Numerical design of Luenberger observers for nonlinear systems

Louise da Costa Ramos, Florent Di Meglio, Valery Morgenthaler, Luís Fernando Figueira da Silva, Pauline Bernard

Numerical design of Luenberger observers for nonlinear systems

Louise da C. Ramos, Florent Di Meglio, Valéry Morgenthaler, Luís F. Figueira da Silva and Pauline Bernard

***Abstract*— In this paper, we propose a method to numerically design observers for nonlinear systems. The method relies on the theory of nonlinear Luenberger observers, which consist in mapping the nonlinear dynamics to a linear system, for which observer design is easy. Relying on results guaranteeing the existence of such mappings, we propose to approximate them by performing nonlinear regression on data simply generated by solving the system and observer dynamics. We detail different approaches for autonomous and excited systems and the choices made for data generation, pre-processing, and regression.**

I. Introduction

A. Context

In this paper, we propose a method to numerically design observers for nonlinear systems.

Observers are dynamical systems used to estimate the unmeasured states of a process, combining real-time data from sensors with a dynamical model of the process. There are few general design approaches for nonlinear systems. The popular Extended Filter [1] relies on linearization around the current estimate, yielding only local guarantees of convergence. High-gain observers [2], in turn, rely on strong assumptions on the observability of the system to map it to a triangular form in which the design is eased. A review of generic observer design methods for nonlinear systems is given in [3].

In a seminal paper [4], the original Luenberger observer design for linear systems is presented: it is shown that observable linear dynamics can be mapped, using an invertible variable change, to a linear, stable dynamical system having the measurement as input, in other words, to a stable linear filter of the output. Implementing this filter from any initial condition, then enables to recover a state estimate by inversion of this mapping. In [5], [6], the same idea of mapping the plant dynamics to a linear filter of its output is progressively extended to more and more general classes of nonlinear systems. In [7], it is shown that a general notion of observability, *backward distinguishability*, is enough to guarantee the existence of such an injective mapping. In [8], a similar result is obtained for non-autonomous nonlinear systems. Even if these results guarantee the existence of the mapping and its (pseudo-)inverse, they are not constructive, and the mapping has, often, no tractable analytical expression.

The main contribution of the paper is a numerical observer design for nonlinear systems that consists in approximating the mappings involved in the Luenberger methodology. The design solely relies on numerical simulations of the dynamics and the use of nonlinear regression. Our approach is as follows. For any chosen (stable) linear observer dynamics, we compute numerical solutions to both the system to be observed and the observer, fed by the corresponding output. We then perform a nonlinear regression on the data to compute an approximation of the mapping and its inverse. In the case of autonomous systems, the method straightforwardly uses the existence result from [7]. For a system with input, the approach consists in computing the mappings corresponding to a specific, well-chosen excitation, and guaranteed to exist by [8]. Indeed, we show that these mappings can then be used with other inputs, provided stronger observability assumptions are made and the observer dynamics are appropriately modified.

To perform the nonlinear regression, we rely here on Neural Networks. This machine learning approach allows the representation of a problem in multiple levels, obtained by composing and connecting simple but non-linear modules, also called neurons. These modules allow to transform the representation at one level into a representation at a higher and more abstract level [9]. Networks architectures composed by a sufficient number of these transformation modules have shown a high learning capability for diverse non-linear problems. For example, different neural network methodologies have been extensively used for image recognition [9]–[12], for language processing [13], self-learning control systems [14], for the prediction of chaotic series [15], or for the predictive modelling of nonlinear dynamical systems [16].

The paper is organized as follows. In Section I-B, we formulate the observer design problem. In Section II, we describe our approach for both autonomous and excited systems. In Section III, the architecture and learning approach of the neural network are presented. In Section IV, we illustrate our approach through numerical simulations on toy examples of autonomous and non-autonomous systems.

B. Problem Statement

Consider a system of general form

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

This work was supported by ANSYS Fr.

L. da Costa Ramos, F. Di Meglio and P. Bernard are with the Centre Automatique et Systèmes, MINES ParisTech, 75272 Paris, France louise.ramos@ansys.com, florent.di_meglio@minesparistech.fr, pauline.bernard@mines-paristech.fr

V. Morgenthaler is with the Research team, ANSYS Fr., 69100 Villeurbanne, Fr. valery.morgenthaler@ansys.com

L. F. Figueira da Silva is with the Faculty of Mechanical Engineering, PUC-Rio., 22541-041 Rio de Janeiro, Fr. luisfer@puc-rio.br

where $x \in \mathbb{R}^{d_x}$ is the state, the measured output is $y(t)$ in $\mathbb{R}^{d_y}$, f is a continuously differentiable function (C^1), h is a continuous function, and u an input in $\mathbb{R}^{d_u}$. For any input u of interest, we are interested in estimating online $x(t)$ from the knowledge of the past values of the output y and input u, under the following two assumptions.

Assumption 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: 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 $\mathcal{X}$, namely there exists $\bar{t} > 0$ such that for any trajectories x_a and x_b of (1) with input u and any $t \geq \bar{t}$ such that $(x_a(t), x_b(t)) \in \mathcal{X} \times \mathcal{X}$ and $x_a(t) \neq x_b(t)$, there exists $s \in [t - \bar{t}, t]$ such that

$$h(x_a(s)) \neq h(x_b(s))$$

and $(x_a(\tau), x_b(\tau)) \in \mathcal{O} \times \mathcal{O}$ for all $\tau \in [s, t]$. In other words, their respective outputs become different in backward finite time and before leaving $\mathcal{O}$.

Remark 1: For non-autonomous systems, we require the time $\bar{t}$ within which the outputs are distinguishable to be uniform with respect to the initial conditions in $\mathcal{X}$. This can be relaxed for autonomous systems, see [7, Definition 2].

Under these mild assumptions, it was shown in [8] that, for any input u of interest, and for almost any controllable pair (D, F) of dimension $d_z := d_y(d_x + 1)$ with D Hurwitz, there exists a map T_u^* such that the following system is an observer for (1)

$$\dot{z}(t) = Dz(t) + Fy(t) \tag{2a}$$
$$\hat{x}(t) = T_u^*(t, z(t)). \tag{2b}$$

In other words, any trajectory of (2) verifies

$$\lim_{t \to +\infty} |\hat{x}(t) - x(t)| = 0,$$

and implementing (2) with any initial condition provides an estimate of the plant's state after a certain time.

The existence of T_u^* relies on the existence and injectivity of a direct transformation $T_u : \mathbb{R} \times \mathbb{R}^{d_x} \to \mathbb{R}^{d_z}$ such that for any solution x to (1), the image $z(t) = T_u(t, x(t))$ evolves according to the dynamics (2a). Straightforward computations show that T_u must satisfy

$$\frac{\partial T_u}{\partial x}(t, x) f(x, u(t)) + \frac{\partial T_u}{\partial t}(t, x) = DT_u(t, x) + Fh(x, u(t)), \tag{3}$$

for all $(t, x) \in [0, +\infty) \times \mathcal{X}$. According to [8], a solution to (3) always exists, and the main difficulty is then to show that $T_u(t, \cdot)$ becomes injective after $\bar{t}$ defined in Assumption 2. For $t \geq \bar{t}$, $T_u^*(t, \cdot)$ can then be designed as a left-inverse of $T_u(t, \cdot)$. Note that, in the particular case where the system (1) is autonomous, the transformation can also be taken autonomous, namely $T : \mathbb{R}^{d_x} \to \mathbb{R}^{d_z}$ and $T^* : \mathbb{R}^{d_z} \to \mathbb{R}^{d_x}$.

Now, although the observer has been proven to exist for a wide category of systems, its implementation requires the knowledge of the map T_u^*. Unfortunately, an explicit expression is rarely available unless one knows how to find a solution T_u to (3) and build a left-inverse. The goal of this paper is therefore to develop a methodology to numerically compute this map T_u^* and, when needed, T_u and its Jacobian, for both autonomous and excited cases. This work relies on precise theoretical results of existence that are recalled in the next section.

II. Methodology

A. Learning Procedure: Autonomous Systems

Consider an autonomous system

$$\begin{cases} \dot{x}(t) &= f(x(t)) \\ y(t) &= h(x(t)) \end{cases}. \tag{4}$$

The following Theorem derived from [7] shows the existence of an autonomous transformation T^*.

Theorem 1 ([7]): Suppose Assumptions 1 and 2 hold. Define $d_z = d_y(d_x + 1)$. Then, there exists $\ell > 0$ and a set 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 $(\lambda_1, \ldots, \lambda_{d_z})$ in $\mathbb{C}^{d_z} \setminus S$ with $\Re\lambda_i < -\ell$, and any $F \in \mathbb{R}^{d_z \times d_x}$ such that (D, F) is controllable, there exists an injective mapping $T : \mathbb{R}^{d_x} \to \mathbb{R}^{d_z}$ and a pseudo-inverse $T^* : \mathbb{R}^{d_z} \to \mathbb{R}^{d_x}$ such that the trajectories of (4) remaining in $\mathcal{X}$ and any trajectory of

$$\dot{z} = Dz + Fy, \tag{5}$$

satisfy

$$|z(t) - T(x(t))| \leq M \, |z(0) - T(x(0))| \, e^{-\lambda_{min} t} \tag{6}$$

for some $M > 0$ and with

$$\lambda_{min} = \min \{|\Re\lambda_1|, \ldots, |\Re\lambda_{d_z}|\} \tag{7}$$

and

$$\lim_{t \to +\infty} |x(t) - T^*(z(t))| = 0. \tag{8}$$

Remark 2: Notice that if the observer is perfectly initialized, i.e. $z(0) = T(x(0))$, then one has $z(t) = T(x(t))$ and, consequently, $x(t) = T^*(z(t)),\ \forall t$.

The existence of the mappings being guaranteed by this theorem, we propose to compute a numerical estimate of T and T^* by generating a large set of (x, z) values and using a nonlinear universal approximator. More precisely, we proceed according to the following steps

1) choose of D and F for the observer system;
2) choose a set of initial conditions (x_0, z_0) ;
3) simulate (2a),(4) in forward time, generating a set of (x, z) pairs ;
4) perform nonlinear regression to find the mapping $x = T^*(z)$.

We now detail specific methods for each step and the rationale behind them.

a) Initial conditions: The choice of the initial conditions affects the distribution in the (x, z)-space of the data used for performing the regression. Ideally, the sampling should be refined where the functions T and T^* are not smooth. In the absence of *a priori* knowledge on their regularity, however, standard statistical sampling methods are used, such as latin hypercube sampling. One should notice that the initial distribution will be strongly modified by the dynamics of the system, in ways that cannot be predicted in advance. Therefore, the distribution of the actual $(x(t), z(t))$ data may be very different from that of the initial condition.

b) Selection of data points: At the second step, simulation data is generated with a numerical ODE solver over a finite time interval $[0, t_f]$, from all of the initial conditions chosen at the previous step. This data must be pre-processed before it can be used for regression. Indeed, since T is unknown, it is impossible to initialize the observer exactly as suggested in Remark 2. Rather, we rely on the stability of the observer and the fact that z "forgets" its initial condition. Inequality (6) suggests that it is reasonable to eliminate from the dataset the pairs $(x(t), z(t))$ for $t < \frac{k}{\lambda_{min}}$ with $k = 3$ or 5.

c) Nonlinear regression: To learn the mappings T and T^*, one could *a priori* choose any nonlinear regression methodology. A neural network is proposed and further discussed in section III for its simplicity of implementation and its ability to represent complex nonlinearities with a low number of parameters [9].

B. Learning Procedure: system with an excitation

Now in [8], Theorem 1 was extended to general non-autonomous systems (1).

Theorem 2 ([8]): Suppose Assumptions 1 and 2 hold with $\bar{t} \geq 0$. Define $d_z = d_y(d_x + 1)$. Then, for any input u of interest, there exists a set S of zero measure in $\mathbb{C}^{d_z}$ such that for any Hurwitz matrix $D \in \mathbb{R}^{d_z \times d_z}$ with eigenvalues $(\lambda_1, \ldots, \lambda_{d_z})$ in $\mathbb{C}^{d_z} \setminus S$ with $\Re\lambda_i < 0$, and any $F \in \mathbb{R}^{d_z \times d_x}$ such that (D, F) is controllable, there exist mappings $T_u : \mathbb{R} \times \mathbb{R}^{d_x} \to \mathbb{R}^{d_z}$ and $T_u^* : \mathbb{R} \times \mathbb{R}^{d_z} \to \mathbb{R}^{d_x}$ such that

1) $T_u(t, \cdot)$ and $T_u^*(t, \cdot)$ depend only on the past values of u on $[0, t]$,
2) $T_u(t, \cdot)$ is injective for all $t \geq \bar{t}$ with $T_u^*(t, \cdot)$ a left-inverse on $\mathcal{X}$,

and any trajectory of (4)-(5) with $x(t)$ remaining in $\mathcal{X}$ satisfies

$$|z(t) - T_u(t, x(t))| \leq M\,|z(0) - T_u(0, x(0))|\,e^{-\lambda_{min} t} \quad (9)$$

for some $M > 0$ and λ_{min} defined as in (7) and

$$\lim_{t \to +\infty} |x(t) - T_u^*(t, z(t))| = 0. \quad (10)$$

Theorem 2 differs from Theorem 1 only through the fact that T_u is time-varying and the eigenvalues no longer have to be sufficiently large. Quite expectedly, the time $\bar{t}$ after which $T(t, \cdot)$ becomes injective is the same as the backward-distinguishability time of Assumption 2.

The main difficulty in numerically estimating the map T_u^* is that it now depends on the input u. In the favorable case where u is known in advance (time-varying systems), it is enough to learn the map T_u^* associated to this input. Otherwise, let us consider the particular case of an input-affine system

$$\begin{cases} \dot{x}(t) = f(x(t)) + g(x(t))u(t) \\ y(t) = h(x(t)) \end{cases} \quad (11)$$

and a nominal input u^0 so that Assumptions 1 and 2 hold. Then, according to Theorem 2, there exist a map T_{u^0} solution to the PDE (3) with $u = u^0$ and $T_{u^0}^*$ its left-inverse after a certain time. Straightforward computations then show that along solutions to (11), $z(t) = T_{u^0}(t, x(t))$ evolves according to

$$\dot{z}(t) = Dz(t) + Fy(t) + \varphi(t, z(t))(u(t) - u^0(t)), \quad (12)$$

where

$$\varphi(t, z) = \frac{\partial T_{u^0}}{\partial x}\left(t, T_{u^0}^*(t, z)\right) g\left(T_{u^0}^*(t, z)\right) \ . \quad (13)$$

Similarly to the previous design, if we manage to estimate $z(t) = T_{u^0}(t, x(t))$ an estimate of $x(t)$ is then obtained thanks to $T_{u^0}^*(t, z(t))$. When the term $\varphi(t, z)$ was absent, the dynamics of z were contracting and it was enough to simulate z with any initial condition to obtain asymptotically an estimate. Unfortunately, this is no longer true, but we have the following result.

Corollary 1: Suppose Assumptions 1 and 2 hold and pick u^0 among the inputs of interest. Assume D, T_{u^0} and $T_{u^0}^*$ given by Theorem 2 are such that for all t, for all z_a, z_b,

$$|\varphi(t, z_a) - \varphi(t, z_b)| \leq L|z_a - z_b| \ .$$

If $\lambda_{min} > L|u - u^0|_\infty$, then any solution to (1)–(12) with x remaining in $\mathcal{X}$ verifies

$$\lim_{t \to +\infty} |x(t) - T_{u^0}^*(t, z(t))| = 0 \ . \quad (14)$$

Proof: Denoting $e(t) = T_{u^0}(t, x(t)) - z(t)$, compute the derivative of $e^\top e$ along trajectories. Then, apply the left-inverse $T_{u^0}^*$. ■

The problem of this contraction condition is that T_{u^0}, and thus λ_{min} depend on D. Therefore, it may not be enough to take D with λ_{min} sufficiently large. Actually, if satisfying this condition is possible for a given u, it means that we can observe the plant for any other input u' such that $|u' - u^0|_\infty \leq |u - u^0|_\infty$. Therefore, the plant should be observable or at least detectable for any such input. It is shown for instance in [8, Theorem 4] that when the plant is observable for any input and the drift system $\dot{x} = f(x)$ is differentially observable of order d_x, L can be bounded independently from λ_{min} and therefore, this observer works with $u^0 = 0$ if λ_{min} is sufficiently large.

Our proposed methodology is as follows;

1) choose D and F for the observer system;
2) choose one initial condition (x_0, z_0);
3) choose an excitation $u = u^0$ to excite (11)

4) simulate (2a),(11) with input u^0 in forward time, generating a set of (z, x, t) data points;
5) use nonlinear regression to find the mapping $z = T_{u^0}(x, t)$, and the associated left-inverse $x = T^*_{u^0}(z, t)$;
6) compute $\partial T_{u^0}/\partial x$;
7) then, for validation, simulate (12), (11) in forward time with a different input $u(t) \neq u^0(t)$.

We now detail specific methods for each step and the rationale behind them.

a) Initial conditions and input signal: Here, the choice of the initial condition is not dominant in the distribution in the (x, z)-space of the data used to perform the regression. The perturbation has a much more significant impact. The question of finding an open-loop control u generating 'rich' data is a classical question in system identification, and highly depends on the considered system. The objective, as in the autonomous case, are *a priori* to have many samples where the functions T and T^* are likely to be non-smooth. In the absence of intuition or a priori knowledge on these, the excitation should make the system explore as uniformly as possible the compact of interest.

b) Selection of data points and nonlinear regression: The generation and pre-processing of data and the nonlinear regression methodology are the same as specified in section II-A. In the next section, we present the regression method we use.

III. Neural Networks

One should notice that any nonlinear regression can potentially be applied to approximate the mappings. We chose here Neural Networks (NN). The basic architecture of the NN is composed of one input layer with $l_i + bias$ neurons, multi-hidden layers with $l_h + bias$ neurons each, and one output layer with l_o neurons, as shown in Figure 1. The output value computed by the network is denoted $h_\Theta(x)$, which depends on the input training values (x) and of the network weights (Θ). As base methodology to implementing a neural network

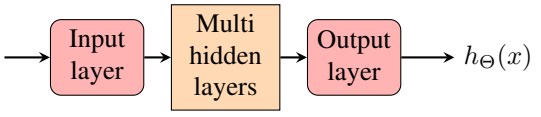


Fig. 1. Neural network simplified flow chart.

model, this work refers to [9], [17], [18].

Given a training dataset with input and output values, the "learning" of the neural network weights can be performed. A cost function (J in $\mathbb{R}$) is used, providing a quantitative measurement of the efficiency of the neural network with respect to a given training dataset [17]. This function can have different forms, and depends on the neural network weights (Θ), the training data set output (Y), and the predicted values with the network $(h_\Theta(X))$. The objective is to find a set of weights that minimizes the cost function and better fits the training data. For that, the back-propagation algorithm is used. For the back-propagation methodology and algorithm one may refer to [17], [18].

It is important to note that, for the back-propagation to work, some assumptions on the cost function [17] are needed ; (1) it can be written as an average $J = 1/m \sum_{i=1}^{m} J_i$ over cost functions (J_i) for individual training examples, which enables the computation of partial derivatives for each single training example, and (2) it can be written as a function of the outputs from the neural network, and thus is a function of the output activation.

A. Proposed Neural Network Methodology

In this work, the neural network architecture is composed of four layers $(L = 4)$: one input (layer (1)), two hidden layers (layers $(2, 3)$), and one output (layer (4)). The number of nodes of the input and output layers depends on the transformation, and are, respectively, d_x and d_y when the network is used to approximate T (and the opposite for T^*). The two hidden layers have 25 units each, which was chosen by trial-and-error, trading-off computational effort against accuracy. We use the following quadratic cost function

$$J_\theta = \frac{1}{2m} \sum_{i=1}^{m} \sum_{k=1}^{l_4} (h_\theta(x_k^i) - y_k^i)^2, \tag{15}$$

where m is the number of training data, l_4 is the number of nodes on the output layer $(L = 4)$, and h_θ is the predicted value from the neural network. Again, this particular choice has been made on a trial-and-error analysis, and is by no means optimal in any sense.

The activation function depends on each layer k. For the first layer, the activation is the proper input values (x) plus a bias $(a_0 = 1)$; $a^{(1)} = [a_0, x_1, \cdots, x_n]^T$. At the hidden layers, the hyperbolic tangent $(\tanh)$ is used as activation function $g(\phi)$, where ϕ is the layer weighted input, as shown in (16). However, a linear function is used at the output, i.e., each of the fourth layer neurons has its value multiplied by its respective weight.

$$g(\phi) = 2\left(\frac{1}{1 + e^{-2\phi}}\right) - 1, \quad g'(\phi) = 1 - g(\phi)^2. \tag{16}$$

In this work, the training methodology is:

1) randomly initialize the weights of each layer, where $\Theta^k \in \mathbb{R}^{l_{k+1} \times (l_k+1)}$, $\forall k \leq (L-1)$, and define three Θ matrices, $\Theta^{(1)} \in \mathbb{R}^{l_1 \times 26}$, $\Theta^{(2)} \in \mathbb{R}^{25 \times 26}$ and $\Theta^{(3)} \in \mathbb{R}^{l_4 \times 26}$.
2) effect the forward propagation to find a first prediction $h_\Theta(x)$.
3) compute the cost function with respect to $h_\Theta(x)$ and the training output values Y.
4) use the back-propagation to compute the partial derivatives of the cost function; $\frac{\partial}{\partial \Theta_i^{(k)}} J(\Theta, h_\Theta(X), Y)$.
5) apply $fmincg$, as optimization method, with the back-propagation to minimize the cost function and find Θ.

a) Optimization method: The $fmincg$ function ©[1] is an optimization routine used to minimize a continuous differentiable multivariate function. The initial condition is given

[1]Copyright (C) 2001 and 2002 by Carl Edward Rasmussen. Date 2002-02-13

by the all of weight values of the neural network, arranged at a $(n_w \times 1)$ vector, where n_w is the number of weights. The optimization is performed over these weights, by minimizing the cost function (eq. (15)). The Polack-Ribiere method of conjugate gradients is applied to define search directions. A quadratic and cubic polynomial approximations is used for line search and the Wolfe-Powell stopping criteria coupled with the slope ratio method is used to guess the initial step sizes.

In the next section, we apply this methodology to compute the transformation T_u and T_u^* of the previously presented Luenberger observer. Two neural networks need to be learned for solving this system; (a) one to solve $z = T_u(x)$, with the nonlinear system X as input and the observer system z as output, and (b) the other with the observer as input and the nonlinear system as output, solving $x = T_u^*(z)$.

IV. Numerical simulation

In this section, we illustrate our approach through numerical simulations on toy problems. Our main goal is to qualitatively highlight how the data selection and processing impacts the observer performance.

A. Autonomous System

Consider the following system

$$f(x) = \begin{cases} \dot{x}_1 = x_2^3 \\ \dot{x}_2 = -x_1 \end{cases} \qquad y = x_1 \tag{17}$$

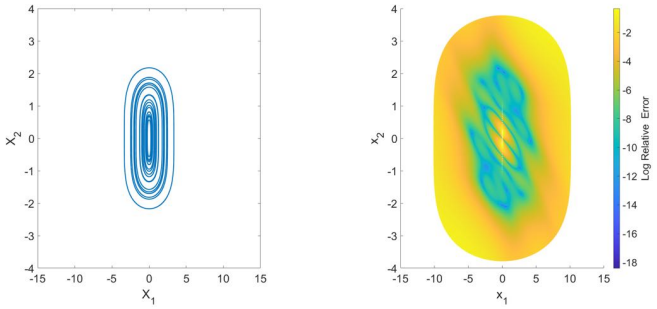
which admits bounded trajectories (where $x_1^2 + x_2^4$ is constant). This system is weakly differentially observable of order 2 in $\mathbb{R}^2$ since the mapping $x \to \mathrm{H}_2(x) = (x_1, x_2^3)$ is injective on $\mathbb{R}^2$, and so, it is considered a fortiori instantaneously backward-distinguishable [7], [8]. Applying then the Luenberger's methodology presented on section II-A, for an observer (2) in $\mathbb{R}^3$ with $D = diag([\lambda_1, \lambda_2, \lambda_3])$ and $F = \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix}$, T^* is computed with a neural network.

The training data is generated for different sets of initial conditions, chosen with different sampling methods. The dynamics are solved over the time interval $[0, t_f]$ with $t_f = 50$ s and a time step $dt = 10^{-2}$. In order to improve the neural network behavior, the data is centered around zero and normalized. We compute the approximation of the transformation $x = T^*(z)$ with the neural network methodology presented in Section III-A, where the input is $z = (z_1, z_2, z_3)$, and the output is $x = (x_1, x_2)$. A maximum of 1000 iterations is used.

We now highlight various design choices.

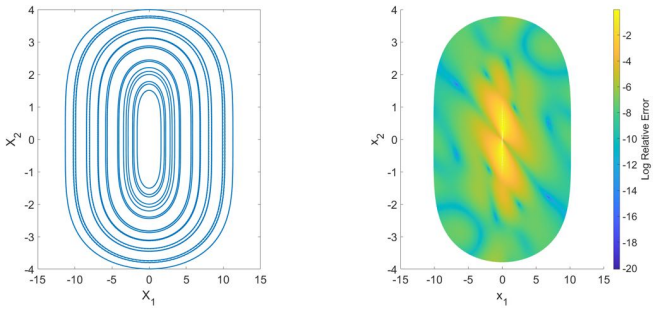
1) Impact of the size of the set $\mathcal{O}$: A critical design choice lies in the size of the subset of the state-space over which the mappings are to be approximated. There is an inherent trade-off: one would ideally want a set as large as possible, but this requires more training data and, most importantly, a more accurate nonlinear approximator. This choice is therefore intimately linked to the complexity of the neural network and to the number of points in the training dataset. For a fixed architecture, trying to cover a larger set has a negative impact on accuracy. This trade-off is illustrated on Figure 2, where the same number of points in the dataset are used for two different sets $\mathcal{O}$. For each figure, we plot the training dataset and the asymptotic logarithmic relative error

$$\lim_{t\to\infty} \log \frac{(x_1(t) - \hat{x}_1(t))^2 + (x_2(t) - \hat{x}_2(t))^2}{x_1^2 + x_2^2}. \tag{18}$$



(a) Trajectory for training $IC \in (-1, 1)$.

(b) Logarithmic relative error mapping for training $IC \in (-1, 1)$.



(c) Trajectory for training $IC \in (-10, 10)$.

(d) Logarithmic relative error mapping for training $IC \in (-10, 10)$.

Fig. 2. Impact of the size of the compact: Comparison of the autonomous system (eq. (17)) solutions for an observer defined in $\mathbb{R}^3$, with a different range for the Gaussian distributed initial conditions. The prediction test is over 100 equally distributed initial condition $(x_{1,0}^t, x_{2,0}^t) \in [-10, 10]$.

Unsurprisingly, both regressions qualitatively perform better in regions where more data points are present. The extrapolation capabilities are extremely limited, as illustrated on Figure 2b, which also illustrates the difficulty of estimating T^* around two symmetrical points close to the origin. This, in theory, should be linked to a small modulus of injectivity for T, which we depict on Figure 3.

In addition, in an attempt to increase the efficiency of the prediction over a large set one can increase the number of neurons and hidden layers. Accordingly, a network with a larger number of neurons has been tested, but no remarkable difference has been found, and for the sake of brevity, the results are not shown here.

2) Impact of the initial condition sampling: We now discuss the impact of the distribution of the initial conditions and the amount of data in the training set.

a) Distribution: Different standard methods for sampling can be used to choose the initial conditions for the

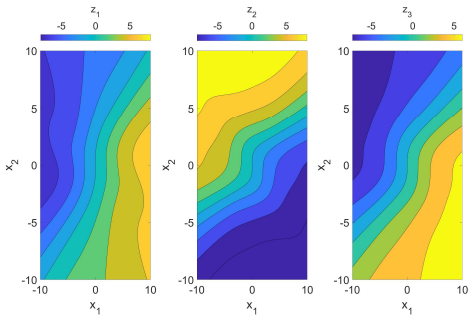
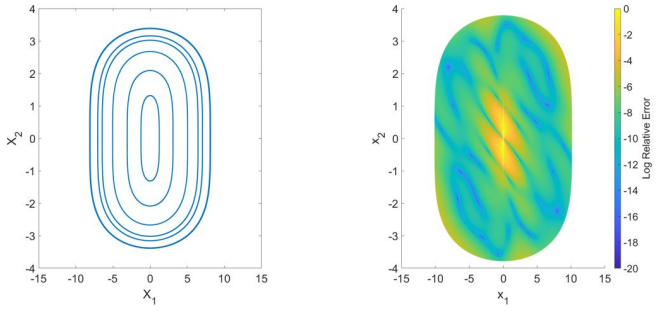


Fig. 3. Transformation T.

training set of data. Here, in a range of $[-10, 10]$, the impact of using a Gaussian distributed and a regular spaced distribution are shown. Figures 2c, 2d and 4a, 4b, respectively depict the results of each of this methods, for a training sample of 20 initial conditions.



(a) Trajectory for $IC \in (-10, 10)$.

(b) Logarithmic relative error mapping for $IC \in (-10, 10)$.

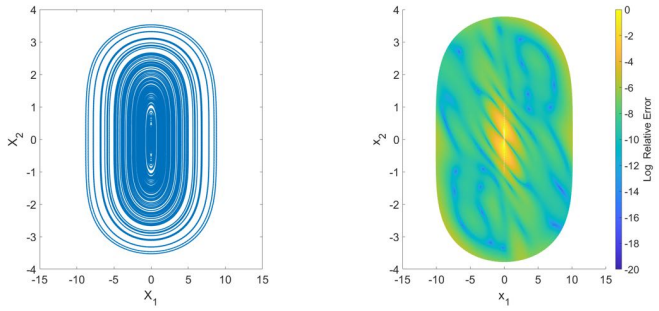
Fig. 4. Solution for the autonomous system (eq. (17)) with an observer z defined in $\mathbb{R}^3$, with regular partition used for choosing the 20 initial conditions $(x_{1,0}, x_{2,0}) \in (-10, 10)$.

Regarding the different distribution methods used to fill the parametric space of initial conditions, small differences are encountered when comparing the Gaussian distribution (Figure 2c and 2d) with the regular partitioning (Figure 4). In general, the same behaviour from Figure 2b is seen in Figure 4b, i.e., both present high error when predicting trajectories closer to $(x_1, x_2) = (0, 0)$.

b) Number of points: The impact of the initial conditions sampling is here tested with a training set computed over 100 different initial condition, with $x_{0,1} \in [-10, 10]$, chosen with a Gaussian distribution, and $x_2 = 0$. The corresponding result is depicted in Figure 5.

The use of a larger number of initial conditions better fills the training domain, as seen in Figure 5a. However, this does not result in a significant quantitative accuracy improvement. This point illustrates the need for either a smarter way to choosing the trajectories of interest, e.g. by using mesh refining methods, or for larger scale brute force methods.

3) Impact of the observer eigenvalues: Another critical design choice lies in the eigenvalues of the matrix D in Equation (2). These are linked to the convergence speed of



(a) Trajectory for 100 $IC \in (-10, 10)$.

(b) Logarithmic relative error mapping for 100 $IC \in (-10, 10)$.

Fig. 5. Solution for the autonomous system (eq. (17)) with an observer z defined in $\mathbb{R}^3$, with regular partition used for choosing the 100 initial conditions $(x_{1,0}, x_{2,0}) \in (-10, 10)$.

the observer, although they do not determine it entirely, as the mappings highly depend on their values. An interesting study of the effect of D on, e.g., the noise filtering properties of the observer is given in [19].

Here, we study the impact of the eigenvalues on the learning process. We compare the results obtained with, on one hand, 'arbitrarily' picked eigenvalues $(-5, -6, -7)$ and, on the other hand, eigenvalues corresponding to a third-order Bessel filter with a cut-off frequency of $2\pi\ rad/s$. The results are depicted at Figure 6. Although these early results are very partial, they seem to indicate that a physically sound choice of eigenvalues makes the learning process easier, for a similar convergence speed.

Future works should include a much deeper study of the impact of the eigenvalues on the transformation and the learning process, in particular, the effect on the convergence speed. Another potential direction would be to perform nonlinear regression, not only in the state and observer variables, but also on the eigenvalues. One could envision finding an approximation of $T^*(z, \omega_c)$, where ω_c is the cut-off frequency of the Bessel filter.

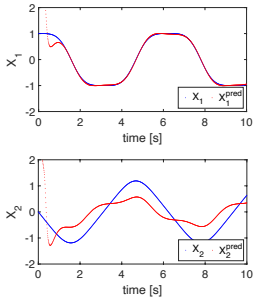
B. Non-autonomous System: Van der Pol

As a first example of a non-autonomous system, we consider now the following Van der Pol oscillator, which is a non-conservative oscillator with nonlinear damping [20]

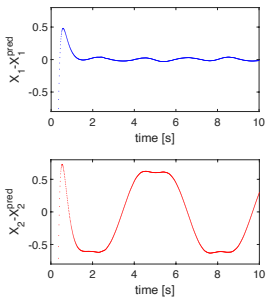
$$f(x) = \begin{cases} \dot{x}_1 = x_2 \\ \dot{x}_2 = \epsilon(1 - x_1^2)x_2 - x_1 + u(t) \end{cases}, \quad y = x_1, \quad (19)$$

where $\epsilon = 1$. Unforced, its trajectories quickly converge to a single asymptotically stable limit cycle, but exhibit chaotic behavior under sinusoidal forcing. These properties perfectly illustrate how the approach described in Section II-B significantly improve on the result of [8, Section IV]. There, it is suggested, for systems of the form (11) to use the autonomous transform corresponding to f only, for an observer with sufficiently fast convergence[2]. This strategy would be extremely difficult to apply to the Van der Pol oscillator, as

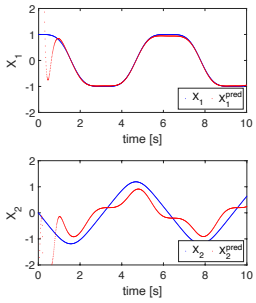
[2]This corresponds to taking $u_0 = 0$ in our approach.



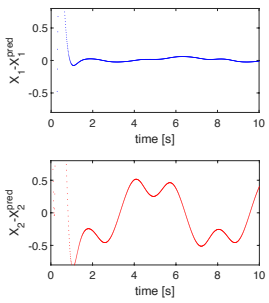
(a) Evolution of the true and estimated states over time, for $(\lambda_1, \lambda_2, \lambda_3) = (5, 6, 7)$.



(b) Evolution of the estimation error over time, for $(\lambda_1, \lambda_2, \lambda_3) = (5, 6, 7)$.



(c) Evolution of the true and estimated states over time, for $(\lambda_1, \lambda_2, \lambda_3)$ the eigenvalues of a Bessel filter with cut-off frequency $2\pi\ rad/s$.



(d) Evolution of the estimation error over time, for $(\lambda_1, \lambda_2, \lambda_3)$ the eigenvalues of a Bessel filter with cut-off frequency $2\pi\ rad/s$.

Fig. 6. Comparison of the autonomous system (eq. (17)) solutions for an observer defined in $\mathbb{R}^3$, with different eigenvalues.

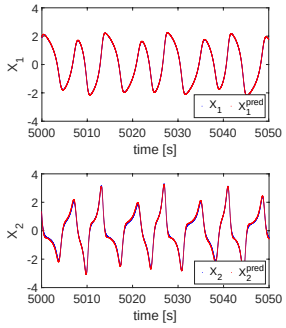
the generated data without forcing would be clustered around the attractive limit cycle. Rather, we apply here, to generate the training set, a linear chirp $u_0(t) = 10^{-3} + 9.99 \times 10^{-5} t$ that makes the system extensively explore the (x_1, x_2)-plane. We then use the corresponding transformation $T^*_{u_0}$ in an observer of the form (12) to estimate the states under a different forcing $u(t) = 0.44 \cos(0.5t)$. We show the results corresponding to an observer z evolving in $\mathbb{R}^3$ and $\mathbb{R}^5$, respectively. The results are depicted in Figure 7.

C. Non-autonomous System: Adding an input to (17)

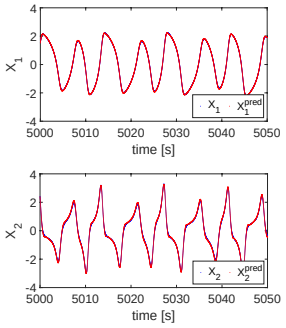
The same approach can be used to generate training data for system (17), by adding an excitation as follows

$$f(x) = \begin{cases} \dot{x}_1 = x_2^3 \\ \dot{x}_2 = -x_1 + u(t) \end{cases}, \quad y = x_1. \tag{20}$$

Then, the methodology presented in section II-B can be applied. First, the training data is generated for one initial condition and a forcing being, again, the linear chirp used in Section IV-B. In this case, the observer is in $\mathbb{R}^3$, we pick a matrix D corresponding to a third-order Bessel filter, with cut-off frequency $\omega = 2\pi\ rad/s$, and $F = \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix}$. The dynamics (17) are then solved using a built-in Matlab variable-step solver over 10^4 seconds. Nonlinear regression is used to find the transformations $(T_{u^0}, T^*_{u^0})$, which are then



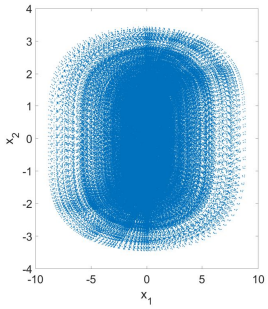
(a) Evolution of the true and predicted state, with an observer $z \in \mathbb{R}^3$.



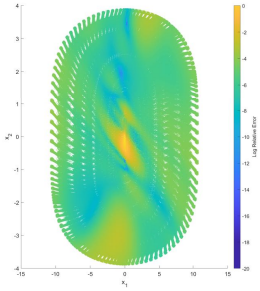
(b) Evolution of the true and predicted state, with an observer $z \in \mathbb{R}^5$.

Fig. 7. Impact of the number of states of the observer on the performance of the design for the Van der Pol oscillator. The plots show the asymptotic convergence performance.

used with an observer of the form (12) with $u = 0$. The results are depicted in Figures 8 and 9.



(a) Training trajectories.



(b) Logarithmic relative error mapping for system with no excitation and $IC \in (-10, 10)$.

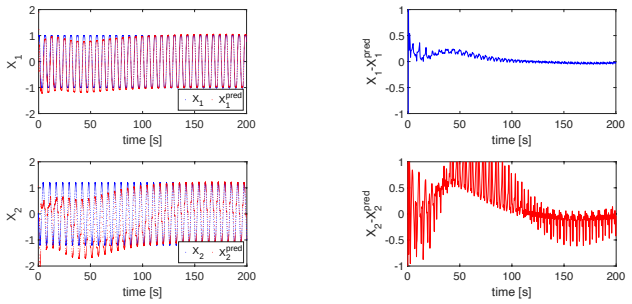
Fig. 8. Training trajectories and estimation error for the artificially excited nonlinear oscillator.

V. Conclusion and Perspectives

This paper is a step towards the combined use of Luenberger observer theory and simulation data-based Machine Learning to systematically design observers for nonlinear systems.

Concerning the application to autonomous systems, the method relies on the very mild assumption of backward distinguishability, which makes it quite general. The main remaining questions are two-fold. First, the generation of relevant data, which is briefly discussed in this paper, should deserve more attention. For instance, one could envision resampling the state-space after a first estimation of the mappings, similarly to mesh refining techniques in numerical simulation. However, such techniques should raise a number of issues regarding the refinement criteria. Second, the appropriate method for nonlinear regression remains a fully open question, and the Neural Network proposed here is by no means a definitive one.

When non-autonomous systems are of interest, the method relies on stronger observability assumptions, somewhat



(a) Trajectories of system (20). (b) Error of prediction.

Fig. 9. Comparison and error of the prediction for trajectory with initial condition $(x_1, x_2) = (1, 0)$ and input $u(t) = 0$.

equivalent to differential observability of the order of the state. This is perhaps unsatisfying but, as is, seems necessary to ensure that the learning can be made for a single excitation, and used with other inputs. Relaxing these assumptions will be the topic of future research. Another interesting aspect is the selection of the appropriate excitation. Indeed, analogously to the autonomous case, the selection criterion used is paramount, with the added possibility here of steering the system towards regions of uncertainty. This problem is linked to active learning [21] and, more generally, input selection for identification [22], [23].

ACKNOWLEDGMENT

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant Agreement nr 766264. L.F. Figueira da Silva was on leave from the Institute Pprime (CNRS, France), and received support by Conselho Nacional de Desenvolvimento Científico e Tecnológico (CNPq grant No. 403904/2016-1).

REFERENCES

[1] A. H. Jazwinski, *Stochastic processes and filtering theory*. Courier Corporation, 2007.

[2] J. Gauthier and G. Bornard, "Observability for any u (t) of a class of nonlinear systems," *IEEE Transactions on Automatic Control*, vol. 26, no. 4, pp. 922–926, 1981.

[3] P. Bernard, *Observer design for nonlinear systems*. Springer, 2019, vol. 479.

[4] D. Luenberger, "Observers for multivariable systems," *IEEE Transactions on Automatic Control*, vol. 11, no. 2, pp. 190–197, 1966.

[5] A. Shoshitaishvili, "On control branching systems with degenerate linearization," *Proc. IFAC Symp. Nonlinear Control Syst.*, pp. 495–500, 1992.

[6] N. Kazantzis and C. Kravaris, "Nonlinear observer design using lyapunov's auxiliary theorem," *Systems & Control Letters*, vol. 34, no. 5, pp. 241–247, 1998.

[7] V. Andrieu and L. Praly, "On the existence of a kazantzis–kravaris/luenberger observer," *SIAM J. Control and Optimization*, vol. 45, pp. 432–456, 02 2006.

[8] P. Bernard and V. Andrieu, "Luenberger observers for nonautonomous nonlinear Systems," *IEEE Transactions on Automatic Control*, vol. 64, no. 1, pp. 270–281, Jan. 2019. [Online]. Available: https://hal-mines-paristech.archives-ouvertes.fr/hal-02429235

[9] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, pp. 436–44, May 2015.

[10] M. I. Quraishi, J. P. Choudhury, and M. De, "Image recognition and processing using artificial neural network," in *2012 1st International Conference on Recent Advances in Information Technology (RAIT)*. IEEE, 2012, pp. 95–100.

[11] J. Fu, H. Zheng, and T. Mei, "Look closer to see better: Recurrent attention convolutional neural network for fine-grained image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 4438–4446.

[12] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," *Commun. ACM*, vol. 60, no. 6, p. 84–90, May 2017. [Online]. Available: https://doi.org/10.1145/3065386

[13] Y. Goldberg, "A primer on neural network models for natural language processing," *J. Artif. Int. Res.*, vol. 57, no. 1, p. 345–420, Sep. 2016.

[14] D. H. Nguyen and B. Widrow, "Neural networks for self-learning control systems," *IEEE Control Systems Magazine*, vol. 10, no. 3, pp. 18–23, 1990.

[15] E. Diaconescu, "The use of narx neural networks to predict chaotic time series," *WSEAS Transactions on Computer Research*, vol. 3, 03 2008.

[16] S. Pan and K. Duraisamy, "Long-time predictive modeling of nonlinear dynamical systems using neural networks," *Complexity*, vol. 2018, 2018.

[17] M. A. Nielsen, "Neural networks and deep learning," 2018. [Online]. Available: http://neuralnetworksanddeeplearning.com/

[18] Hecht-Nielsen, "Theory of the backpropagation neural network," in *International 1989 Joint Conference on Neural Networks*, 1989, pp. 593–605 vol.1.

[19] 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, MINES ParisTech, PSL University, 2014.

[20] T. Kanamaru, "Van der Pol oscillator," *Scholarpedia*, vol. 2, no. 1, p. 2202, 2007, revision #138698.

[21] M. Buisson-Fenet, F. Solowjow, and S. Trimpe, "Actively learning gaussian process dynamics," 2019.

[22] L. Ljung, "System identification: theory for the user," *PTR Prentice Hall, Upper Saddle River, NJ*, pp. 1–14, 1999.

[23] E. Walter and L. Pronzato, "Identification of parametric models," *Communications and control engineering*, vol. 8, 1997.