||x_k - T^{-1}(T(x_k))||. \tag{8}$$

The norm above $||\cdot||$ is mean-squared error.

Training — We train the model minimizing loss functions (7)-(8) on a large dataset $D = \{x_k, x_{k+1}\}$, which is generated automatically from the considered systems. To allow a proper exploration of the state space, samples x_k are obtained from a uniform random distribution on $\mathcal{X}$ while samples x_{k+1} are obtained by applying the model dynamic (2).

The data are split into training and validation sets. The models are trained on the training set, while hyper-parameters (network architectures, learning rates, early stopping etc.) are optimized over the validation set to avoid overfitting. We evaluate on a test set, different from training and validation, detailed in section IV. More details on network architectures and training hyper-parameters are given in Appendix VI-A.

A difficulty arising with the chosen learning scheme is that the values of the latent representation $z = T(x)$ in losses (7)-(8) are not part of the dataset D. It is therefore not possible to scale them, which is an important step in machine learning ensuring that neural networks operate in a favorable regime of their point-wise non-linearities (see e.g [13, Chapter 8]). This is a stringend problem, as, depending on the studied system (2) and on the choices of A and B, the observer variables can take very large values, and they can also be clustered around very small values. We found in our experiments that this can have a large impact on training.

In what follows we show that z can be scaled during the learning phase by adapting B. Taking

$$\tilde{B} = diag([b_1, b_2, ...])B \tag{9}$$

where $b_1, b_2, ... \in \mathbb{R}$ and denoting $\tilde{T}$ the mapping corresponding to $\tilde{B}$, from (6) we can see that

$$\begin{aligned} \tilde{T}(x) &= \sum_{i=0}^{+\infty} A^i diag([b_1, b_2, ...])Bh(f^{-(i+1)}(x)) && (10)\\ &= diag([b_1, b_2, ...])T(x). && (11) \end{aligned}$$

The algorithm is summarized Algorithm 1[1]. Notice that B is iteratively adapted to standardize z with respect to its standard deviation (lines 5-9).

Algorithm 1: Learning T and T^{-1}

```
1  Choose $A$ and $B$ for the observer system
2  Sample a set $D$ of transitions $\{(x_k, x_{k+1})\}$ from the
     autonomous system
3  Initialize $T$ network parameters $\Theta$ at random
4  for $iter \leftarrow 1$ to $iter^{max}$ do
       // Update B
5      Compute $\{z_k = T(x_k)\}$ for each $x_k \in D$
6      for $i \leftarrow 1$ to $d_z$ do
7          Compute $b_i$ as the inverse of the standard
             deviation of $\{z_{i,k}\}$
8      end
9      $B \leftarrow diag([b_1, b_2, ...])B$
       // Update $T$ network
10     Compute $\mathcal{L}_{dyn}$ following (7)
11     Update $\Theta$ using stochastic gradient descent to
         minimize $\mathcal{L}_{dyn}$
12 end
13 Initialize $T^{-1}$ network parameters $\Theta'$ at random
14 for $iter' \leftarrow 1$ to $iter'^{max}$ do
       // Update $T^{-1}$ network
15     Compute $\mathcal{L}_{recon}$ following (8)
16     Update Θ' using stochastic gradient descent to
         minimize $\mathcal{L}_{recon}$
17 end
```

B. Non-autonomous systems

We now turn our attention to non-autonomous systems and consider system (1) and u^0 a constant input such that

[1] Code is available at https://github.com/jolindien-git/DeepKKL

Assumptions 1 and 2 hold. Then, $F(x_k, u^0)$ is autonomous and from Theorem 1 there exists a map T solution to

$$T(F(x_k, u^0)) = AT(x_k) + By_k. \tag{12}$$

Adding $T(F(x_k, u_k))$ to each side of previous equation, this becomes

$$T(F(x_k, u_k)) = AT(x_k)+By_k+T(F(x_k, u_k))-T(F(x_k, u^0)). \tag{13}$$

Hence along solutions to (1), $z_k = T(x_k)$ evolves according to

$$z_{k+1} = Az_k + By_k + \Psi(z_k, u_k), \tag{14}$$

where

$$\Psi(z_k, u_k) = T(F(T^{-1}(z_k), u_k)) - T(F(T^{-1}(z_k), u^0)). \tag{15}$$

Note that, in the autonomous case, the term $\Psi(z_k, u_k)$ does not appear, the dynamics of z were contracting and it was enough to asymptotically simulate z with any initial condition. Unfortunately this is no longer true, but we have the following result.

Corollary 1: Let u^0 be a constant input such that Assumptions 1 and 2 hold. Assume that A can be designed such that T, T^{-1} given by Theorem 1 satisfy

$$\left|\Psi(z^1, u) - \Psi(z^2, u)\right| \leq C_u \left|z^1 - z^2\right|$$

for all $u \in \mathcal{U}$ and for all z^1, z^2. Then for u such that $\rho(A + C_u I) < 1$ we get that any solution to (1)-(14) verifies

$$\lim_{k\to+\infty} \left|x_k - T^{-1}(z_k)\right| = 0.$$

Proof: Let denote by e_k the estimation error $e_k = z_k - T(x_k)$.

T being injective it is sufficient to prove that $e_{k+1} = z_{k+1} - T(x_{k+1})$ converges geometrically towards zero. From (1)-(14)-(13) we have

$$\begin{aligned} e_{k+1} =& z_{k+1} - T(F(x_k, u_k) \\ =& A\left(z_k - T(x_k)\right) + \Psi(z_k, u_k) - \Psi(T(x_k), u_k) \end{aligned}$$

Then assuming that $\Psi(z_k, u_k)$ is Lipschitz[2], for small $\left|u_k - u^0\right|$ and for $\rho(A)$ small enough, system (14) is still a contraction and $z_{k+1} - T(x_{k+1})$ converges towards zero. ■

Hence the mappings T and T^{-1}, learned on the autonomous system following the procedure detailed in Section III-A, can be used to design the observer (14).

At this point it is worth to discuss and detail the major difference with previous work addressing a similar objective in a different way. In [12], the mapping T is identified in a supervised way: i) a set of initial conditions $\{(x_0, z_0)\}$ is sampled ii) trajectories of x and z are then computed using system and observer dynamics, and iii) the mapping $z = T(x)$ is learned with a fully supervised objective. The main drawback is that z_0 is not known *a priori*. Therefore, the authors rely on the stability of the observer, and therefore on the fact that z forgets its initial conditions, to eliminate the early stages (the burn-in phase) of the simulation from the dataset.

Unfortunately, this procedure does not guarantee a proper exploration of the state space during training, especially for systems which quickly converge to a limit cycle. In the case of non autonomous systems, the authors of [12] propose to select an input $u(t)$ that makes the system extensively explore the state space. Then a time varying mapping T_u (and so a time varying pseudo inverse T_u^{-1}) is identified.

Our method, on the other hand, avoids these limitations by design through unsupervised learning, and is therefore simpler to apply. First, there are not constraints on inputs, which can be selected to properly explore that state space. Furthermore, the task of learning T may be easier as it is not time dependant. More importantly, in the case of a time varying mapping, an identification should be available for the (possibly infinite) duration of the observer experiment.

IV. Simulation results

We illustrate and evaluate our approach through numerical simulations.

In Example 1, we explore a system, where an analytic expression of the mapping T is known. It demonstrates the ability of the unsupervised learning method to correctly learn this mapping as well as accurately estimate the state.

Example 2 is inspired from [12], and we show that our observer results significantly outperform the accuracy reported in this important baseline.

Examples 3 and 4 address non-autonomous systems, on which we assess the robustness of the observer in presence of measurement noise.

A. Autonomous systems

Example 1: We consider the following example with linear dynamic and polynomial output:

$$\left\{ \begin{array}{l} x_{1,k+1} = x_{1,k} + \delta_t x_{2,k} \\ x_{2,k+1} = x_{2,k} - \delta_t x_{1,k} \\ y_k = x_{1,k}^2 - x_{2,k}^2 + x_{1,k} + x_{2,k} \end{array} \right. \tag{16}$$

where we consider a step time $\delta_t = 0.01$s. It can be shown that this system is weakly differentially observable of order 4 and [8] gives an analytic expression of T for an observer in $\mathbb{R}^3$.

Applying the methodology presented on section III-A, samples x_k are obtained from an uniform random distribution on $\mathcal{X} = [-1, 1] \times [-1, 1]$. Figure 2 compares the mapping identified with the neural network and the analytic one for $A = diag([1 - \delta_t, 1 - 2\delta_t, 1 - 3\delta_t]$ and $B = [1, 1, 1]$. It illustrates the ability of our unsupervised method to identify the (unique) mapping T.

The resulting observer behavior is illustrated in Figure 3 for a given initial condition, exhibiting good performances. The observer accuracy is further evaluated on a batch of $1,000$ random initial conditions, which are different from the training and validation configurations. It can be seen in

[2] Notice that in [6], the authors give a proof in the case of continuous time under similar assumptions.

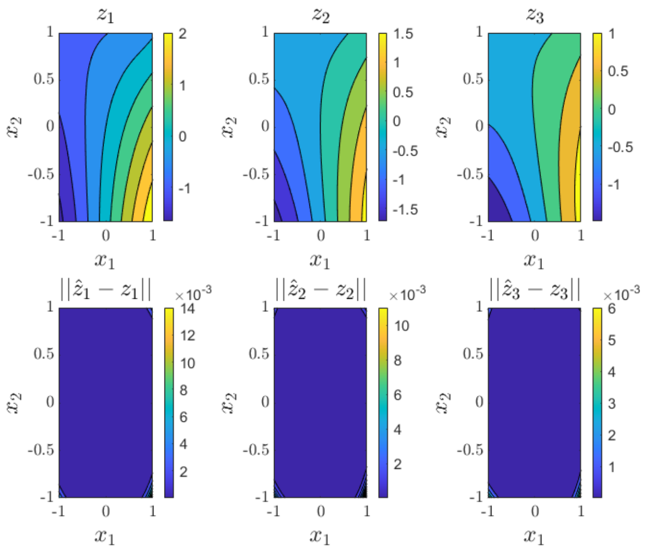


Fig. 2: Example 1: T transformation: z_i contours from analytic solution and their corresponding estimation errors $|\hat{z}_i - z_i|$.

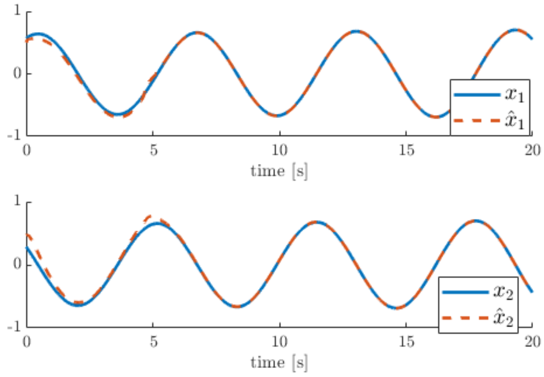


Fig. 3: Example 1: observer result for a given initial condition.

Figure 4 that the resulting mean error stabilizes after about 15s at a very low level.

Example 2: We consider the (discretized) system taken from [12]

$$\begin{cases} x_{1,k+1} = x_{1,k} + \delta_t x_{2,k}^3 \\ x_{2,k+1} = x_{2,k} - \delta_t x_{2,k} \\ y_k = x_{1,k} \end{cases} \tag{17}$$

with $\delta_t = 0.01$s. Results obtained in [12] highlight the difficulty to identify the mapping T especially around the origin. As in [12] we consider $\mathcal{X} = [-10, 10] \times [-4, 4]$. We also consider similar eigenvalues for the observer, corresponding to $A = diag([1 - 5\delta_t, 1 - 6\delta_t, 1 - 7\delta_t]$ in our discrete framework.

The resulting observer behavior is illustrated in Figure 5 for $x_0 = [1, 0]$. It can be compared to the results of [12] (Figure 6). Clearly, significant improvements are obtained. It can be supposed that this improvement is the result of a better exploration to the state space allowed by our method.

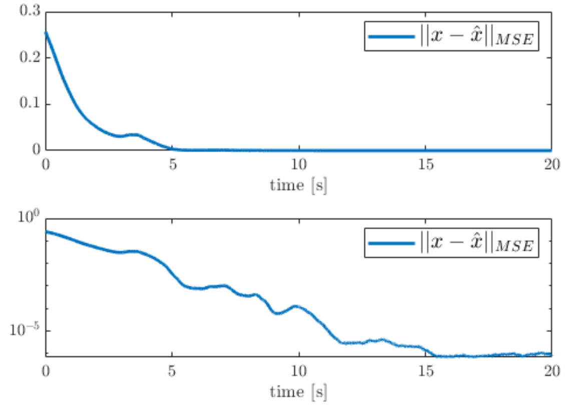


Fig. 4: Example 1: mean observer error on a batch of $1,000$ random initial conditions (second plot is on a semi-log scale).

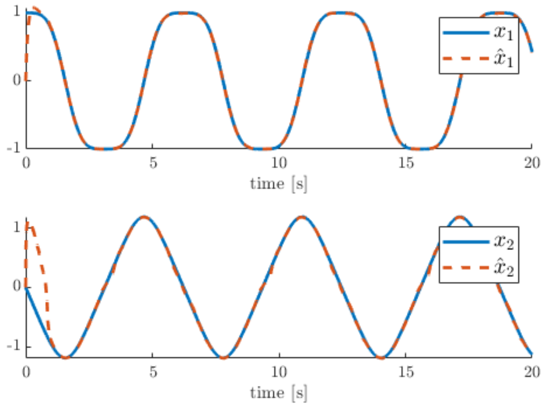


Fig. 5: Example 2: evolution of the true and estimated states over time.

B. Non Autonomous systems

Example 3: Adding an input to system (16), we consider

$$\begin{cases} x_{1,k+1} = x_{1,k} + \delta_t x_{2,k} \\ x_{2,k+1} = x_{2,k} - \delta_t x_{1,k} + \delta t u_k \\ y_k = x_{1,k}^2 - x_{2,k}^2 + x_{1,k} + x_{2,k}. \end{cases} \tag{18}$$

As explained in Section III-B, T and T^{-1} previously learned on the corresponding autonomous system (16) (i.e with $u_k = 0$) are used in the observer design (14).

An input $u_k = \cos(10k \times \delta t)$ is applied to (18). Despite this excitation, which significantly affects the dynamic, in Figure 6 it is shown that the observer still exhibits good performances, both during the transient phase and during asymptotic convergence.

Example 4: Finally we consider the following *Van der Pol* oscillator, which is a non-conservative oscillator with

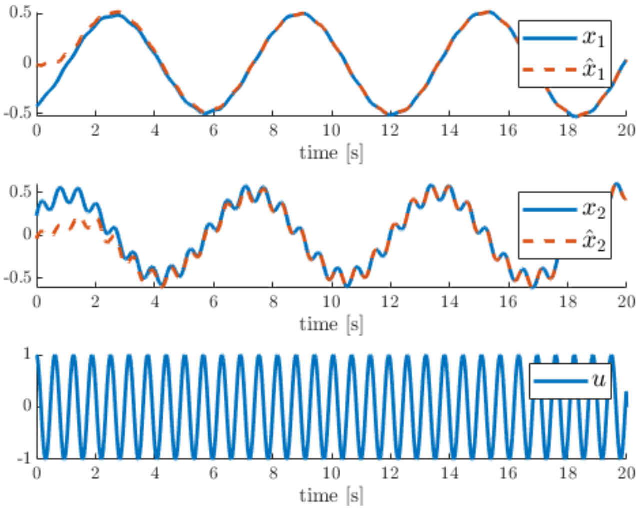


Fig. 6: Example 3: evolution of the true and estimated states over time

nonlinear damping

$$\begin{cases} x_{1,k+1} = x_{1,k} + \delta_t x_{2,k} \\ x_{2,k+1} = x_{2,k} + \delta_t \left((1 - x_{1,k}^2)x_{2,k} - x_{1,k} + u_k\right) \\ y_k = x_{1,k}, \end{cases} \tag{19}$$

with $\delta_t = 0.01$s.

As highlighted in [12], system (19) quickly converges to a stable limit cycle when unforced, motivating the authors to use time-varying mappings.

Here, as in the previous example, the observer (14) makes use of T and T^{-1} learned on the unforced system (i.e with $u_k = 0$).

Similarly to [12], during evaluation we consider an input signal $u_k = 0.44\cos(0.5k \times \delta_t)$. It is found that picking x_0 in $\mathcal{X}_0 = [-2, 2] \times [-3, 3]$, x_k remains in $\mathcal{X} = [-2.5, 2.5] \times [-3.5, 3.5]$. Hence, training data are uniformly sampled in $\mathcal{X}$ while validation experiments are conducted with $x_0 \in \mathcal{X}_0$.

To study the noise filtering properties of resulting observer, additional noise is considered on measurements (a Gaussian signal with standard deviation of 0.2 is added to y_k). As one can expect (see e.g [14, Chapter 6] for an an interesting study on a practical implementation of a KKL observer in presence of noise), the choice of A spectrum affects the capacity of noise rejection.

Figures 7-8 present experimental results with respectively $A = diag([1-0.5\delta_t, 1-0.6\delta_t, 1-0.7\delta_t])$ and $A = diag([1-5\delta_t, 1-6\delta_t, 1-7\delta_t])$. As expected, the later choice of A results in faster convergence properties but also in higher estimation errors in presence of measurement noise.

V. Conclusion

The approach proposed in this paper gives a theoretical framework to construct the mapping which transforms the original discrete-time nonlinear system into a stable linear one (modulo an output injection) to allow a KKL observer design. Based on existence results, we have shown that Deep Learning is relevant to identify this mapping when an explicit expression is not available. The approach has been illustrated and evaluated through numerical simulations which showed that our observer results significantly outperform existing baselines.

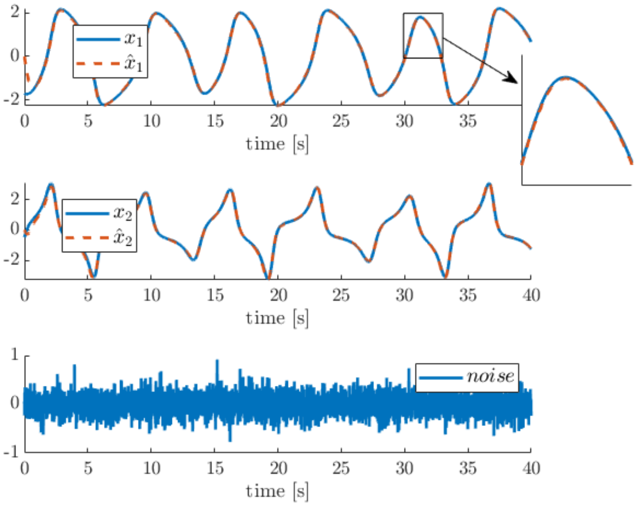


Fig. 7: Example 4 (Van der Pol oscillator): noise rejection for $A = diag([1 - 0.5\delta_t, 1 - 0.6\delta_t, 1 - 0.7\delta_t])$

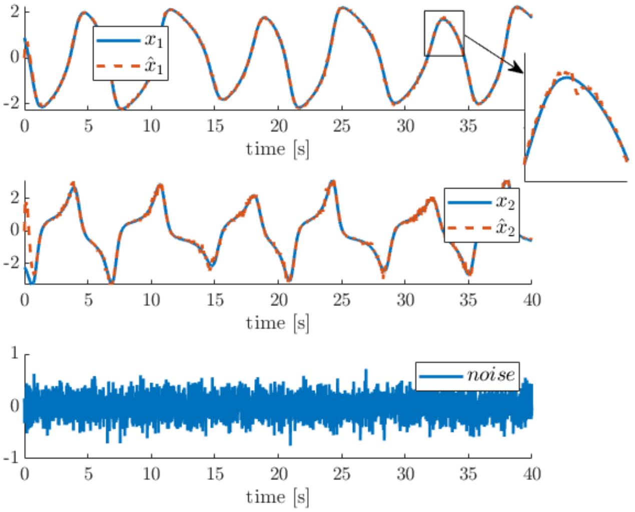


Fig. 8: Example 4 (Van der Pol oscillator): noise rejection for $A = diag([1 - 5\delta_t, 1 - 6\delta_t, 1 - 7\delta_t])$

Even if the performances of the proposed approach are promising, the design assumptions in non-autonomous case may be relaxed to provide more general results. This will be the aim of future work.

Furthermore, this work has opened up many work perspectives; among them the choice of optimal eigenvalues in the observer dynamic remains an open question. An interesting optimization problem would be to minimize the estimation error in presence of measurement noise. Another direction is to assess the proposed method on industrial applications and more specifically on high dimensional systems where the explicit solution is impossible to obtain.

VI. Appendix

A. Deep Learning details

Dataset creation. For each dynamical system, we choose $50,000$ initial conditions for the validation set, and $200,000$ for the training set. Initial state conditions are obtained randomly from a uniform distribution on the set of interest.

Code. The code has been implemented in python and the Pytorch framework for differentiable programming[15]. The code is available at `https://github.com/jolindien-git/DeepKKL`.

Hyperparameters. We use the Adam optimizer for training. The stochastic gradient descent is performed with batch sizes of 100. The learning rate is initialized at 0.001 and then reduced by a factor of 10 after every 10 epochs showing no loss improvement. Network architectures are also the same for each dynamical system. Classic Multilayer Perceptrons (also called Dense Neural Networks) are used to identify T and T^{-1}. Each hidden layer has the form of $Wx+b$ followed by an activation with the hyperbolic tangent function $tanh$:

$$T(x) = W_p\phi(W_{p-1}...\phi(W_2\phi(W_1x+b_1)+b_2)...)+b_p. \quad (20)$$

In our experiments, training was significantly improved with $tanh$ activation functions compared to rectified linear units (ReLU). Two hidden layers with a width of 500 neurons were used.

References

[1] M. Boutayeb and D. Aubry, "A strong tracking extended kalman observer for nonlinear discrete-time systems," *IEEE Transactions on Automatic Control*, vol. 44, no. 8, pp. 1550–1556, 1999.

[8] L. Brivadis, V. Andrieu, and U. Serres, "Luenberger observers for discrete-time nonlinear systems," in *2019 IEEE 58th Conference on Decision and Control (CDC)*. IEEE, 2019.

[2] K. Reif and R. Unbehauen, "The extended kalman filter as an exponential observer for nonlinear systems," *IEEE Transactions on Signal Processing*, vol. 47, no. 8, pp. 2324–2328, 1999.

[3] Y. Song and J. Grizzle, "The extended kalman filter as a local asymptotic observer for nonlinear discrete-time systems," *Journal of Mathematical Systems Estimation and Control*, vol. 5, no. 1, pp. 59–78, 1995.

[4] C. Califano, S. Monaco, and D. Normand-Cyrot, "Canonical observer forms for multi-output systems up to coordinate and output transformations in discrete time," *Automatica*, vol. 45, no. 11, pp. 2483–2490, 2009.

[5] G. Besançon, H. Hammouri, and S. Benamor, "State equivalence of discrete-time nonlinear control systems to state affine form up to input/output injection," *Systems Control Letters*, vol. 33, no. 1, pp. 1–10, 1998. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S016769119700090X

[6] P. Bernard and V. Andrieu, "Luenberger observers for nonautonomous nonlinear systems," *IEEE Transactions on Automatic Control*, vol. 64, no. 1, pp. 270–281, 2018.

[7] N. Kazantzis and C. Kravaris, "Discrete-time nonlinear observer design using functional equations," *Systems Control Letters*, vol. 42, no. 2, pp. 81–94, 2001.

[9] F. Poulain, L. Praly, and R. Ortega, "An observer for permanent magnet synchronous motors with currents and voltages as only measurements," in *2008 47th IEEE Conference on Decision and Control*. IEEE, 2008, pp. 5390–5395.

[10] K. Hornik, M. Stinchcombe, and H. White, "Universal approximation of an unknown mapping and its derivatives using multilayer feedforward networks," *Neural networks*, vol. 3, no. 5, pp. 551–560, 1990.

[11] Z. Lu, H. Pu, F. Wang, Z. Hu, and L. Wang, "The Expressive Power of Neural Networks: A View from the Width," in *NeurIPS*, 2017.

[12] L. d. C. Ramos, F. Di Meglio, V. Morgenthaler, L. F. F. da Silva, and P. Bernard, "Numerical design of luenberger observers for nonlinear systems," in *2020 59th IEEE Conference on Decision and Control (CDC)*. IEEE, 2020, pp. 5435–5442.

[13] C. M. Bishop, *Pattern recognition and machine learning*. Springer, 2006.

[14] N. Henwood, "Estimation en ligne de paramètres de machines électriques pour véhicule en vue d'un suivi de la température de ses composants," Ph.D. dissertation, Paris, ENMP, 2014.

[15] A. Paszke, S. Gross, F. Massa, A. Lerer, G. C. James Bradbury, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. K. Alban Desmaison, Z. D. Edward Yang, M. Raison, A. Tejani, S. Chilamkurthy, L. F. Benoit Steiner, J. Bai, and S. Chintala, "PyTorch: An imperative style, high-performance deep learning library," in *Advances in Neural Information Processing Systems (NeurIPS)*, 2019.